

Package: moocore (via r-universe)

October 28, 2026

Type Package

Title Core Mathematical Functions for Multi-Objective Optimization

Version 0.3.2.900

Description Fast implementations of mathematical operations and performance metrics for multi-objective optimization, including filtering and ranking of dominated vectors according to Pareto optimality, hypervolume metric, C.M. Fonseca, L. Paquete, M. López-Ibáñez (2006) <[doi:10.1109/CEC.2006.1688440](https://doi.org/10.1109/CEC.2006.1688440)>, epsilon indicator, inverted generational distance, computation of the empirical attainment function, V.G. da Fonseca, C.M. Fonseca, A.O. Hall (2001) <[doi:10.1007/3-540-44719-9_15](https://doi.org/10.1007/3-540-44719-9_15)>, and Vorob'ev threshold, expectation and deviation, M. Binois, D. Ginsbourger, O. Roustant (2015) <[doi:10.1016/j.ejor.2014.07.032](https://doi.org/10.1016/j.ejor.2014.07.032)>, among others.

Depends R (>= 4.1)

Imports matrixStats, Rdpack (>= 2.4)

Suggests doctest (>= 0.4.0), spelling, testthat (>= 3.2.0), withr

License LGPL (>= 2.1)

Copyright file COPYRIGHTS

URL <https://multi-objective.github.io/moocore/r/>,
<https://github.com/multi-objective/moocore>

BugReports <https://github.com/multi-objective/moocore/issues>

LazyLoad true

LazyData true

Encoding UTF-8

UseLTO true

Roxygen list(markdown = TRUE, roclets = c("`collate", "`rd",
 "`namespace", "`doctest::dt_roclet"))

SystemRequirements GNU make

RdMacros Rdpack

Config/testthat/edition 3

Language en-GB

Config/Needs/website Hmisc, data.table, dplyr, ggplot2, htmltools,
kableExtra, knitr, plotly, rmarkdown, scales, tidyr,
viridisLite

Config/roxygen2/version 8.0.0

Config/pak/sysreqs make

Repository <https://multi-objective.r-universe.dev>

Date/Publication 2026-10-28 20:41:08 UTC

RemoteUrl <https://github.com/multi-objective/moocore>

RemoteRef HEAD

RemoteSha f383e717f8eeca20931601859025969487118a77

RemoteSubdir r

Contents

as_double_matrix	3
attsurf2df	3
choose_eafdiff	4
compute_eaf_call	5
compute_eafdiff_call	6
CPFs	7
eaf	7
eaf_as_list	10
eafdiff	11
epsilon	12
generate_ndset	14
hv_approx	17
hv_contributions	20
HybridGA	22
hypervolume	23
igd	25
is_nondominated	28
largest_eafdiff	30
normalise	31
pareto_rank	32
r2_exact	34
rbind_datasets	36
read_datasets	37
SPEA2minstoptimeRichmond	38
SPEA2relativeRichmond	39
SPEA2relativeVanzyl	39
tpls50x20_1_MWT	40
transform_maximise	41

<i>as_double_matrix</i>	3
-------------------------	---

vorob_t	41
whv_hype	43
whv_rect	45
write_datasets	47

Index	48
--------------	-----------

<i>as_double_matrix</i>	<i>Convert input to a matrix with "double" storage mode (base::storage.mode()).</i>
-------------------------	---

Description

Convert input to a matrix with "double" storage mode ([base::storage.mode\(\)](#)).

Usage

`as_double_matrix(x)`

Arguments

`x` `data.frame()`|`matrix()`
A numerical data frame or matrix with at least 1 row and 2 columns.

Value

`x` is coerced to a numerical `matrix()`.

<i>attsurf2df</i>	<i>Convert a list of attainment surfaces to a single EAF <code>data.frame</code>.</i>
-------------------	---

Description

Convert a list of attainment surfaces to a single EAF `data.frame`.

Usage

`attsurf2df(x)`

Arguments

`x` `list()`
List of `data.frames` or matrices. The names of the list give the percentiles of the attainment surfaces. This is the format returned by [eaf_as_list\(\)](#).

Value

```
data.frame()
```

Data frame with as many columns as objectives and an additional column percentiles.

See Also

```
eaf_as_list()
```

Examples

```
data(SPEA2relativeRichmond)
attsurfs <- eaf_as_list(eaf(SPEA2relativeRichmond, percentiles = c(0,50,100)))
str(attsurfs)
eaf_df <- attsurf2df(attsurfs)
str(eaf_df)
```

choose_eafdiff	<i>Interactively choose according to empirical attainment function differences</i>
----------------	--

Description

Interactively choose according to empirical attainment function differences

Usage

```
choose_eafdiff(x, left = stop("'left' must be either TRUE or FALSE"))
```

Arguments

x	matrix() Matrix of rectangles representing EAF differences returned by <code>eafdiff()</code> with <code>rectangles=TRUE</code> .
left	logical(1) With <code>left=TRUE</code> return the rectangles with positive differences, otherwise return those with negative differences but differences are converted to positive.

