Package: manydist (via r-universe)

June 10, 2026

Type Package

Title Distance-Based Learning for Mixed-Type Data

Version 0.5.0

Description Provides tools for constructing, computing, and using distance measures for numerical, categorical, and mixed-type data. The package implements a flexible framework in which continuous and categorical components can be combined under additive, commensurable, and association-aware specifications. Supported methods include classical distances such as Gower, Euclidean, Manhattan, and Mahalanobis-type distances; categorical dissimilarities such as simple matching, occurrence-frequency, and association-based measures; and mixed-type presets designed to reduce biases due to variable type, scale, distribution, redundancy, and number of categories. The package also provides scaling options, supervised and unsupervised distance constructions, leave-one-variable-out tools for distance-based variable importance, and integration with distance-based learning workflows such as nearest-neighbour prediction, partitioning around medoids, and spectral clustering. Methods are motivated by van de Velden, Iodice D'Enza, Markos, and Cavicchia (2026) <doi:10.1080/10618600.2026.2680181> and related work on categorical and mixed-type dissimilarities.

Imports aricode, cluster, clusterGeneration, data.table, dials, distances, dplyr, entropy, fastDummies, forcats, fpc, generics, ggplot2, kdml, magrittr, Matrix, parsnip, philentropy, purrr, readr, recipes, Rfast, rlang, rsample, stats, tibble, tidyr, tidyselect, tune

Depends R (>= 4.5.0)

Suggests arules, clustMixType, FD, klaR, mclust, palmerpenguins, parallelDist, StatMatch, workflows

License GPL-3

Encoding UTF-8

LazyData true

NeedsCompilation no

Config/roxygen2/version 8.0.0

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

Maintainer Alfonso Iodice D'Enza <iodicede@unina.it>

Config/pak/sysreqs cmake make libicu-dev libuv1-dev libx11-dev

Repository https://alfonsoiodicede.r-universe.dev

Date/Publication 2026-06-09 21:40:08 UTC

RemoteUrl https://github.com/cran/manydist

RemoteRef HEAD

RemoteSha 16e6ceab71e3faecb805633e3f7806249f1782f6

Contents

`compare_lovo_mdist` *Compare LOVO diagnostics across multiple distance specifications*

Description

This function compares **leave-one-variable-out (LOVO)** diagnostics across multiple distance definitions supported by `manydist`.

Usage

```
compare_lovo_mdist(
  x,
  methods,
  dims = 2,
  keep_dist = FALSE,
  .progress = FALSE,
  ...
)
```

Arguments

x	A data frame or tibble containing the predictors.

methods	A **named list** describing the distance specifications to compare. Each element must be a list of arguments passed to lovo_mdist.
	For example:
	`methods = list(`
	`  gower = list(preset = "gower"),`
	`  u_dep = list(preset = "unbiased_dependent"),`
	`  custom = list(`
	`  preset = "custom",`
	`  method_cat = "matching",`
	`  method_num = "std",`
	`  commensurable = TRUE`
	`  )`
	`)`

dims	Number of dimensions used for the MDS configuration when computing congruence-based diagnostics.
keep_dist	Logical; if TRUE, distance matrices from the LOVO computations are retained. This increases memory usage.
.progress	Logical; if TRUE, progress messages are printed while computing LOVO diagnostics for each method.
...	Additional arguments passed to lovo_mdist unless overridden in the method-specific argument list. These may include optional clustering diagnostics, for example cluster_k, cluster_methods, and hclust_method.

Details

For each distance specification, the function:

1. Computes the full mixed-type distance using mdist().
2. Recomputes the distance repeatedly leaving out one variable at a time.
3. Measures the impact of each variable using metrics such as mean absolute deviation (MAD), congruence-based diagnostics, and, when requested, clustering-based agreement measures.

The results are combined across methods and returned as an `MDistLOVOCompare` object, which supports `print()`, `summary()`, and `ggplot2::autoplot()`.

