

# Package ‘philentropy’

February 13, 2019

**Type** Package

**Title** Similarity and Distance Quantification Between Probability Functions

**Version** 0.3.0

**Date** 2019-02-12

**Maintainer** Hajk-Georg Drost <hgd23@cam.ac.uk>

**Description** Computes 46 optimized distance and similarity measures for comparing probability functions (Drost (2018) <doi:10.21105/joss.00765>). These comparisons between probability functions have their foundations in a broad range of scientific disciplines from mathematics to ecology. The aim of this package is to provide a core framework for clustering, classification, statistical inference, goodness-of-fit, non-parametric statistics, information theory, and machine learning tasks that are based on comparing univariate or multivariate probability functions.

**Depends** R (>= 3.1.2)

**Imports** Rcpp, dplyr, KernSmooth

**License** GPL-2

**LazyData** true

**LinkingTo** Rcpp

**URL** <https://github.com/HajkD/philentropy>

**Suggests** testthat, knitr

**VignetteBuilder** knitr

**BugReports** <https://github.com/HajkD/philentropy/issues>

**RoxygenNote** 6.1.1

**NeedsCompilation** yes

**Author** Hajk-Georg Drost [aut, cre] (<<https://orcid.org/0000-0002-1567-306X>>)

**Repository** CRAN

**Date/Publication** 2019-02-13 11:50:33 UTC

**R topics documented:**

additive_symm_chi_sq . . . . .	3
avg . . . . .	4
bhattacharyya . . . . .	4
binned.kernel.est . . . . .	5
canberra . . . . .	6
CE . . . . .	6
chebyshev . . . . .	8
clark_sq . . . . .	8
cosine_dist . . . . .	9
czekanowski . . . . .	9
dice_dist . . . . .	10
dist.diversity . . . . .	11
distance . . . . .	12
divergence_sq . . . . .	15
estimate.probability . . . . .	16
euclidean . . . . .	17
fidelity . . . . .	18
getDistMethods . . . . .	18
gJSD . . . . .	19
gower . . . . .	20
H . . . . .	21
harmonic_mean_dist . . . . .	22
hellinger . . . . .	22
inner_product . . . . .	23
intersection_dist . . . . .	23
jaccard . . . . .	24
JE . . . . .	25
jeffreys . . . . .	26
jensen_difference . . . . .	26
jensen_shannon . . . . .	27
JSD . . . . .	28
KL . . . . .	29
kulczynski_d . . . . .	31
kullback_leibler_distance . . . . .	32
kumar_hassebrook . . . . .	33
kumar_johnson . . . . .	33
k_divergence . . . . .	34
lin.cor . . . . .	35
lorentzian . . . . .	36
manhattan . . . . .	36
matusita . . . . .	37
MI . . . . .	38
minkowski . . . . .	39
motyka . . . . .	39
neyman_chi_sq . . . . .	40
pearson_chi_sq . . . . .	40

<i>additive_symm_chi_sq</i>	3
prob_symm_chi_sq . . . . .	41
ruzicka . . . . .	42
soergel . . . . .	42
sorensen . . . . .	43
squared_chi_sq . . . . .	43
squared_chord . . . . .	44
squared_euclidean . . . . .	45
taneja . . . . .	45
tanimoto . . . . .	46
topsoe . . . . .	46
wave_hedges . . . . .	47
<b>Index</b>	<b>48</b>

---

*additive\_symm\_chi\_sq* *Additive symmetric chi-squared distance (lowlevel function)*

---

## Description

The lowlevel function for computing the *additive\_symm\_chi\_sq* distance.

## Usage

```
additive_symm_chi_sq(P, Q, testNA)
```

## Arguments

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

## Author(s)

Hajk-Georg Drost

## Examples

```
additive_symm_chi_sq(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

avg	<i>AVG distance (lowlevel function)</i>
-----	---

---

**Description**

The lowlevel function for computing the avg distance.

**Usage**

```
avg(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
avg(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

bhattacharyya	<i>Bhattacharyya distance (lowlevel function)</i>
---------------	---

---

**Description**

The lowlevel function for computing the bhattacharyya distance.

**Usage**

```
bhattacharyya(P, Q, testNA, unit)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.
unit	type of log function. Option are <ul style="list-style-type: none"> <li>• unit = "log"</li> <li>• unit = "log2"</li> <li>• unit = "log10"</li> </ul>

**Author(s)**

Hajk-Georg Drost

**Examples**

```
bhattacharyya(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE, unit = "log2")
```

---

binned.kernel.est      *Kernel Density Estimation*


---

**Description**

This function implements an interface to the kernel density estimation functions provided by the **KernSmooth** package.

**Usage**

```
binned.kernel.est(data, kernel = "normal", bandwidth = NULL,
  canonical = FALSE, scalest = "minim", level = 2L,
  gridsize = 401L, range.data = range(data), truncate = TRUE)
```

**Arguments**

data	a numeric vector containing the sample on which the kernel density estimate is to be constructed.
kernel	character string specifying the smoothing kernel
bandwidth	the kernel bandwidth smoothing parameter.
canonical	a logical value indicating whether canonically scaled kernels should be used
scalest	estimate of scale. <ul style="list-style-type: none"> <li>• "stdev" - standard deviation is used.</li> <li>• "iqr" - inter-quartile range divided by 1.349 is used.</li> <li>• "minim" - minimum of "stdev" and "iqr" is used.</li> </ul>
level	number of levels of functional estimation used in the plug-in rule.
gridsize	the number of equally-spaced points over which binning is performed to obtain kernel functional approximation.
range.data	vector containing the minimum and maximum values of data at which to compute the estimate. The default is the minimum and maximum data values.
truncate	logical value indicating whether data with x values outside the range specified by range.data should be ignored.

**Author(s)**

Hajk-Georg Drost

**References**

Matt Wand (2015). KernSmooth: Functions for Kernel Smoothing Supporting Wand & Jones (1995). R package version 2.23-14.

Henry Deng and Hadley Wickham (2011). Density estimation in R. <http://vita.had.co.nz/papers/density-estimation.pdf>.

---

canberra	<i>Canberra distance (lowlevel function)</i>
----------	--

---

**Description**

The lowlevel function for computing the canberra distance.

**Usage**

```
canberra(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
canberra(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

CE	<i>Shannon's Conditional-Entropy <math>H(X Y)</math></i>
----	--

---

**Description**

Compute Shannon's Conditional-Entropy based on the chain rule  $H(X|Y) = H(X, Y) - H(Y)$  based on a given joint-probability vector  $P(X, Y)$  and probability vector  $P(Y)$ .