Value

matrix() where the first 4 columns give the coordinates of two corners of each rectangle and the last column. In both cases, the last column gives the positive differences in favor of the chosen side.

Examples

```

extdata_dir <- system.file(package="moocore", "extdata")
A1 <- read_datasets(file.path(extdata_dir, "wrots_l100w10_dat"))
A2 <- read_datasets(file.path(extdata_dir, "wrots_l10w100_dat"))
# Choose A1
rectangles <- eafdiff(A1, A2, intervals = 5, rectangles = TRUE)
rectangles <- choose_eafdiff(rectangles, left = TRUE)
reference <- c(max(A1[, 1], A2[, 1]), max(A1[, 2], A2[, 2]))
x <- split.data.frame(A1[,1:2], A1[,3])
hv_A1 <- sapply(split.data.frame(A1[, 1:2], A1[, 3]),
               hypervolume, reference=reference)
hv_A2 <- sapply(split.data.frame(A2[, 1:2], A2[, 3]),
               hypervolume, reference=reference)
print(fivenum(hv_A1))
print(fivenum(hv_A2))
whv_A1 <- sapply(split.data.frame(A1[, 1:2], A1[, 3]),
               whv_rect, rectangles=rectangles, reference=reference)
whv_A2 <- sapply(split.data.frame(A2[, 1:2], A2[, 3]),
               whv_rect, rectangles=rectangles, reference=reference)
print(fivenum(whv_A1))
print(fivenum(whv_A2))

```

compute_eaf_call	<i>Same as <code>eaf()</code> but performs no checks and does not transform the input or the output. This function should be used by other packages that want to avoid redundant checks and transformations.</i>
------------------	--

Description

Same as `eaf()` but performs no checks and does not transform the input or the output. This function should be used by other packages that want to avoid redundant checks and transformations.

Usage

```
compute_eaf_call(x, cumsizes, percentiles)
```

Arguments

x	matrix() data.frame() Matrix or data frame of numerical values that represents multiple sets of points, where each row represents a point. If sets is missing, the last column of x gives the sets.
cumsizes	integer() Cumulative size of the different sets of points in x.
percentiles	numeric() Vector indicating which percentiles are computed (values between 0 and 100). NULL computes all.

Value

`data.frame()`

A data frame containing the exact representation of EAF. The last column gives the percentile that corresponds to each point. If `groups` is not `NULL`, then an additional column indicates to which group the point belongs.

See Also

[as_double_matrix\(\)](#) [transform_maximise\(\)](#)

<code>compute_eafdiff_call</code>	<i>Same as eafdiff() but performs no checks and does not transform the input or the output. This function should be used by other packages that want to avoid redundant checks and transformations.</i>
-----------------------------------	---

Description

Same as [eafdiff\(\)](#) but performs no checks and does not transform the input or the output. This function should be used by other packages that want to avoid redundant checks and transformations.

Usage

```
compute_eafdiff_call(x, y, cumsizes_x, cumsizes_y, intervals, ret)
```

Arguments

<code>x, y</code>	<code>matrix data.frame()</code> Data frames corresponding to the input data of left and right sides, respectively. Each data frame has at least three columns, the last one is the set of each point. See also read_datasets() .
<code>cumsizes_x, cumsizes_y</code>	Cumulative size of the different sets of points in <code>x</code> and <code>y</code> .
<code>intervals</code>	<code>integer(1)</code> The absolute range of the differences $[0, 1]$ is partitioned into the number of intervals provided.
<code>ret</code>	<code>("points" "rectangles" "polygons")</code> The format of the returned EAF differences.

Value

With `rectangle=FALSE`, a `data.frame` containing points where there is a transition in the value of the EAF differences. With `rectangle=TRUE`, a `matrix` where the first 4 columns give the coordinates of two corners of each rectangle. In both cases, the last column gives the difference in terms of sets in `x` minus sets in `y` that attain each point (i.e., negative values are differences in favour `y`).

See Also

[as_double_matrix\(\)](#) [transform_maximise\(\)](#)

CPFs	<i>Conditional Pareto fronts obtained from Gaussian processes simulations.</i>
------	--

Description

The data has the only goal of providing an example of use of `vorob_t()` and `vorob_dev()`. It has been obtained by fitting two Gaussian processes on 20 observations of a bi-objective problem, before generating conditional simulation of both GPs at different locations and extracting non-dominated values of coupled simulations.

Usage

CPFs

Format

A data frame with 2967 observations on the following 3 variables.

f1 first objective values.

f2 second objective values.

set indices of corresponding conditional Pareto fronts.

