

Package: rdist (via r-universe)

October 26, 2026

Title Calculate Pairwise Distances

Version 0.0.6

Description A common framework for calculating distance matrices.

Depends R (>= 3.2.2)

License GPL

URL <https://github.com/blasern/rdist>

BugReports <https://github.com/blasern/rdist/issues>

Encoding UTF-8

LinkingTo Rcpp, RcppArmadillo

Imports Rcpp, methods

RoxygenNote 7.2.3

Suggests testthat

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

Date/Publication 2026-09-17 12:35:16 UTC

RemoteUrl <https://github.com/blasern/rdist>

RemoteRef HEAD

RemoteSha bf40be0018cd53025943a193a5b39831231680f3

Contents

farthest_point_sampling	2
is_metric	3
product_metric	3
rdist	4

Index	6
--------------	----------

`farthest_point_sampling`*Farthest point sampling*

Description

Farthest point sampling returns a reordering of the metric space $P = \{p_1, \dots, p_k\}$, such that each p_i is the farthest point from the first $i - 1$ points.

Usage

```
farthest_point_sampling(  
  mat,  
  metric = "precomputed",  
  k = nrow(mat),  
  initial_point_index = 1L,  
  return_clusters = FALSE  
)
```

Arguments

<code>mat</code>	Original distance matrix
<code>metric</code>	Distance metric to use (either "precomputed" or a metric from rdist)
<code>k</code>	Number of points to sample
<code>initial_point_index</code>	Index of p_1
<code>return_clusters</code>	Should the indices of the closest farthest points be returned?

Examples

```
# generate data  
df <- matrix(runif(200), ncol = 2)  
dist_mat <- pdist(df)  
# farthest point sampling  
fps <- farthest_point_sampling(dist_mat)  
fps2 <- farthest_point_sampling(df, metric = "euclidean")  
all.equal(fps, fps2)  
# have a look at the fps distance matrix  
rdist(df[fps[1:5], ])  
dist_mat[fps, fps][1:5, 1:5]
```

is_metric	<i>Metric and triangle inequality</i>
-----------	---------------------------------------

Description

Does the distance matrix come from a metric

Usage

```
is_distance_matrix(mat, tolerance = .Machine$double.eps^0.5)

triangle_inequality(mat, tolerance = .Machine$double.eps^0.5)
```

Arguments

mat	The matrix to evaluate
tolerance	Differences smaller than tolerance are not reported.

Examples

```
data <- matrix(rnorm(20), ncol = 2)
dm <- pdist(data)
is_distance_matrix(dm)
triangle_inequality(dm)

dm[1, 2] <- 1.1 * dm[1, 2]
is_distance_matrix(dm)
```

product_metric	<i>Product metric</i>
----------------	-----------------------

Description

Returns the p-product metric of two metric spaces. Works for output of 'rdist', 'pdist' or 'cdist'.

Usage

```
product_metric(..., p = 2)
```

Arguments

...	Distance matrices or dist objects
p	The power of the Minkowski distance

Examples

```
# generate data
df <- matrix(runif(200), ncol = 2)
# distance matrices
dist_mat <- pdist(df)
dist_1 <- pdist(df[, 1])
dist_2 <- pdist(df[, 2])
# product distance matrix
dist_prod <- product_metric(dist_1, dist_2)
# check equality
all.equal(dist_mat, dist_prod)
```

rdist	<i>rdist: an R package for distances</i>
-------	--

Description

rdist provide a common framework to calculate distances. There are three main functions:

- `rdist` computes the pairwise distances between observations in one matrix and returns a `dist` object,
- `pdist` computes the pairwise distances between observations in one matrix and returns a `matrix`, and
- `cdist` computes the distances between observations in two matrices and returns a `matrix`.

In particular the `cdist` function is often missing in other distance functions. All calculations involving NA values will consistently return NA.

Usage

```
rdist(X, metric = "euclidean", p = 2L)
```

```
pdist(X, metric = "euclidean", p = 2)
```

```
cdist(X, Y, metric = "euclidean", p = 2)
```

Arguments

<code>X, Y</code>	A matrix
<code>metric</code>	The distance metric to use
<code>p</code>	The power of the Minkowski distance

Details

Available distance measures are (written for two vectors v and w):

- "euclidean": $\sqrt{\sum_i (v_i - w_i)^2}$
- "minkowski": $(\sum_i |v_i - w_i|^p)^{1/p}$
- "manhattan": $\sum_i (|v_i - w_i|)$
- "maximum" or "chebyshev": $\max_i (|v_i - w_i|)$
- "canberra": $\sum_i (\frac{|v_i - w_i|}{|v_i| + |w_i|})$
- "angular": $\cos^{-1}(\text{cor}(v, w))$
- "correlation": $\sqrt{\frac{1 - \text{cor}(v, w)}{2}}$
- "absolute_correlation": $\sqrt{1 - |\text{cor}(v, w)|^2}$
- "hamming": $(\sum_i v_i \neq w_i) / \sum_i 1$
- "jaccard": $(\sum_i v_i \neq w_i) / \sum_i 1_{v_i \neq 0 \cup w_i \neq 0}$
- Any function that defines a distance between two vectors.

Index

`cdist(rdist)`, [4](#)

`farthest_point_sampling`, [2](#)

`is_distance_matrix(is_metric)`, [3](#)
`is_metric`, [3](#)

`pdist(rdist)`, [4](#)
`product_metric`, [3](#)

`rdist`, [2](#), [4](#)
`rdist-package(rdist)`, [4](#)

`triangle_inequality(is_metric)`, [3](#)