**Usage**

```
CE(xy, y, unit = "log2")
```

**Arguments**

xy	a numeric joint-probability vector $P(X, Y)$ for which Shannon's Joint-Entropy $H(X, Y)$ shall be computed.
y	a numeric probability vector $P(Y)$ for which Shannon's Entropy $H(Y)$ (as part of the chain rule) shall be computed. It is important to note that this probability vector must be the probability distribution of random variable Y ( $P(Y)$ for which $H(Y)$ is computed).
unit	a character string specifying the logarithm unit that shall be used to compute distances that depend on log computations.

**Details**

This function might be useful to fastly compute Shannon's Conditional-Entropy for any given joint-probability vector and probability vector.

**Value**

Shannon's Conditional-Entropy in bit.

**Note**

Note that the probability vector  $P(Y)$  must be the probability distribution of random variable Y (  $P(Y)$  for which  $H(Y)$  is computed ) and furthermore used for the chain rule computation of  $H(X|Y) = H(X, Y) - H(Y)$ .

**Author(s)**

Hajk-Georg Drost

**References**

Shannon, Claude E. 1948. "A Mathematical Theory of Communication". *Bell System Technical Journal* **27** (3): 379-423.

**See Also**

[H, JE](#)

**Examples**

```
CE(1:10/sum(1:10),1:10/sum(1:10))
```

---

chebyshev	<i>Chebyshev distance (lowlevel function)</i>
-----------	---

---

**Description**

The lowlevel function for computing the chebyshev distance.

**Usage**

```
chebyshev(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
chebyshev(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

clark_sq	<i>Clark squared distance (lowlevel function)</i>
----------	---

---

**Description**

The lowlevel function for computing the clark\_sq distance.

**Usage**

```
clark_sq(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.



**Author(s)**

Hajk-Georg Drost

**Examples**

```
clark_sq(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

cosine_dist	<i>Cosine distance (lowlevel function)</i>
-------------	--

---

**Description**

The lowlevel function for computing the cosine\_dist distance.

**Usage**

```
cosine_dist(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
cosine_dist(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

czekanowski	<i>Czekanowski distance (lowlevel function)</i>
-------------	---

---

**Description**

The lowlevel function for computing the czekanowski distance.

**Usage**

```
czekanowski(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
czekanowski(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

dice_dist	<i>Dice distance (lowlevel function)</i>
-----------	--

---

**Description**

The lowlevel function for computing the dice\_dist distance.

**Usage**

```
dice_dist(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
dice_dist(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

dist.diversity	<i>Distance Diversity between Probability Density Functions</i>
----------------	---

---

## Description

This function computes all distance values between two probability density functions that are available in [getDistMethods](#) and returns a vector storing the corresponding distance measures. This vector is *named distance diversity vector*.

## Usage

```
dist.diversity(x, p, test.na = FALSE, unit = "log2")
```

## Arguments

- |         |   |
|---------|---|
| x       | a numeric data.frame or matrix (storing probability vectors) or a numeric data.frame or matrix storing counts (if est.prob is specified).   |
| p       | power of the Minkowski distance.  |
| test.na | a boolean value indicating whether input vectors should be tested for NA values. Faster computations if test.na = FALSE.  |
| unit    | a character string specifying the logarithm unit that should be used to compute distances that depend on log computations. Options are: <ul style="list-style-type: none"><li>• unit = "log"</li><li>• unit = "log2"</li><li>• unit = "log10"</li></ul> |

## Author(s)

Hajk-Georg Drost

## Examples

```
dist.diversity(rbind(1:10/sum(1:10), 20:29/sum(20:29)), p = 2, unit = "log2")
```

---

distance *Distances between Probability Density Functions*

---

### Description

This functions computes the distance/dissimilarity between two probability density functions.

### Usage

```
distance(x, method = "euclidean", p = NULL, test.na = TRUE,
         unit = "log", est.prob = NULL)
```

### Arguments

x	a numeric <code>data.frame</code> or <code>matrix</code> (storing probability vectors) or a numeric <code>data.frame</code> or <code>matrix</code> storing counts (if <code>est.prob</code> is specified).
method	a character string indicating whether the distance measure that should be computed.
p	power of the Minkowski distance.
test.na	a boolean value indicating whether input vectors should be tested for NA values. Faster computations if <code>test.na = FALSE</code> .
unit	a character string specifying the logarithm unit that should be used to compute distances that depend on log computations.
est.prob	method to estimate probabilities from input count vectors such as non-probability vectors. Default: <code>est.prob = NULL</code> . Options are: <ul style="list-style-type: none"> <li>• <code>est.prob = "empirical"</code>: The relative frequencies of each vector are computed internally. For example an input matrix <code>rbind(1:10, 11:20)</code> will be transformed to a probability vector <code>rbind(1:10 / sum(1:10), 11:20 / sum(11:20))</code></li> </ul>

### Details

Here a distance is defined as a quantitative degree of how far two mathematical objects are apart from eachother (Cha, 2007).