Source

Binois M, Ginsbourger D, Roustant O (2015). “Quantifying uncertainty on Pareto fronts with Gaussian process conditional simulations.” *European Journal of Operational Research*, **243**(2), 386–394. doi:[10.1016/j.ejor.2014.07.032](https://doi.org/10.1016/j.ejor.2014.07.032).

Examples

```
data(CPFs)
vorob_t(CPFs, reference = c(2, 200))
```

eaf	<i>Exact computation of the Empirical Attainment Function (EAF)</i>
-----	---

Description

This function computes the EAF given a set of 2D or 3D points and a vector set that indicates to which set each point belongs.

Usage

```
eaf(x, sets, percentiles = NULL, maximise = FALSE, groups = NULL)
```

Arguments

<code>x</code>	<code>matrix()</code> <code>data.frame()</code> Matrix or data frame of numerical values that represents multiple sets of points, where each row represents a point. If <code>sets</code> is missing, the last column of <code>x</code> gives the sets.
<code>sets</code>	<code>integer()</code> Vector that indicates the set of each point in <code>x</code> . If missing, the last column of <code>x</code> is used instead.
<code>percentiles</code>	<code>numeric()</code> Vector indicating which percentiles are computed (values between 0 and 100). <code>NULL</code> computes all.
<code>maximise</code>	<code>logical()</code> Whether the objectives must be maximised instead of minimised. Either a single logical value that applies to all objectives or a vector of logical values, with one value per objective.
<code>groups</code>	<code>factor()</code> Indicates that the EAF must be computed separately for data belonging to different groups.

Details

The empirical first-order attainment function (EAF) is used to assess the performance of stochastic multiobjective optimisers such as multiobjective evolutionary algorithms (Grunert da Fonseca et al. 2001). It is an estimator for the first-order attainment function, which provides information about the location and, to some extent, the variability of the random sets of nondominated objective vectors produced by such optimisers when applied to given problem instances.

Given a set $A \subset \mathbb{R}^d$, the attainment function of A , denoted by $\alpha_A: \mathbb{R}^d \rightarrow \{0, 1\}$, specifies which points in the objective space are weakly dominated by A , where $\alpha_A(\vec{z}) = 1$ if $\exists \vec{a} \in A, \vec{a} \leq \vec{z}$, and $\alpha_A(\vec{z}) = 0$, otherwise.

Let $\mathcal{A} = \{A_1, \dots, A_n\}$ be a multi-set of n sets $A_i \subset \mathbb{R}^d$, the EAF (Grunert da Fonseca et al. 2001; Grunert da Fonseca and Fonseca 2010) is the function $\hat{\alpha}_{\mathcal{A}}: \mathbb{R}^d \rightarrow [0, 1]$, such that:

$$\hat{\alpha}_{\mathcal{A}}(\vec{z}) = \frac{1}{n} \sum_{i=1}^n \alpha_{A_i}(\vec{z})$$

The EAF is a coordinate-wise non-decreasing step function, similar to the empirical cumulative distribution function (ECDF) (López-Ibáñez et al. 2025). Thus, a finite representation of the EAF can be computed as the set of minima, in terms of Pareto optimality, with a value of the EAF not smaller than a given t/n , where $t = 1, \dots, n$ (Fonseca et al. 2011). Formally, the EAF can be represented by the sequence $(L_1, L_2, \dots, L_n)$, where:

$$L_t = \min\{\vec{z} \in \mathbb{R}^d : \hat{\alpha}_{\mathcal{A}}(\vec{z}) \geq t/n\}$$

It is also common to refer to the $k\% \in [0, 100]$ percentile. For example, the *median* (or 50%) attainment surface corresponds to $L_{\lceil n/2 \rceil}$ and it is the lower boundary of the vector space attained by at least 50% of the input sets A_i . Similarly, L_1 is called the *best* attainment surface ($\frac{1}{n}\%$) and

represents the lower boundary of the space attained by at least one input set, whereas L_{100} is called the *worst* attainment surface (100%) and represents the lower boundary of the space attained by all input sets.

In the current implementation, the EAF is computed using the algorithms proposed by Fonseca et al. (2011), which have complexity $\Theta(m \log m + nm)$ in 2D and $O(n^2 m \log m)$ in 3D, where n is the number of input sets and m is the total number of input points.

Value

`data.frame()`

A data frame containing the exact representation of EAF. The last column gives the percentile that corresponds to each point. If `groups` is not `NULL`, then an additional column indicates to which group the point belongs.

Note

There are several examples of data sets in `system.file(package="moocore", "extdata")`. The current implementation only supports two and three dimensional points.

Author(s)

Manuel López-Ibáñez

References

Fonseca CM, Guerreiro AP, López-Ibáñez M, Paquete L (2011). “On the Computation of the Empirical Attainment Function.” In Takahashi RHC, Deb K, Wanner EF, Greco S (eds.), *Evolutionary Multi-criterion Optimization, EMO 2011*, volume 6576 of *Lecture Notes in Computer Science*, 106–120. Springer, Berlin~/ Heidelberg. doi:10.1007/9783642198939_8.