Value

An object of class `MDistLOVOCompare` containing:

results A tibble with one row per method-variable combination.

methods The list of distance specifications used.

dims Number of MDS dimensions used.

n_obs Number of observations in the dataset.

See Also

`lovo_mdist`, `mdist`

Examples

```
## Not run:
library(manydist)
library(palmerpenguins)

data <- penguins |>
  dplyr::select(-species) |>
  tidyr::drop_na()

cmp <- compare_lovo_mdist(
  x = data,
  methods = list(
    gower = list(preset = "gower"),
    u_dep = list(preset = "unbiased_dependent")
  ),
  cluster_k = 3
)

summary(cmp)
autoplot(cmp, metric = "mad_importance")
autoplot(cmp, metric = "pam_importance")

## End(Not run)
```

---

congruence_coeff *Congruence coefficient between two configurations*

Description

Computes the congruence coefficient between two data configurations using the Frobenius inner product of their pairwise distance matrices.

Usage

```
congruence_coeff(L1, L2)
```

Arguments

L1	A numeric matrix or data frame (rows = observations)
L2	A numeric matrix or data frame with the same number of rows as L1

Details

The congruence coefficient is defined as

$$\frac{\langle D_1, D_2 \rangle_F}{\sqrt{\langle D_1, D_1 \rangle_F \langle D_2, D_2 \rangle_F}}$$

where D_1 and D_2 are the pairwise distance matrices derived from L1 and L2.

Value

A scalar in $[-1, 1]$ measuring similarity between the two configurations.

---

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)
```

gen_mixed *Generate mixed-type clustering data with signal and noise*

Description

Generates synthetic mixed datasets with controllable numerical and categorical signal/noise structure and balanced cluster sizes.

Usage

```
gen_mixed(
  k_true,
  clustSizeEq = 50,
  numsignal = 2,
  numnoise = 2,
  catsignal = 2,
  catnoise = 2,
  q = 5,
  q_err = 9,
  numsep = 0.1,
  catsep = 0.5,
  seed = NULL,
  error_type = c("normal", "chisq"),
  error_df = 2,
  error_scale = 1
)
```

Arguments

k_true	Number of clusters
clustSizeEq	Observations per cluster
numsignal	Number of numerical signal variables
numnoise	Number of numerical noise variables
catsignal	Number of categorical signal variables
catnoise	Number of categorical noise variables
q	Number of categories for signal categorical variables
q_err	Number of categories for categorical noise
numsep	Separation for numerical signal
catsep	Separation for categorical signal
seed	Optional seed
error_type	Error distribution ("normal" or "chisq")
error_df	Degrees of freedom for chi-square noise
error_scale	Scale of noise

Value

A list with elements `df`, `X_num`, `X_cat`, `y`, `num_cols`, `cat_cols`

`generate_dataset` *generate toy mixed datasets for the supplementary material: figures 1 to 3*

Description

generate toy mixed datasets for the supplementary material: figures 1 to 3

Usage

```
generate_dataset(
  n,
  porig,
  pn,
  pnnoise,
  pcnoise,
  sigma,
  qoptions,
  seed = NULL,
  mode = "per_variable"
)
```

Arguments

`n`	number of observations
`porig`	number of original informative continuous variables
`pn`	number of continuous variables (total, before adding noise variables)
`pnnoise`	number of extra numeric noise variables
`pcnoise`	number of extra categorical noise variables
`sigma`	sd for noise added to informative variables
`qoptions`	number of bins for categorization (vector if per_variable, scalar if shared)
`seed`	optional seed
`mode`	either "per_variable" or "shared"

`lovo_mdist` *Leave-one-variable-out diagnostics for distance-based variable importance*

Description

Computes leave-one-variable-out (LOVO) diagnostics for a distance specification. The function first computes the full dissimilarity matrix using [mdist()]. It then removes one predictor at a time, recomputes the dissimilarity matrix, and compares each leave-one-variable-out matrix with the full one.

Usage