This function implements the following distance/similarity measures to quantify the distance between probability density functions:

- $L_p$  Minkowski family
  - Euclidean :  $d = \sqrt{\sum |P_i - Q_i|^2}$
  - Manhattan :  $d = \sum |P_i - Q_i|$
  - Minkowski :  $d = (\sum |P_i - Q_i|^p)^{1/p}$
  - Chebyshev :  $d = \max |P_i - Q_i|$
- $L_1$  family
  - Sorensen :  $d = \sum |P_i - Q_i| / \sum (P_i + Q_i)$
  - Gower :  $d = 1/d * \sum |P_i - Q_i|$

- Soergel :  $d = \sum |P_i - Q_i| / \sum \max(P_i, Q_i)$
- Kulczynski d :  $d = \sum |P_i - Q_i| / \sum \min(P_i, Q_i)$
- Canberra :  $d = \sum |P_i - Q_i| / (P_i + Q_i)$
- Lorentzian :  $d = \sum \ln(1 + |P_i - Q_i|)$
- Intersection family
  - Intersection :  $s = \sum \min(P_i, Q_i)$
  - Non-Intersection :  $d = 1 - \sum \min(P_i, Q_i)$
  - Wave Hedges :  $d = \sum |P_i - Q_i| / \max(P_i, Q_i)$
  - Czekanowski :  $d = \sum |P_i - Q_i| / \sum |P_i + Q_i|$
  - Motyka :  $d = \sum \min(P_i, Q_i) / (P_i + Q_i)$
  - Kulczynski s :  $d = 1 / \sum |P_i - Q_i| / \sum \min(P_i, Q_i)$
  - Tanimoto :  $d = \sum (\max(P_i, Q_i) - \min(P_i, Q_i)) / \sum \max(P_i, Q_i)$  ; equivalent to Soergel
  - Ruzicka :  $s = \sum \min(P_i, Q_i) / \sum \max(P_i, Q_i)$  ; equivalent to  $1 - \text{Tanimoto} = 1 - \text{Soergel}$
- Inner Product family
  - Inner Product :  $s = \sum P_i * Q_i$
  - Harmonic mean :  $s = 2 * \sum (P_i * Q_i) / (P_i + Q_i)$
  - Cosine :  $s = \sum (P_i * Q_i) / \sqrt{\sum P_i^2} * \sqrt{\sum Q_i^2}$
  - Kumar-Hassebrook (PCE) :  $s = \sum (P_i * Q_i) / (\sum P_i^2 + \sum Q_i^2 - \sum (P_i * Q_i))$
  - Jaccard :  $d = 1 - \sum (P_i * Q_i) / (\sum P_i^2 + \sum Q_i^2 - \sum (P_i * Q_i))$  ; equivalent to  $1 - \text{Kumar-Hassebrook}$
  - Dice :  $d = \sum (P_i - Q_i)^2 / (\sum P_i^2 + \sum Q_i^2)$
- Squared-chord family
  - Fidelity :  $s = \sum \sqrt{P_i * Q_i}$
  - Bhattacharyya :  $d = -\ln \sum \sqrt{P_i * Q_i}$
  - Hellinger :  $d = 2 * \sqrt{1 - \sum \sqrt{P_i * Q_i}}$
  - Matusita :  $d = \sqrt{2 - 2 * \sum \sqrt{P_i * Q_i}}$
  - Squared-chord :  $d = \sum (\sqrt{P_i} - \sqrt{Q_i})^2$
- Squared L<sub>2</sub> family ( $X^2$  squared family)
  - Squared Euclidean :  $d = \sum (P_i - Q_i)^2$
  - Pearson  $X^2$  :  $d = \sum ((P_i - Q_i)^2 / Q_i)$
  - Neyman  $X^2$  :  $d = \sum ((P_i - Q_i)^2 / P_i)$
  - Squared  $X^2$  :  $d = \sum ((P_i - Q_i)^2 / (P_i + Q_i))$
  - Probabilistic Symmetric  $X^2$  :  $d = 2 * \sum ((P_i - Q_i)^2 / (P_i + Q_i))$
  - Divergence :  $X^2$  :  $d = 2 * \sum ((P_i - Q_i)^2 / (P_i + Q_i)^2)$
  - Clark :  $d = \sqrt{\sum (|P_i - Q_i| / (P_i + Q_i))^2}$
  - Additive Symmetric  $X^2$  :  $d = \sum (((P_i - Q_i)^2 * (P_i + Q_i)) / (P_i * Q_i))$
- Shannon's entropy family
  - Kullback-Leibler :  $d = \sum P_i * \log(P_i / Q_i)$
  - Jeffreys :  $d = \sum (P_i - Q_i) * \log(P_i / Q_i)$

- K divergence :  $d = \sum P_i * \log(2 * P_i / (P_i + Q_i))$
- Topsoe :  $d = \sum (P_i * \log(2 * P_i / (P_i + Q_i)) + (Q_i * \log(2 * Q_i / (P_i + Q_i)))$
- Jensen-Shannon :  $d = 0.5 * (\sum P_i * \log(2 * P_i / (P_i + Q_i)) + \sum Q_i * \log(2 * Q_i / (P_i + Q_i)))$
- Jensen difference :  $d = \sum ((P_i * \log(P_i) + Q_i * \log(Q_i)) / 2) - (P_i + Q_i) / 2 * \log((P_i + Q_i) / 2)$

- Combinations

- Taneja :  $d = \sum (P_i + Q_i / 2) * \log((P_i + Q_i) / (2 * \sqrt{P_i * Q_i}))$
- Kumar-Johnson :  $d = \sum (P_i^2 - Q_i^2)^2 / 2 * (P_i * Q_i)^{1.5}$
- Avg(L\_1, L\_n) :  $d = \sum |P_i - Q_i| + \max |P_i - Q_i| / 2$

In cases where `x` specifies a count matrix, the argument `est.prob` can be selected to first estimate probability vectors from input count vectors and second compute the corresponding distance measure based on the estimated probability vectors.

The following probability estimation methods are implemented in this function:

- `est.prob = "empirical"` : relative frequencies of counts.

### Value

The following results are returned depending on the dimension of `x`:

- in case `nrow(x) = 2` : a single distance value.
- in case `nrow(x) > 2` : a distance matrix storing distance values for all pairwise probability vector comparisons.