Grunert da Fonseca V, Fonseca CM (2010). “The Attainment-Function Approach to Stochastic Multiobjective Optimizer Assessment and Comparison.” In Bartz-Beielstein T, Chiarandini M, Paquete L, Preuss M (eds.), *Experimental Methods for the Analysis of Optimization Algorithms*, 103–130. Springer, Berlin~/ Heidelberg. doi:10.1007/9783642025389_5.

Grunert da Fonseca V, Fonseca CM, Hall AO (2001). “Inferential Performance Assessment of Stochastic Optimisers and the Attainment Function.” In Zitzler E, Deb K, Thiele L, Coello Coello CA, Corne D (eds.), *Evolutionary Multi-criterion Optimization, EMO 2001*, volume 1993 of *Lecture Notes in Computer Science*, 213–225. Springer, Berlin~/ Heidelberg. doi:10.1007/354044719-9_15.

López-Ibáñez M, Vermetten D, Dreó J, Doerr C (2025). “Using the Empirical Attainment Function for Analyzing Single-objective Black-box Optimization Algorithms.” *IEEE Transactions on Evolutionary Computation*, 29(5), 1774–1782. doi:10.1109/TEVC.2024.3462758.

See Also

[read_datasets\(\)](#)

Examples

```
extdata_path <- system.file(package="moocore", "extdata")

x <- read_datasets(file.path(extdata_path, "example1_dat"))
# Compute full EAF (sets is the last column)
str(eaf(x))

# Compute only best, median and worst
str(eaf(x[,1:2], sets = x[,3], percentiles = c(0, 50, 100)))

x <- read_datasets(file.path(extdata_path, "spherical-250-10-3d.txt"))
y <- read_datasets(file.path(extdata_path, "uniform-250-10-3d.txt"))
x <- rbind(data.frame(x, groups = "spherical"),
           data.frame(y, groups = "uniform"))
# Compute only median separately for each group
z <- eaf(x[,1:3], sets = x[,4], groups = x[,5], percentiles = 50)
str(z)
```

eaf_as_list

Convert an EAF data frame to a list of data frames, where each element of the list is one attainment surface. The function [attsurf2df\(\)](#) can be used to convert the list into a single data frame.

Description

Convert an EAF data frame to a list of data frames, where each element of the list is one attainment surface. The function [attsurf2df\(\)](#) can be used to convert the list into a single data frame.

Usage

```
eaf_as_list(eaf)
```

Arguments

eaf	<code>data.frame()</code> <code>matrix()</code> Data frame or matrix that represents the EAF.
-----	--

Value

```
list()
```

A list of data frames. Each `data.frame` represents one attainment surface.

See Also

[eaf\(\)](#) [attsurf2df\(\)](#)

Examples

```
extdata_path <- system.file(package="moocore", "extdata")
x <- read_datasets(file.path(extdata_path, "example1_dat"))
attsurfs <- eaf_as_list(eaf(x, percentiles = c(0, 50, 100)))
str(attsurfs)
```

eafdiff

*Compute empirical attainment function differences***Description**

Calculate the differences between the empirical attainment functions of two data sets.

Usage

```
eafdiff(x, y, intervals = NULL, maximise = FALSE, rectangles = FALSE)
```

Arguments

<code>x, y</code>	<code>matrix data.frame()</code> Data frames corresponding to the input data of left and right sides, respectively. Each data frame has at least three columns, the last one is the set of each point. See also read_datasets() .
<code>intervals</code>	<code>integer(1)</code> The absolute range of the differences $[0, 1]$ is partitioned into the number of intervals provided.
<code>maximise</code>	<code>logical()</code> Whether the objectives must be maximised instead of minimised. Either a single logical value that applies to all objectives or a vector of logical values, with one value per objective.
<code>rectangles</code>	<code>logical(1)</code> If TRUE, the output is in the form of rectangles of the same color.

Details

This function calculates the differences between the EAFs of two data sets.

Value

With `rectangle=FALSE`, a `data.frame` containing points where there is a transition in the value of the EAF differences. With `rectangle=TRUE`, a `matrix` where the first 4 columns give the coordinates of two corners of each rectangle. In both cases, the last column gives the difference in terms of sets in `x` minus sets in `y` that attain each point (i.e., negative values are differences in favour `y`).

See Also

[read_datasets\(\)](#)

Examples

```
A1 <- read_datasets(text='
3 2
2 3

2.5 1
1 2

1 2
')
```

```
A2 <- read_datasets(text='
4 2.5
3 3
2.5 3.5

3 3
2.5 3.5

2 1
')
d <- eafdiff(A1, A2)
str(d)
d
```

```
d <- eafdiff(A1, A2, rectangles = TRUE)
str(d)
d
```

Description

Computes the epsilon metric, either additive or multiplicative.