```
lovo_mdist(
  x,
  response = NULL,
  ...,
  dims = 2,
  keep_dist = FALSE,
  cluster_k = NULL,
  cluster_methods = c("pam", "hclust", "spectral"),
  hclust_method = "average",
  spectral_sigma = NULL,
  spectral_nstart = 50,
  response_used = TRUE
)
```

Arguments

`x`	A data frame or object coercible to a tibble. Rows are observations and columns are variables used to compute the dissimilarity.
`response`	Optional response variable. It can be supplied as an unquoted column name or as a character string. When supplied and ‘response_used = TRUE‘, it is passed to [mdist()] for response-aware distance construction. The response column is not treated as a predictor in the leave-one-variable-out loop.
`...`	Additional arguments passed to [mdist()], such as ‘preset‘, ‘method_cat‘, ‘method_num‘, ‘commensurable‘, or ‘interaction‘.
`dims`	Integer. Number of dimensions used by classical multidimensional scaling when computing congruence-based diagnostics.
`keep_dist`	Logical. If ‘TRUE‘, store the full dissimilarity matrix and all leave-one-variable-out dissimilarity matrices in the returned object.
`cluster_k`	Optional integer. Number of clusters used when computing clustering-based LOVO diagnostics. If ‘NULL‘, clustering diagnostics are not computed.
`cluster_methods`	Character vector specifying the clustering methods used for clustering-based diagnostics. Possible values are ‘"pam"‘, ‘"hclust"‘, and ‘"spectral"‘.

`hclust_method` Character string specifying the linkage method passed to [stats::hclust()] when '"hclust"' is included in 'cluster_methods'.

`spectral_sigma` Optional numeric value for the Gaussian affinity bandwidth used by spectral clustering. If 'NULL', the default used by [spectral_dist()] is applied.

`spectral_nstart` Integer. Number of random starts used by the k-means step in spectral clustering.

`response_used` Logical. If 'TRUE', the response variable, when supplied, is used in the distance construction. If 'FALSE', the response column is removed before computing distances.

Details

'lovo_mdist()' is useful for assessing how strongly each predictor contributes to a distance-based representation. A predictor is considered influential when removing it produces a large change in the dissimilarity matrix, the multidimensional scaling configuration, or an optional clustering partition.

The returned object contains several LOVO diagnostics. The main distance contribution is measured by the mean absolute difference between the full dissimilarity matrix and each leave-one-variable-out matrix ('mad_importance'). The normalized version is stored as 'relative_distance'.

The function also compares classical multidimensional scaling configurations computed from the full and leave-one-variable-out dissimilarities. These diagnostics are stored as 'mds_congruence' / 'cc_importance' and 'ac_importance', the latter corresponding to an alienation coefficient.

If 'cluster_k' is supplied, the function additionally computes clustering partitions from the full and leave-one-variable-out dissimilarities and compares them using the adjusted Rand index. The corresponding importance measures are defined as '1 - ARI' and are stored as 'pam_importance', 'hclust_importance', or 'spectral_importance', depending on the selected clustering methods.

Clustering-based diagnostics require the suggested package 'mclust'.

Value

An object of class '"MDistLOVO"'. The main results are stored in the '$results' field as a tibble with one row per left-out variable. The object also has print, summary, and autoplot methods.

See Also

[mdist()], [compare_lovo_mdist()], [spectral_dist()]

Examples

```
if (requireNamespace("palmerpenguins", quietly = TRUE)) {
  data("penguins", package = "palmerpenguins")

  penguins_small <- palmerpenguins::penguins |>
    dplyr::select(
      species, bill_length_mm, bill_depth_mm, flipper_length_mm,
      body_mass_g, island, sex
    ) |>
```

```
    tidyr::drop_na()

  # LOVO diagnostics for a Gower distance
  res <- lovo_mdist(
    penguins_small,
    preset = "gower",
    response = species,
    response_used = FALSE
  )

  res
  summary(res)