### Note

According to the reference in some distance measure computations invalid computations can occur when dealing with 0 probabilities.

In these cases the convention is treated as follows:

- division by zero - case  $\theta/\theta$ : when the divisor and dividend become zero,  $\theta/\theta$  is treated as  $\theta$ .
- division by zero - case  $n/\theta$ : when only the divisor becomes  $\theta$ , the corresponding  $\theta$  is replaced by a small  $\epsilon = 0.00001$ .
- log of zero - case  $\theta * \log(\theta)$ : is treated as  $\theta$ .
- log of zero - case  $\log(\theta)$ : zero is replaced by a small  $\epsilon = 0.00001$ .

### Author(s)

Hajk-Georg Drost

### References

Sung-Hyuk Cha. (2007). *Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions*. International Journal of Mathematical Models and Methods in Applied Sciences 4: 1.

### See Also

[getDistMethods](#), [estimate.probability](#), [dist.diversity](#)

**Examples**

```

# Simple Examples

# receive a list of implemented probability distance measures
getDistMethods()

## compute the euclidean distance between two probability vectors
distance(rbind(1:10/sum(1:10), 20:29/sum(20:29)), method = "euclidean")

## compute the euclidean distance between all pairwise comparisons of probability vectors
ProbMatrix <- rbind(1:10/sum(1:10), 20:29/sum(20:29), 30:39/sum(30:39))
distance(ProbMatrix, method = "euclidean")

# compute distance matrix without testing for NA values in the input matrix
distance(ProbMatrix, method = "euclidean", test.na = FALSE)

# Specialized Examples

CountMatrix <- rbind(1:10, 20:29, 30:39)

## estimate probabilities from a count matrix
distance(CountMatrix, method = "euclidean", est.prob = "empirical")

## compute the euclidean distance for count data
## NOTE: some distance measures are only defined for probability values,
distance(CountMatrix, method = "euclidean")

## compute the Kullback-Leibler Divergence with different logarithm bases:
### case: unit = log (Default)
distance(ProbMatrix, method = "kullback-leibler", unit = "log")

### case: unit = log2
distance(ProbMatrix, method = "kullback-leibler", unit = "log2")

### case: unit = log10
distance(ProbMatrix, method = "kullback-leibler", unit = "log10")

```

---

divergence\_sq

*Divergence squared distance (lowlevel function)*


---

**Description**

The lowlevel function for computing the divergence\_sq distance.

**Usage**

```
divergence_sq(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
divergence_sq(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

estimate.probability *Estimate Probability Vectors From Count Vectors*

---

**Description**

This function takes a numeric count vector and returns estimated probabilities of the corresponding counts.

The following probability estimation methods are implemented in this function:

- method = "empirical" : generates the relative frequency of the data  $x/\text{sum}(x)$ .
- 
- 

**Usage**

```
estimate.probability(x, method = "empirical")
```

**Arguments**

x	a numeric vector storing count values.
method	a character string specifying the estimation method tht should be used to estimate probabilities from input counts.

**Value**

a numeric probability vector.

**Author(s)**

Hajk-Georg Drost



**Examples**

```
# generate a count vector
x <- runif(100)

# generate a probability vector from corresponding counts
# method = "empirical"
x.prob <- estimate.probability(x, method = "empirical")
```

---

euclidean	<i>Euclidean distance (lowlevel function)</i>
-----------	---

---

**Description**

The lowlevel function for computing the euclidean distance.

**Usage**

```
euclidean(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
euclidean(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

fidelity	<i>Fidelity distance (lowlevel function)</i>
----------	--

---

**Description**

The lowlevel function for computing the fidelity distance.

**Usage**

```
fidelity(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
fidelity(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

getDistMethods	<i>Get method names for distance</i>
----------------	--------------------------------------

---

**Description**

This function returns the names of the methods that can be applied to compute distances between probability density functions using the [distance](#) function.

**Usage**

```
getDistMethods()
```

**Author(s)**

Hajk-Georg Drost

**Examples**

```
getDistMethods()
```

gJSD

*Generalized Jensen-Shannon Divergence***Description**

This function computes the Generalized Jensen-Shannon Divergence of a probability matrix.

**Usage**

```
gJSD(x, unit = "log2", weights = NULL, est.prob = NULL)
```

**Arguments**

x	a probability matrix.
unit	a character string specifying the logarithm unit that shall be used to compute distances that depend on log computations.
weights	a numeric vector specifying the weights for each distribution in x. Default: weights = NULL; in this case all distributions are weighted equally (= uniform distribution of weights). In case users wish to specify non-uniform weights for e.g. 3 distributions, they can specify the argument weights = c(0.5, 0.25, 0.25). This notation denotes that vec1 is weighted by 0.5, vec2 is weighted by 0.25, and vec3 is weighted by 0.25 as well.
est.prob	method to estimate probabilities from input count vectors such as non-probability vectors. Default: est.prob = NULL. Options are: <ul style="list-style-type: none"> <li>est.prob = "empirical": The relative frequencies of each vector are computed internally. For example an input matrix rbind(1:10, 11:20) will be transformed to a probability vector rbind(1:10 / sum(1:10), 11:20 / sum(11:20))</li> </ul>

**Details**

Function to compute the Generalized Jensen-Shannon Divergence

$$JSD_{\pi_1, \dots, \pi_n}(P_1, \dots, P_n) = H(\sum_{i=1}^n \pi_i * P_i) - \sum_{i=1}^n \pi_i * H(P_i)$$

where  $\pi_1, \dots, \pi_n$  denote the weights selected for the probability vectors  $P_1, \dots, P_n$  and  $H(P_i)$  denotes the Shannon Entropy of probability vector  $P_i$ .

**Value**

The Jensen-Shannon divergence between all possible combinations of comparisons.

**Author(s)**

Hajk-Georg Drost

**See Also**

[KL](#), [H](#), [JSD](#), [CE](#), [JE](#)

**Examples**

```
# define input probability matrix
Prob <- rbind(1:10/sum(1:10), 20:29/sum(20:29), 30:39/sum(30:39))