Usage

```
epsilon_additive(x, reference, maximise = FALSE)
```

```
epsilon_mult(x, reference, maximise = FALSE)
```

Arguments

x	matrix() data.frame() Matrix or data frame of numerical values, where each row gives the coordinates of a point.
reference	matrix data.frame Reference set as a matrix or data.frame of numerical values.
maximise	logical() Whether the objectives must be maximised instead of minimised. Either a single logical value that applies to all objectives or a vector of logical values, with one value per objective.

Details

The epsilon metric of a set $A \subset \mathbb{R}^m$ with respect to a reference set $R \subset \mathbb{R}^m$ is defined as

$$\epsilon(A, R) = \max_{r \in R} \min_{a \in A} \max_{1 \leq i \leq m} \epsilon(a_i, r_i)$$

where a and r are objective vectors of length m .

In the case of minimization of objective i , $\epsilon(a_i, r_i)$ is computed as a_i/r_i for the multiplicative variant (respectively, $a_i - r_i$ for the additive variant), whereas in the case of maximization of objective i , $\epsilon(a_i, r_i) = r_i/a_i$ for the multiplicative variant (respectively, $r_i - a_i$ for the additive variant). This allows computing a single value for problems where some objectives are to be maximized while others are to be minimized. Moreover, a lower value corresponds to a better approximation set, independently of the type of problem (minimization, maximization or mixed). However, the meaning of the value is different for each objective type. For example, imagine that objective 1 is to be minimized and objective 2 is to be maximized, and the multiplicative epsilon computed here for $\epsilon(A, R) = 3$. This means that A needs to be multiplied by 1/3 for all a_1 values and by 3 for all a_2 values in order to weakly dominate R .

The multiplicative variant can be computed as $\exp(\epsilon_{+}(\log(A), \log(R)))$, which makes clear that the computation of the multiplicative version for zero or negative values doesn't make sense. See the examples below.

The current implementation uses the naive algorithm that requires $O(m \cdot |A| \cdot |R|)$, where m is the number of objectives (dimension of vectors).

Value

numeric(1) A single numerical value.

Author(s)

Manuel López-Ibáñez

References

Zitzler E, Thiele L, Laumanns M, Fonseca CM, Grunert da Fonseca V (2003). “Performance Assessment of Multiobjective Optimizers: an Analysis and Review.” *IEEE Transactions on Evolutionary Computation*, 7(2), 117–132. doi:10.1109/TEVC.2003.810758.

Examples

```
# Fig 6 from Zitzler et al. (2003).
A1 <- matrix(c(9,2,8,4,7,5,5,6,4,7), ncol=2, byrow=TRUE)
A2 <- matrix(c(8,4,7,5,5,6,4,7), ncol=2, byrow=TRUE)
A3 <- matrix(c(10,4,9,5,8,6,7,7,6,8), ncol=2, byrow=TRUE)
if (requireNamespace("graphics", quietly = TRUE)) {
  plot(A1, xlab=expression(f[1]), ylab=expression(f[2]),
       panel.first=grid(nx=NULL), pch=4, cex=1.5, xlim = c(0,10), ylim=c(0,8))
  points(A2, pch=0, cex=1.5)
  points(A3, pch=1, cex=1.5)
  legend("bottomleft", legend=c("A1", "A2", "A3"), pch=c(4,0,1),
        pt.bg="gray", bg="white", bty = "n", pt.cex=1.5, cex=1.2)
}
epsilon_mult(A1, A3) # A1 epsilon-dominates A3 => e = 9/10 < 1
epsilon_mult(A1, A2) # A1 weakly dominates A2 => e = 1
epsilon_mult(A2, A1) # A2 is epsilon-dominated by A1 => e = 2 > 1
# Equivalence between additive and multiplicative
exp(epsilon_additive(log(A2), log(A1)))

# A more realistic example
extdata_path <- system.file(package="moocore", "extdata")
path.A1 <- file.path(extdata_path, "ALG_1_dat.xz")
path.A2 <- file.path(extdata_path, "ALG_2_dat.xz")
A1 <- read_datasets(path.A1)[,1:2]
A2 <- read_datasets(path.A2)[,1:2]
ref <- filter_dominated(rbind(A1, A2))
epsilon_additive(A1, ref)
epsilon_additive(A2, ref)
# Multiplicative version of epsilon metric
ref <- filter_dominated(rbind(A1, A2))
epsilon_mult(A1, ref)
epsilon_mult(A2, ref)
```

Description

Generate a random set of n mutually nondominated points of dimension d with the shape defined by method.

When `integer = FALSE` (the default), the points are generated within the hypercube $(0, 1)^d$ and can be scaled to another range using `normalise()`. Otherwise, points are scaled to a non-negative integer range that keeps the points mutually nondominated.