  # Plot the relative distance contribution of each predictor
  p <- res$autoplot(metric = "relative_distance", reorder = TRUE)
  p
}
```

`lovo_method_spec` *Create a LOVO method specification for 'compare_lovo_mdist()'*

Description

Helper to build one method specification for [compare_lovo_mdist()]. This allows tidy-style specification of 'response', e.g. 'response = Name', while storing the method definition as a regular list.

Usage

```
lovo_method_spec(response = NULL, ...)
```

Arguments

`response`	Optional response column, supplied either unquoted (e.g. 'Name') or quoted (e.g. '"Name"').
`...`	Additional arguments passed on to [lovo_mdist()] through [compare_lovo_mdist()].

Value

A named list of arguments suitable for one element of the 'methods' argument in [compare_lovo_mdist()].

Examples

```
## Not run:
methods <- list(
  tvd_sup = lovo_method_spec(
    response = species,
    preset = "custom",
```

```
    method_cat = "tvd",
    method_num = "std",
    commensurable = TRUE,
    response_used = TRUE
  ),
  tvd_unsup = lovo_method_spec(
    response = species,
    preset = "custom",
    method_cat = "tvd",
    method_num = "std",
    commensurable = TRUE,
    response_used = FALSE
  )
)

## End(Not run)
```

`make_mdist_recipe` *Build a recipe that computes mdist-based distance features*

Description

This helper creates a tidymodels recipe using step_mdist(). It supports both mdist presets and custom param sets.

Usage

```
make_mdist_recipe(df, mdist_type, mdist_preset, param_set, outcome)
```

Arguments

`df`	A data frame.
`mdist_type`	"preset" or "custom".
`mdist_preset`	Name of the preset (if mdist_type == "preset").
`param_set`	A list of custom mdist arguments (if mdist_type == "custom").
`outcome`	Name of the outcome variable.

Value

A ‘recipes::recipe()‘ object.

mdist *Mixed-type dissimilarities for distance-based learning*

Description

Computes a dissimilarity object for numerical, categorical, or mixed-type data. The function combines continuous and categorical components according to either a predefined ‘preset‘ or a user-defined custom specification.

Usage