# compute the Generalized JSD comparing the PS probability matrix
gJSD(Prob)

# Generalized Jensen-Shannon Divergence between three vectors using different log bases
gJSD(Prob, unit = "log2") # Default
gJSD(Prob, unit = "log")
gJSD(Prob, unit = "log10")

# Jensen-Shannon Divergence Divergence between count vectors P.count and Q.count
P.count <- 1:10
Q.count <- 20:29
R.count <- 30:39
x.count <- rbind(P.count, Q.count, R.count)
gJSD(x.count, est.prob = "empirical")
```

---

gower

*Gower distance (lowlevel function)*

---

**Description**

The lowlevel function for computing the gower distance.

**Usage**

```
gower(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
gower(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

H	<i>Shannon's Entropy</i> $H(X)$
---	---------------------------------

---

**Description**

Compute the Shannon's Entropy  $H(X) = - \sum P(X) * \log_2(P(X))$  based on a given probability vector  $P(X)$ .

**Usage**

```
H(x, unit = "log2")
```

**Arguments**

x	a numeric probability vector $P(X)$ for which Shannon's Entropy $H(X)$ shall be computed.
unit	a character string specifying the logarithm unit that shall be used to compute distances that depend on log computations.

**Details**

This function might be useful to fastly compute Shannon's Entropy for any given probability vector.

**Value**

a numeric value representing Shannon's Entropy in bit.

**Author(s)**

Hajk-Georg Drost

**References**

Shannon, Claude E. 1948. "A Mathematical Theory of Communication". *Bell System Technical Journal* **27** (3): 379-423.

**See Also**

[JE](#), [CE](#), [KL](#), [JSD](#), [gJSD](#)

**Examples**

```
H(1:10/sum(1:10))
```

---

harmonic\_mean\_dist      *Harmonic mean distance (lowlevel function)*

---

**Description**

The lowlevel function for computing the harmonic\_mean\_dist distance.

**Usage**

```
harmonic_mean_dist(P, Q, testNA)
```

**Arguments**

P                    a numeric vector storing the first distribution.  
Q                    a numeric vector storing the second distribution.  
testNA              a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
harmonic_mean_dist(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

hellinger              *Hellinger distance (lowlevel function)*

---

**Description**

The lowlevel function for computing the hellinger distance.

**Usage**

```
hellinger(P, Q, testNA)
```

**Arguments**

P                    a numeric vector storing the first distribution.  
Q                    a numeric vector storing the second distribution.  
testNA              a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
hellinger(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

inner_product	<i>Inner product distance (lowlevel function)</i>
---------------	---

---

**Description**

The lowlevel function for computing the inner\_product distance.

**Usage**

```
inner_product(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
inner_product(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

intersection_dist	<i>Intersection distance (lowlevel function)</i>
-------------------	--

---

**Description**

The lowlevel function for computing the intersection\_dist distance.

**Usage**

```
intersection_dist(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
intersection_dist(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

jaccard	<i>Jaccard distance (lowlevel function)</i>
---------	---

---

**Description**

The lowlevel function for computing the jaccard distance.

**Usage**

```
jaccard(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
jaccard(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```



---

**JE***Shannon's Joint-Entropy*  $H(X, Y)$ 

---

**Description**

This function computes Shannon's Joint-Entropy  $H(X, Y) = -\sum \sum P(X, Y) * \log_2(P(X, Y))$  based on a given joint-probability vector  $P(X, Y)$ .

**Usage**

```
JE(x, unit = "log2")
```

**Arguments**

<code>x</code>	a numeric joint-probability vector $P(X, Y)$ for which Shannon's Joint-Entropy $H(X, Y)$ shall be computed.
<code>unit</code>	a character string specifying the logarithm unit that shall be used to compute distances that depend on log computations.

**Value**

a numeric value representing Shannon's Joint-Entropy in bit.

**Author(s)**

Hajk-Georg Drost

**References**

Shannon, Claude E. 1948. "A Mathematical Theory of Communication". *Bell System Technical Journal* **27** (3): 379-423.

**See Also**

[H](#), [CE](#), [KL](#), [JSD](#), [gJSD](#), [distance](#)

**Examples**

```
JE(1:100/sum(1:100))
```

---

jeffreys                      *Jeffreys distance (lowlevel function)*

---

**Description**

The lowlevel function for computing the jeffreys distance.

**Usage**

```
jeffreys(P, Q, testNA, unit)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.
unit	type of log function. Option are <ul style="list-style-type: none"><li>• unit = "log"</li><li>• unit = "log2"</li><li>• unit = "log10"</li></ul>

**Author(s)**

Hajk-Georg Drost

**Examples**

```
jeffreys(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE, unit = "log2")
```

---

jensen\_difference                      *Jensen difference (lowlevel function)*

---

**Description**

The lowlevel function for computing the jensen\_difference distance.

**Usage**

```
jensen_difference(P, Q, testNA, unit)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.
unit	type of log function. Option are <ul style="list-style-type: none"> <li>• unit = "log"</li> <li>• unit = "log2"</li> <li>• unit = "log10"</li> </ul>

**Author(s)**

Hajk-Georg Drost

**Examples**

```
jensen_difference(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE, unit = "log2")
```

---

jensen_shannon	<i>Jensen-Shannon distance (lowlevel function)</i>
----------------	--

---

**Description**

The lowlevel function for computing the jensen\_shannon distance.

**Usage**

```
jensen_shannon(P, Q, testNA, unit)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.
unit	type of log function. Option are <ul style="list-style-type: none"> <li>• unit = "log"</li> <li>• unit = "log2"</li> <li>• unit = "log10"</li> </ul>

**Author(s)**

Hajk-Georg Drost

**Examples**

```
jensen_shannon(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE, unit = "log2")
```

JSD

*Jensen-Shannon Divergence***Description**

This function computes a distance matrix or distance value based on the Jensen-Shannon Divergence with equal weights.