See the visualisations in the vignette [Sampling Random Nondominated Sets](#).

Usage

```
generate_ndset(n, d, method, seed = NULL, integer = FALSE)
```

Arguments

<code>n</code>	<code>integer(1)</code> Number of rows in the output.
<code>d</code>	<code>integer(1)</code> Number of columns in the output.
<code>method</code>	<code>character(1)</code> Method used to generate the random nondominated set. See Details below for more information.
<code>seed</code>	<code>integer(1)</code> Integer seed for random number generation. If <code>NULL</code> , a random seed is generated.
<code>integer</code>	<code>logical(1)</code> If <code>TRUE</code> , return integer-valued points.

Details

The available methods are:

'simplex', 'linear', or 'L' Uniformly samples points on the standard simplex. This shape of nondominated set is also called 'linear' in the literature (Lacour et al. 2017).

The standard $(d - 1)$ -simplex is defined by $\{x \in \mathbb{R}_+^d : \sum_i x_i = 1\}$. Each point $\vec{z} \in (0, 1)^d \subset \mathbb{R}^d$ is generated by sampling d independent and identically distributed values $(x_1, x_2, \dots, x_d)$ from the exponential distribution, then dividing each value by the L1-norm of the vector, $z_i = x_i / \sum_{i=1}^d x_i$ (Rubinstein and Melamed 1998). Values sampled from the exponential distribution are guaranteed to be positive. Sampling from either the standard normal distribution (Guerreiro et al. 2021) or the uniform distribution (Lacour et al. 2017) does not produce a uniform distribution when projected onto the simplex.

'concave-sphere', 'sphere', or 'C' Uniformly samples points on the positive orthant of the hypersphere, which is concave when all objectives are minimised.

Each point $\vec{z} \in (0, 1)^d \subset \mathbb{R}^d$ is generated by sampling d independent and identically distributed values $\vec{x} = (x_1, x_2, \dots, x_d)$ from the standard normal distribution, then dividing each value by the l2-norm of the vector, $z_i = \frac{|x_i|}{\|\vec{x}\|_2}$ (Muller 1959). The absolute value in the numerator ensures that points are sampled on the positive orthant of the hypersphere. Sampling from the uniform distribution (Lacour et al. 2017) would not result in a uniform sampling when projected onto the surface of the hypersphere.

'convex-sphere' or 'X' Equivalent to $1 - \text{generate_ndset}(\dots, \text{method}='concave-sphere')$, which is convex for minimisation. This shape has also been called *inverted convex* (Ishibuchi et al. 2019). This sampling is uniform.

It corresponds to translating points from the negative orthant of the hypersphere to the positive orthant. Thus, the sampling remains uniform.

'concave-cliff' Each point $\vec{z} \in (0,1)^d \subset \mathbb{R}^d$ is generated as follows (Emmerich and Fonseca 2011; Guerreiro and Fonseca 2018). Each pair of dimensions $(z_{2i-1}, z_{2i}), i = 1, 2, \dots, \lfloor d/2 \rfloor$, is generated by sampling a pair of independent and identically distributed values $\vec{x} = (x_{2i-1}, x_{2i})$ from the standard normal distribution, then dividing each value by the l2-norm of the vector: $z_{2i-1} = \frac{|x_{2i-1}|}{\|\vec{x}\|_2}$ and $z_{2i} = \frac{|x_{2i}|}{\|\vec{x}\|_2}$. This is equivalent to generating a 2D set using $\text{method}='concave-sphere'$ for each pair of dimensions. If d is odd, the last dimension is generated uniformly at random within the unit hypercube, on $[0, 1)$. This method requires at least 3 dimensions; in 2D it would be equivalent to $\text{method}='concave-sphere'$.

In the resulting set, each pair of columns is mutually nondominated in its projection to the plane formed by those dimensions. In the odd case, the last column does not provide any ordering. These sets are adversarial for dimension-sweep algorithms that aim to exploit dominance structure.

Dimensions generated uniformly at random should be chosen adversarially according to the algorithm being tested, e.g., the columns along which dimension-sweep occurs. Alternatively, the columns may be randomly shuffled.

'convex-cliff' Equivalent to $1 - \text{generate_ndset}(\dots, \text{method}='concave-cliff')$. This method does not make sense for $d=2$.

'concave-simplex' Equivalent to $1 - \text{generate_ndset}(\dots, \text{method}='convex-simplex')$, which is concave for minimisation. This shape has also been called *inverted concave* (Ishibuchi et al. 2019). This sampling is *not* uniform because $\text{method}='convex-simplex'$ is also not uniform.

'convex-simplex' Equivalent to $\text{generate_ndset}(\dots, \text{method}='simplex')^2$, which is convex for minimisation. Such a set cannot be obtained by any affine transformation of a subset of the hypersphere. This sampling is *not* uniform.