```
mdist(
  x,
  new_data = NULL,
  response = NULL,
  method_cat = "tvd",
  method_num = "std",
  commensurable = TRUE,
  ncomp = NULL,
  threshold = NULL,
  preset = "custom",
  interaction = FALSE,
  prop_nn = 0.1,
  score = "ba",
  decision = "prior_corrected",
  gower_average = TRUE
)
```

Arguments

x	A data frame or matrix containing the training observations. Columns can be numeric, factors, or a mixture of both.
new_data	Optional data frame or matrix containing new observations. If supplied, distances are computed from rows of ‘new_data‘ to rows of ‘x‘, producing a rectangular test-to-training dissimilarity matrix.
response	Optional response variable used for response-aware categorical dissimilarities. It can be supplied as an unquoted column name or as a character string. The response column is removed from the predictors before computing distances.
method_cat	Character string specifying the categorical-variable dissimilarity used when ‘preset = "custom"‘. Common values include ‘"matching"‘ and ‘"tvd"‘. Use [all_dist_method_specs()] to inspect available methods.
method_num	Character string specifying the numerical-variable preprocessing used when ‘preset = "custom"‘. Available options include ‘"none"‘ for no preprocessing, ‘"std"‘ for standard-deviation scaling, ‘"range"‘ for range scaling, ‘"robust"‘ for interquartile-range-based scaling, and ‘"pc_scores"‘ for principal-component score scaling.

commensurable	Logical. If ‘TRUE‘, dissimilarities are scaled so that the average contribution of each variable to the overall distance is equal to 1.
ncomp	Integer or ‘NULL‘. Number of principal components to retain when ‘method_num = "pc_scores"‘. If ‘NULL‘, all available components are used unless ‘threshold‘ is supplied and supported by the underlying method.
threshold	Numeric or ‘NULL‘. Optional cumulative variance threshold used when ‘method_num = "pc_scores"‘.
preset	Character string specifying a predefined distance specification. Available values include ‘"custom"‘, ‘"gower"‘, ‘"unbiased_dependent"‘, ‘"u_dep"‘, ‘"u_indep"‘, ‘"u_mix"‘, ‘"hl"‘, ‘"gudmm"‘, ‘"dkss"‘, ‘"mod_gower"‘, and ‘"euclidean"‘. When ‘preset‘ is not ‘"custom"‘, arguments such as ‘method_cat‘, ‘method_num‘, ‘commensurable‘, and ‘interaction‘ are handled by the preset and user-supplied values for those arguments are ignored.
interaction	Logical. If ‘TRUE‘, adds an interaction-aware continuous-categorical component based on local predictive separability.
prop_nn	Numeric. Proportion of nearest neighbours used when ‘interaction = TRUE‘.
score	Character string specifying the score used when ‘interaction = TRUE‘. Available values include ‘"ba"‘ for balanced accuracy and ‘"logloss"‘.
decision	Character string specifying the decision rule used when ‘score = "ba"‘. The default is ‘"prior_corrected"‘.
gower_average	Logical; only used when ‘preset = "gower"‘. If ‘TRUE‘, returns the standard Gower dissimilarity averaged over variables, matching the scale of [cluster::daisy()] with ‘metric = "gower"‘. If ‘FALSE‘, returns the sum of per-variable Gower contributions, equivalent to multiplying the averaged Gower dissimilarity by the number of active variables.

Details

‘mdist()‘ is the main distance-construction function in ‘manydist‘. It can return ordinary train-train dissimilarities or rectangular test-to-training dissimilarities when ‘new_data‘ is supplied. The resulting object stores both the dissimilarity matrix and metadata about the distance specification that was used.

With ‘preset = "custom"‘, users manually choose the numerical preprocessing, categorical dissimilarity, commensurability, and optional interaction term.

The ‘"gower"‘ preset follows the usual Gower construction based on range scaling for continuous variables and matching dissimilarities for categorical variables. The ‘gower_average‘ argument controls whether the result is averaged over variables or returned as a sum of variable-wise contributions.

The ‘"u_dep"‘, ‘"unbiased_dependent"‘, ‘"u_indep"‘, and ‘"u_mix"‘ presets are convenience specifications for unbiased or commensurable mixed-variable dissimilarities. The ‘"euclidean"‘ preset computes a Euclidean distance after one-hot encoding categorical variables. The ‘"gudmm"‘, ‘"dkss"‘, and ‘"mod_gower"‘ presets provide additional mixed-type distance constructions. Some presets currently support only train-train distances and will stop if ‘new_data‘ is supplied.

Use [all_dist_method_specs()] to inspect the available distance components and method specifications.

Value

An object of class ‘"MDist"‘. The object contains the computed dissimilarity in its ‘$distance‘ field, the selected ‘preset‘, the training data, and a list of parameters describing the fitted distance specification. Square train-train dissimilarities are stored as ‘"dissimilarity"‘/‘"dist"‘ objects; rectangular test-to-training dissimilarities are stored as ‘"dissimilarity"‘/‘"matrix"‘ objects.

See Also

[step_mdist()], [all_dist_method_specs()]

Examples

```
if (requireNamespace("palmerpenguins", quietly = TRUE)) {
  data("penguins", package = "palmerpenguins")

  penguins_small <- palmerpenguins::penguins |>
    dplyr::select(
      bill_length_mm, bill_depth_mm, flipper_length_mm,
      body_mass_g, species, island, sex
    ) |>
    tidyr::drop_na()