**Usage**

```
JSD(x, test.na = TRUE, unit = "log2", est.prob = NULL)
```

**Arguments**

<code>x</code>	a numeric <code>data.frame</code> or <code>matrix</code> (storing probability vectors) or a numeric <code>data.frame</code> or <code>matrix</code> storing counts (if <code>est.prob = TRUE</code> ). See <a href="#">distance</a> for details.
<code>test.na</code>	a boolean value specifying whether input vectors shall be tested for NA values.
<code>unit</code>	a character string specifying the logarithm unit that shall be used to compute distances that depend on log computations.
<code>est.prob</code>	method to estimate probabilities from input count vectors such as non-probability vectors. Default: <code>est.prob = NULL</code> . Options are: <ul style="list-style-type: none"> <li>• <code>est.prob = "empirical"</code>: The relative frequencies of each vector are computed internally. For example an input matrix <code>rbind(1:10, 11:20)</code> will be transformed to a probability vector <code>rbind(1:10 / sum(1:10), 11:20 / sum(11:20))</code></li> </ul>

**Details**

Function to compute the Jensen-Shannon Divergence  $JSD(P \parallel Q)$  between two probability distributions  $P$  and  $Q$  with equal weights  $\pi_1 = \pi_2 = 1/2$ .

The Jensen-Shannon Divergence  $JSD(P \parallel Q)$  between two probability distributions  $P$  and  $Q$  is defined as:

$$JSD(P \parallel Q) = 0.5 * (KL(P \parallel R) + KL(Q \parallel R))$$

where  $R = 0.5 * (P + Q)$  denotes the mid-point of the probability vectors  $P$  and  $Q$ , and  $KL(P \parallel R)$ ,  $KL(Q \parallel R)$  denote the Kullback-Leibler Divergence of  $P$  and  $R$ , as well as  $Q$  and  $R$ .

**General properties of the Jensen-Shannon Divergence:**

- 1) JSD is non-negative.
- 2) JSD is a symmetric measure  $JSD(P \parallel Q) = JSD(Q \parallel P)$ .
- 3)  $JSD = 0$ , if and only if  $P = Q$ .

**Value**

a distance value or matrix based on JSD computations.

**Author(s)**

Hajk-Georg Drost

**References**

Lin J. 1991. "Divergence Measures Based on the Shannon Entropy". IEEE Transactions on Information Theory. (33) 1: 145-151.

Endres M. and Schindelin J. E. 2003. "A new metric for probability distributions". IEEE Trans. on Info. Thy. (49) 3: 1858-1860.

**See Also**

[KL](#), [H](#), [CE](#), [gJSD](#), [distance](#)

**Examples**

```
# Jensen-Shannon Divergence between P and Q
P <- 1:10/sum(1:10)
Q <- 20:29/sum(20:29)
x <- rbind(P,Q)
JSD(x)

# Jensen-Shannon Divergence between P and Q using different log bases
JSD(x, unit = "log2") # Default
JSD(x, unit = "log")
JSD(x, unit = "log10")

# Jensen-Shannon Divergence Divergence between count vectors P.count and Q.count
P.count <- 1:10
Q.count <- 20:29
x.count <- rbind(P.count,Q.count)
JSD(x.count, est.prob = "empirical")

# Example: Distance Matrix using JSD-Distance

Prob <- rbind(1:10/sum(1:10), 20:29/sum(20:29), 30:39/sum(30:39))

# compute the KL matrix of a given probability matrix
JSDMatrix <- JSD(Prob)

# plot a heatmap of the corresponding JSD matrix
heatmap(JSDMatrix)
```

**Description**

This function computes the Kullback-Leibler divergence of two probability distributions P and Q.

**Usage**

```
KL(x, test.na = TRUE, unit = "log2", est.prob = NULL)
```

**Arguments**

x	a numeric <code>data.frame</code> or <code>matrix</code> (storing probability vectors) or a numeric <code>data.frame</code> or <code>matrix</code> storing counts (if <code>est.prob = TRUE</code> ). See <a href="#">distance</a> for details.
test.na	a boolean value indicating whether input vectors should be tested for NA values.
unit	a character string specifying the logarithm unit that shall be used to compute distances that depend on log computations.
est.prob	method to estimate probabilities from a count vector. Default: <code>est.prob = NULL</code> .

**Details**

$$KL(P||Q) = \sum P(P) * \log_2(P(P)/P(Q)) = H(P, Q) - H(P)$$

where  $H(P,Q)$  denotes the joint entropy of the probability distributions P and Q and  $H(P)$  denotes the entropy of probability distribution P. In case  $P = Q$  then  $KL(P,Q) = 0$  and in case  $P \neq Q$  then  $KL(P,Q) > 0$ .

The KL divergence is a non-symmetric measure of the directed divergence between two probability distributions P and Q. It only fulfills the *positivity* property of a *distance metric*.

Because of the relation  $KL(P||Q) = H(P,Q) - H(P)$ , the Kullback-Leibler divergence of two probability distributions P and Q is also named *Cross Entropy* of two probability distributions P and Q.

**Value**

The Kullback-Leibler divergence of probability vectors.

**Author(s)**

Hajk-Georg Drost

**References**

Cover Thomas M. and Thomas Joy A. 2006. Elements of Information Theory. *John Wiley & Sons*.

**See Also**

[H](#), [CE](#), [JSD](#), [gJSD](#), [distance](#)

## Examples

```
# Kulback-Leibler Divergence between P and Q
P <- 1:10/sum(1:10)
Q <- 20:29/sum(20:29)
x <- rbind(P,Q)
KL(x)

# Kulback-Leibler Divergence between P and Q using different log bases
KL(x, unit = "log2") # Default
KL(x, unit = "log")
KL(x, unit = "log10")

# Kulback-Leibler Divergence between count vectors P.count and Q.count
P.count <- 1:10
Q.count <- 20:29
x.count <- rbind(P.count,Q.count)
KL(x, est.prob = "empirical")

# Example: Distance Matrix using KL-Distance

Prob <- rbind(1:10/sum(1:10), 20:29/sum(20:29), 30:39/sum(30:39))

# compute the KL matrix of a given probability matrix
KLMatrix <- KL(Prob)

# plot a heatmap of the corresponding KL matrix
heatmap(KLMatrix)
```

---

kulczynski\_d

*Kulczynski\_d distance (lowlevel function)*

---

## Description

The lowlevel function for computing the kulczynski\_d distance.