The corresponding surface is equivalent to a simplex curved towards the origin. The generated points $\vec{z} \in (0,1)^d \subset \mathbb{R}^d$ satisfy $\sum_{i=1}^d \sqrt{z_i} = 1$. Although the sampling on the simplex is uniform, the transformed points are not.

'inverted-simplex' or 'inverted-linear' Equivalent to $1 - \text{generate_ndset}(\dots, \text{method}='simplex')$. This sampling is uniform.

Methods 'inverted-simplex', 'concave-simplex' and 'convex-sphere' are translations of 'simplex', 'convex-simplex' and 'concave', respectively. These translations have been called *inverted* shapes in the literature and have different properties than their *regular* counterparts (Ishibuchi et al. 2019).

Value

A numeric matrix of size $n \times d$ containing nondominated points.

References

Emmerich MTM, Fonseca CM (2011). "Computing Hypervolume Contributions in Low Dimensions: Asymptotically Optimal Algorithm and Complexity Results." In Takahashi RHC, Deb K,

Wanner EF, Greco S (eds.), *Evolutionary Multi-criterion Optimization, EMO 2011*, volume 6576 of *Lecture Notes in Computer Science*, 121–135. Springer, Berlin~/ Heidelberg. doi:10.1007/978-3642198939_9.

Guerreiro AP, Fonseca CM (2018). “Computing and Updating Hypervolume Contributions in Up to Four Dimensions.” *IEEE Transactions on Evolutionary Computation*, **22**(3), 449–463. doi:10.1109/tevc.2017.2729550.

Guerreiro AP, Fonseca CM, Paquete L (2021). “The Hypervolume Indicator: Computational Problems and Algorithms.” *ACM Computing Surveys*, **54**(6), 1–42. doi:10.1145/3453474.

Ishibuchi H, He L, Shang K (2019). “Regular Pareto Front Shape is not Realistic.” In *Proceedings of the 2019 Congress on Evolutionary Computation (CEC 2019)*, 2034–2041. doi:10.1109/cec.2019.8790342.

Lacour R, Klamroth K, Fonseca CM (2017). “A box decomposition algorithm to compute the hypervolume indicator.” *Computers & Operations Research*, **79**, 347–360. doi:10.1016/j.cor.2016.06.021.

Muller ME (1959). “A Note on a Method for Generating Points Uniformly on N-Dimensional Spheres.” *Communications of the ACM*, **2**(4), 19–20. doi:10.1145/377939.377946.

Rubinstein RY, Melamed B (1998). *Modern simulation and modeling*. Wiley, New York, NY. Uniform sampling from the simplex.

Examples

```
generate_ndset(5, 3, "simplex", seed = 42)
generate_ndset(5, 3, "simplex", seed = 42, integer = TRUE)
generate_ndset(4, 2, "sphere", seed = 123)
generate_ndset(3, 5, "convex-sphere", seed = 123)
generate_ndset(3, 4, "concave-cliff", seed = 42)
generate_ndset(3, 5, "convex-cliff", seed = 42)
generate_ndset(4, 4, "convex-simplex", seed = 123)
```

hv_approx

Approximate the hypervolume indicator.

Description

Approximate the value of the hypervolume metric with respect to a given reference point assuming minimization of all objectives. Methods "Rphi-FWE+" and "DZ2019-HW" are deterministic and ignore the parameter seed, while method="DZ2019-MC" relies on Monte-Carlo sampling (Deng and Zhang 2019). All methods tend to get more accurate with higher values of nsamples, but the increase in accuracy is not monotonic, as shown in the vignette [Approximating the hypervolume](#).

Usage

```

hv_approx(
  x,
  reference,
  maximise = FALSE,
  nsamples = 262144L,
  seed = NULL,
  method = c("Rphi-FWE+", "DZ2019-HW", "DZ2019-MC")
)

```

Arguments

x	matrix() data.frame() Matrix or data frame of numerical values, where each row gives the coordinates of a point.
reference	numeric() Reference point as a vector of numerical values.
maximise	logical() Whether the objectives must be maximised instead of minimised. Either a single logical value that applies to all objectives or a vector of logical values, with one value per objective.
nsamples	integer(1) Number of samples for Monte-Carlo sampling. Higher values typically produce more accurate approximations of the true hypervolume, but require more time.
seed	integer(1) Random seed.
method	character(1) Method to generate the sampling weights. See ‘Details’.