  # Gower distance on mixed-type data
  d_gower <- mdist(penguins_small, preset = "gower")
  d_gower

  # Custom mixed-type specification
  d_custom <- mdist(
    penguins_small,
    preset = "custom",
    method_cat = "matching",
    method_num = "std",
    commensurable = TRUE
  )

  d_custom

  # Train-to-new-data distances
  penguin_split <- rsample::initial_split(penguins_small, prop = 0.75)
  penguin_train <- rsample::training(penguin_split)
  penguin_test  <- rsample::testing(penguin_split)

  d_new <- mdist(
    penguin_train,
    new_data = penguin_test,
    preset = "gower"
  )

  d_new
}
```

mdist_summary_impl *Internal: summary method implementation for MDist R6 objects*

Description

Internal: summary method implementation for MDist R6 objects

Usage

```
mdist_summary_impl(object, ...)
```

Arguments

object	An 'MDist' object to summarize.
...	Additional arguments currently not used.

Value

Invisibly returns 'object'.

pam_dist *PAM clustering specification based on manydist dissimilarities*

Description

PAM clustering specification based on manydist dissimilarities

Usage

```
pam_dist(num_clusters = NULL)
```

Arguments

num_clusters	Number of clusters.

Value

A 'pam_dist_spec' object.

spectral_dist *Spectral clustering specification based on manydist dissimilarities*

Description

Spectral clustering specification based on manydist dissimilarities

Usage

```
spectral_dist(num_clusters = NULL, sigma = NULL, nstart = 50)
```

Arguments

num_clusters	Number of clusters.
sigma	Optional bandwidth for the Gaussian affinity. If 'NULL', the median pairwise distance is used.
nstart	Number of random starts for k-means.

Value

A 'spectral_dist_spec' object.

Examples

```
## Not run:
library(manydist)
library(palmerpenguins)
library(recipes)
library(generics)

data <- penguins |>
  dplyr::select(-species) |>
  tidyr::drop_na()

rec <- recipes::recipe(~ ., data = data) |>
  step_mdist(all_predictors(), preset = "gower", output = "pairwise")

spec <- spectral_dist(num_clusters = 3)

fit_obj <- generics::fit(spec, recipe = rec, data = data)

print(fit_obj)
predict(fit_obj)
predict(fit_obj, type = "embed")

## End(Not run)
```

spectral_from_dist *Spectral clustering from a distance matrix*

Description

Spectral clustering from a distance matrix

Usage

```
spectral_from_dist(D, k, affinity_method = "selftune")
```

Arguments

D	A distance matrix
k	Number of clusters
affinity_method	Method to build affinity

Value

A vector of cluster labels

step_mdist *Add manydist dissimilarities to a recipe*

Description

‘step_mdist()‘ is a [recipes::recipe()] step that replaces selected predictors by a distance-based representation computed with [mdist()]. It is designed for distance-based learning workflows, especially nearest-neighbour prediction and clustering models that operate on dissimilarity matrices.

Usage

```
step_mdist(
  recipe,
  ...,
  role = "predictor",
  trained = FALSE,
  output = "distance_to_training",
  preset = "custom",
  method_cat = "tot_var_dist",
  method_num = "none",
  commensurable = FALSE,
  ncomp = NULL,
  threshold = NULL,
```