## Usage

```
kulczynski_d(P, Q, testNA)
```

## Arguments

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
kulczynski_d(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

kullback\_leibler\_distance

*kullback-Leibler distance (lowlevel function)*

---

**Description**

The lowlevel function for computing the kullback\_leibler\_distance distance.

**Usage**

```
kullback_leibler_distance(P, Q, testNA, unit)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.
unit	type of log function. Option are <ul style="list-style-type: none"><li>• unit = "log"</li><li>• unit = "log2"</li><li>• unit = "log10"</li></ul>

**Author(s)**

Hajk-Georg Drost

**Examples**

```
kullback_leibler_distance(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE, unit = "log2")
```



---

kumar\_hassebrook      *Kumar hassebrook distance (lowlevel function)*

---

**Description**

The lowlevel function for computing the kumar\_hassebrook distance.

**Usage**

```
kumar_hassebrook(P, Q, testNA)
```

**Arguments**

P                    a numeric vector storing the first distribution.  
Q                    a numeric vector storing the second distribution.  
testNA              a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
kumar_hassebrook(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

kumar\_johnson      *Kumar-Johnson distance (lowlevel function)*

---

**Description**

The lowlevel function for computing the kumar\_johnson distance.

**Usage**

```
kumar_johnson(P, Q, testNA)
```

**Arguments**

P                    a numeric vector storing the first distribution.  
Q                    a numeric vector storing the second distribution.  
testNA              a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
kumar_johnson(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

k_divergence	<i>K-Divergence (lowlevel function)</i>
--------------	---

---

**Description**

The lowlevel function for computing the k\_divergence distance.

**Usage**

```
k_divergence(P, Q, testNA, unit)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.
unit	type of log function. Option are <ul style="list-style-type: none"><li>• unit = "log"</li><li>• unit = "log2"</li><li>• unit = "log10"</li></ul>

**Author(s)**

Hajk-Georg Drost

**Examples**

```
k_divergence(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE, unit = "log2")
```

---

`lin.cor`*Linear Correlation*

---

### Description

This function computed the linear correlation between two vectors or a correlation matrix for an input matrix.

The following methods to compute linear correlations are implemented in this function:

### Usage

```
lin.cor(x, y = NULL, method = "pearson", test.na = FALSE)
```

### Arguments

<code>x</code>	a numeric vector, matrix, or data.frame.
<code>y</code>	a numeric vector that should be correlated with <code>x</code> .
<code>method</code>	the method to compute the linear correlation between <code>x</code> and <code>y</code> .
<code>test.na</code>	a boolean value indicating whether input data should be checked for NA values.

### Details

- `method = "pearson"` : Pearson's correlation coefficient (centred).
- `method = "pearson2"` : Pearson's uncentred correlation coefficient.
- `method = "sq_pearson"` . Squared Pearson's correlation coefficient.
- `method = "kendall"` : Kendall's correlation coefficient.
- `method = "spearman"` : Spearman's correlation coefficient.

Further Details:

- *Pearson's correlation coefficient (centred)* :

### Author(s)

Hajk-Georg Drost

---

lorentzian	<i>Lorentzian distance (lowlevel function)</i>
------------	--

---

**Description**

The lowlevel function for computing the lorentzian distance.

**Usage**

```
lorentzian(P, Q, testNA, unit)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.
unit	type of log function. Option are <ul style="list-style-type: none"><li>• unit = "log"</li><li>• unit = "log2"</li><li>• unit = "log10"</li></ul>

**Author(s)**

Hajk-Georg Drost

**Examples**

```
lorentzian(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE, unit = "log2")
```

---

manhattan	<i>Manhattan distance (lowlevel function)</i>
-----------	---

---

**Description**

The lowlevel function for computing the manhattan distance.

**Usage**

```
manhattan(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
manhattan(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

matusita	<i>Matusita distance (lowlevel function)</i>
----------	--

---

**Description**

The lowlevel function for computing the matusita distance.

**Usage**

```
matusita(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
matusita(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

MI

*Shannon's Mutual Information  $I(X, Y)$*

---

### Description

Compute Shannon's Mutual Information based on the identity  $I(X, Y) = H(X) + H(Y) - H(X, Y)$  based on a given joint-probability vector  $P(X, Y)$  and probability vectors  $P(X)$  and  $P(Y)$ .

### Usage

```
MI(x, y, xy, unit = "log2")
```

### Arguments

x	a numeric probability vector $P(X)$ .
y	a numeric probability vector $P(Y)$ .
xy	a numeric joint-probability vector $P(X, Y)$ .
unit	a character string specifying the logarithm unit that shall be used to compute distances that depend on log computations.

### Details

This function might be useful to fastly compute Shannon's Mutual Information for any given joint-probability vector and probability vectors.

### Value

Shannon's Mutual Information in bit.

### Author(s)

Hajk-Georg Drost

### References

Shannon, Claude E. 1948. "A Mathematical Theory of Communication". *Bell System Technical Journal* **27** (3): 379-423.

### See Also

[H](#), [JE](#), [CE](#)

### Examples

```
MI( x = 1:10/sum(1:10), y = 20:29/sum(20:29), xy = 1:10/sum(1:10) )
```

---

minkowski	<i>Minkowski distance (lowlevel function)</i>
-----------	---

---

**Description**

The lowlevel function for computing the minkowski distance.

**Usage**

```
minkowski(P, Q, n, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
n	index for the minkowski exponent.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
minkowski(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), n = 2, testNA = FALSE)
```

---

motyka	<i>Motyka distance (lowlevel function)</i>
--------	--

---

**Description**

The lowlevel function for computing the motyka distance.

**Usage**

```
motyka(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
motyka(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

neyman_chi_sq	<i>Neyman chi-squared distance (lowlevel function)</i>
---------------	--

---

**Description**

The lowlevel function for computing the neyman\_chi\_sq distance.

**Usage**

```
neyman_chi_sq(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
neyman_chi_sq(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

pearson_chi_sq	<i>Pearson chi-squared distance (lowlevel function)</i>
----------------	---

---

**Description**

The lowlevel function for computing the pearson\_chi\_sq distance.

**Usage**

```
pearson_chi_sq(P, Q, testNA)
```



**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
pearson_chi_sq(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

prob_symm_chi_sq	<i>Probability symmetric chi-squared distance (lowlevel function)</i>
------------------	---

---

**Description**

The lowlevel function for computing the prob\_symm\_chi\_sq distance.

**Usage**

```
prob_symm_chi_sq(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
prob_symm_chi_sq(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

ruzicka	<i>Ruzicka distance (lowlevel function)</i>
---------	---

---

**Description**

The lowlevel function for computing the ruzicka distance.

**Usage**

```
ruzicka(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
ruzicka(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

soergel	<i>Soergel distance (lowlevel function)</i>
---------	---

---

**Description**

The lowlevel function for computing the soergel distance.

**Usage**

```
soergel(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
soergel(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

sorensen

*Sorensen distance (lowlevel function)***Description**

The lowlevel function for computing the sorensen distance.

**Usage**

```
sorensen(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
sorensen(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

squared\_chi\_sq

*Squared chi-squared distance (lowlevel function)***Description**

The lowlevel function for computing the squared\_chi\_sq distance.

**Usage**

```
squared_chi_sq(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
squared_chi_sq(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

squared_chord	<i>Squared chord distance (lowlevel function)</i>
---------------	---

---

**Description**

The lowlevel function for computing the squared\_chord distance.

**Usage**

```
squared_chord(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
squared_chord(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

squared_euclidean	<i>Squared euclidean distance (lowlevel function)</i>
-------------------	---

---

**Description**

The lowlevel function for computing the squared\_euclidean distance.

**Usage**

```
squared_euclidean(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
squared_euclidean(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

taneja	<i>Taneja difference (lowlevel function)</i>
--------	--

---

**Description**

The lowlevel function for computing the taneja distance.

**Usage**

```
taneja(P, Q, testNA, unit)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.
unit	type of log function. Option are <ul style="list-style-type: none"> <li>• unit = "log"</li> <li>• unit = "log2"</li> <li>• unit = "log10"</li> </ul>

**Author(s)**

Hajk-Georg Drost

**Examples**

```
taneja(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE, unit = "log2")
```

---

tanimoto

*Tanimoto distance (lowlevel function)*

---

**Description**

The lowlevel function for computing the tanimoto distance.

**Usage**

```
tanimoto(P, Q, testNA)
```

**Arguments**

P                    a numeric vector storing the first distribution.  
 Q                    a numeric vector storing the second distribution.  
 testNA              a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
tanimoto(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

---

topsoe

*Topsoe distance (lowlevel function)*

---

**Description**

The lowlevel function for computing the topsoe distance.

**Usage**

```
topsoe(P, Q, testNA, unit)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.
unit	type of log function. Option are <ul style="list-style-type: none"> <li>• unit = "log"</li> <li>• unit = "log2"</li> <li>• unit = "log10"</li> </ul>

**Author(s)**

Hajk-Georg Drost

**Examples**

```
topsoe(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE, unit = "log2")
```

---

wave_hedges	<i>Wave hedges distance (lowlevel function)</i>
-------------	---

---

**Description**

The lowlevel function for computing the wave\_hedges distance.

**Usage**

```
wave_hedges(P, Q, testNA)
```

**Arguments**

P	a numeric vector storing the first distribution.
Q	a numeric vector storing the second distribution.
testNA	a logical value indicating whether or not distributions shall be checked for NA values.

**Author(s)**

Hajk-Georg Drost

**Examples**

```
wave_hedges(P = 1:10/sum(1:10), Q = 20:29/sum(20:29), testNA = FALSE)
```

# Index

additive\_symm\_chi\_sq, 3  
avg, 4

bhattacharyya, 4  
binned.kernel.est, 5

canberra, 6  
CE, 6, 19, 21, 25, 29, 30, 38  
chebyshev, 8  
clark\_sq, 8  
cosine\_dist, 9  
czekanowski, 9

dice\_dist, 10  
dist.diversity, 11, 14  
distance, 12, 18, 25, 28–30  
divergence\_sq, 15

estimate.probability, 14, 16  
euclidean, 17

fidelity, 18

getDistMethods, 11, 14, 18  
gJSD, 19, 21, 25, 29, 30  
gower, 20

H, 7, 19, 21, 25, 29, 30, 38  
harmonic\_mean\_dist, 22  
hellinger, 22

inner\_product, 23  
intersection\_dist, 23

jaccard, 24  
JE, 7, 19, 21, 25, 38  
jeffreys, 26  
jensen\_difference, 26  
jensen\_shannon, 27  
JSD, 19, 21, 25, 28, 30

k\_divergence, 34  
KL, 19, 21, 25, 29, 29  
kulczynski\_d, 31  
kullback\_leibler\_distance, 32  
kumar\_hassebrook, 33  
kumar\_johnson, 33

lin.cor, 35  
lorentzian, 36

manhattan, 36  
matusita, 37  
MI, 38  
minkowski, 39  
motyka, 39

neyman\_chi\_sq, 40

pearson\_chi\_sq, 40  
prob\_symm\_chi\_sq, 41

ruzicka, 42

soergel, 42  
sorensen, 43  
squared\_chi\_sq, 43  
squared\_chord, 44  
squared\_euclidean, 45

taneja, 45  
tanimoto, 46  
topsoe, 46

wave\_hedges, 47