```
  columns = NULL,
  train_predictors = NULL,
  preprocessor = NULL,
  skip = FALSE,
  id = recipes::rand_id("mdist")
)
```

Arguments

recipe	A recipe object.
...	Selector(s) for the predictor columns to be used in [mdist()]. These are passed to [recipes::recipes_eval_select()] during preparation.
role	Role for the new distance columns. The default is ‘"predictor"‘.
trained	Logical for recipes internals. Do not set manually.
output	Character string specifying the type of distance output. ‘"distance_to_training"‘ returns distances from the baked data to the training observations and is the usual choice for prediction workflows. ‘"pairwise"‘ returns the within-training pairwise dissimilarity matrix and is intended for training-only distance-based clustering workflows.
preset	Character string specifying the distance preset passed to [mdist()]. Available values include ‘"custom"‘, ‘"gower"‘, ‘"unbiased_dependent"‘, ‘"u_dep"‘, ‘"u_indep"‘, ‘"u_mix"‘, ‘"hl"‘, ‘"gudmm"‘, ‘"dkss"‘, ‘"mod_gower"‘, and ‘"euclidean"‘.
method_cat	Character string specifying the categorical-variable dissimilarity passed to [mdist()] when ‘preset = "custom"‘. Common values include ‘"matching"‘ and ‘"tvd"‘. Use [all_dist_method_specs()] to inspect available methods.
method_num	Character string specifying the numerical-variable preprocessing passed to [mdist()] when ‘preset = "custom"‘. Available options include ‘"none"‘ for no preprocessing, ‘"std"‘ for standard-deviation scaling, ‘"range"‘ for range scaling, ‘"robust"‘ for inter-quartile-range-based scaling, and ‘"pc_scores"‘ for principal-component score scaling.
commensurable	Logical. If ‘TRUE‘, dissimilarities are scaled so that the average contribution of each variable to the overall distance is equal to 1, when supported by the selected distance specification.
ncomp	Integer or ‘NULL‘. Number of principal components to retain when ‘method_num = "pc_scores"‘. If ‘NULL‘, all available components are used unless ‘threshold‘ is supplied and supported by the underlying method.
threshold	Numeric or ‘NULL‘. Optional cumulative variance threshold used when ‘method_num = "pc_scores"‘.
columns	Names of columns selected at prep time. Used internally by recipes.
train_predictors	Training predictors stored at prep time. Used internally by recipes to compute distances from new observations to the training observations.
preprocessor	Internal fitted manydist preprocessor.
skip	Logical. Standard recipes argument indicating whether the step should be skipped when baking new data.
id	Character string. Unique step identifier.

Details

The step can produce either distances from new observations to the training observations, or the within-training pairwise dissimilarity matrix. The former is the usual choice for supervised prediction workflows; the latter is useful for distance-based clustering workflows fitted on the training data.

During [recipes::prep()], ‘step_mdist()‘ stores the selected training predictors and fits the internal manydist preprocessor. During [recipes::bake()], the selected predictors are removed and replaced by distance columns named ‘dist_1‘, ‘dist_2‘, and so on.

With ‘output = "distance_to_training"‘, baking the training data returns the training pairwise distances, while baking new data returns distances from each new observation to each training observation. This rectangular representation is suitable for nearest-neighbour prediction models.

With ‘output = "pairwise"‘, the step returns the within-training pairwise dissimilarity matrix. Baking genuinely new data is not supported in this mode, because the output is intended for training-only clustering workflows such as [pam_dist()] or [spectral_dist()].

Value

An updated recipe with a manydist step.

See Also

[mdist()], [nearest_neighbor_dist()], [pam_dist()], [spectral_dist()], [all_dist_method_specs()]

Examples

```
if (requireNamespace("palmerpenguins", quietly = TRUE)) {
  data("penguins", package = "palmerpenguins")

  penguins_small <- palmerpenguins::penguins |>
    dplyr::select(
      species, bill_length_mm, bill_depth_mm, flipper_length_mm,
      body_mass_g, island, sex
    ) |>
    tidyr::drop_na()

  # Distance-to-training representation for prediction workflows
  rec <- recipes::recipe(species ~ ., data = penguins_small) |>
    step_mdist(
      recipes::all_predictors(),
      preset = "gower",
      output = "distance_to_training"
    )

  rec_prep <- recipes::prep(rec, training = penguins_small)
  baked <- recipes::bake(rec_prep, new_data = penguins_small)

  baked |> dplyr::slice_head(n=5)

  # Pairwise representation for clustering workflows
  rec_pairwise <- recipes::recipe(~ ., data = penguins_small) |>
```

```
    step_mdist(
      recipes::all_predictors(),
      preset = "gower",
      output = "pairwise"
    )

  rec_pairwise_prep <- recipes::prep(rec_pairwise, training = penguins_small)
  pairwise_dist <- recipes::bake(rec_pairwise_prep, new_data = penguins_small)

  pairwise_dist
}
```

Index