Details

All available methods approximate the hypervolume as a $(m - 1)$ -dimensional integral over the surface of hypersphere (Deng and Zhang 2019):

$$\widehat{HV}_r(A) = \frac{2\pi^{\frac{m}{2}}}{\Gamma(\frac{m}{2})} \frac{1}{m2^m} \frac{1}{n} \sum_{i=1}^n \max_{y \in A} s(w^{(i)}, y)^m$$

where m is the number of objectives, $w^{(i)}$ are weights uniformly distributed on S_+ , i.e., the positive orthant of the $(m - 1)$ -D unit hypersphere, n is the number of weights sampled, $\Gamma()$ is the gamma function `gamma()`, i.e., the analytical continuation of the factorial function, and $s(w, y) = \min_{k=1}^m (r_k - y_k) / w_k$.

In the default method="Rphi-FWE+" (López-Ibáñez 2026), the weights $w^{(i)}, i = 1 \dots n$ are defined using the deterministic low-discrepancy sequence R_ϕ (Roberts 2018) mapped to the positive orthant of the hypersphere using a modified version of Fang and Wang efficient mapping (Fang and Wang 1994).

In method="DZ2019-HW" (Deng and Zhang 2019), the weights $w^{(i)}, i = 1 \dots n$ are defined using a deterministic low-discrepancy sequence. The weight values depend on their number (nsamples), thus increasing the number of weights may not necessarily increase accuracy because the set of weights would be different.

In method="DZ2019-MC" (Deng and Zhang 2019), the weights $w^{(i)}, i = 1 \dots n$ are sampled from the unit normal vector such that each weight $w = \frac{|x|}{\|x\|_2}$ where each component of x is independently sampled from the standard normal distribution (Muller 1959).

The original source code in C++/MATLAB for both "DZ2019-HW" and "DZ2019-MC" methods can be found at <https://github.com/Ksrma/Hypervolume-Approximation-using-polar-coordinate>.

López-Ibáñez (2026) empirically shows that "Rphi-FWE+" typically produces an approximation error as low as the other methods, with a computational cost similar to "DZ2019-MC" and significantly faster than "DZ2019-HW".

Value

A single numerical value.

Author(s)

Manuel López-Ibáñez

References

Deng J, Zhang Q (2019). "Approximating Hypervolume and Hypervolume Contributions Using Polar Coordinate." *IEEE Transactions on Evolutionary Computation*, **23**(5), 913–918. doi:10.1109/tevc.2019.2895108.

Fang KT, Wang Y (1994). *Number-Theoretic Methods in Statistics*. Chapman & Hall/CRC, London, UK.

López-Ibáñez M (2026). "Approximating the Hypervolume Indicator using Fast Low-Discrepancy Sequences." In Hu T, Trujillo L (eds.), *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2026*. ACM Press, New York, NY. doi:10.1145/3795095.3805198.

Muller ME (1959). "A Note on a Method for Generating Points Uniformly on N-Dimensional Spheres." *Communications of the ACM*, **2**(4), 19–20. doi:10.1145/377939.377946.

Roberts M (2018). "The Unreasonable Effectiveness of Quasirandom Sequences." <https://extremelearning.com.au/unreasonable-effectiveness-of-quasirandom-sequences/>. Last visited: 13/04/2025.

Examples

```
x <- matrix(c(5, 5, 4, 6, 2, 7, 7, 4), ncol=2, byrow=TRUE)
hypervolume(x, ref=10)
hv_approx(x, ref=10, method="Rphi-FWE+")
hv_approx(x, ref=10, method="DZ2019-HW")
hv_approx(x, ref=10, seed=42, method="DZ2019-MC")
```

hv_contributions	<i>Hypervolume contribution of a set of points</i>
------------------	--

Description

Computes the hypervolume contribution of each point of a set of points with respect to a given reference point. Duplicated and dominated points have zero contribution. By default, dominated points are ignored, that is, they do not affect the contribution of other points. See the Notes below for more details. For details about the hypervolume, see [hypervolume\(\)](#).

Usage

```
hv_contributions(x, reference, maximise = FALSE, ignore_dominated = TRUE)
```

Arguments

x	matrix() data.frame() Matrix or data frame of numerical values, where each row gives the coordinates of a point.
reference	numeric() Reference point as a vector of numerical values.
maximise	logical() Whether the objectives must be maximised instead of minimised. Either a single logical value that applies to all objectives or a vector of logical values, with one value per objective.
ignore_dominated	logical(1) Whether dominated points are ignored when computing the contribution of non-dominated points. The value of this parameter has an effect on the return values only if the input contains dominated points. Setting this to FALSE slows down the computation significantly. See the Notes below for a detailed explanation.

Details

The hypervolume contribution of point $\vec{p} \in X$ is defined as $\text{hvc}(\vec{p}) = \text{hyp}(X) - \text{hyp}(X \setminus \{\vec{p}\})$. This definition implies that duplicated points have zero contribution even if not dominated, because removing one of the duplicates does not change the hypervolume of the remaining set. Moreover, dominated points also have zero contribution. However, a point that is dominated by a single (dominating) nondominated point reduces the contribution of the latter, because removing the dominating point makes the dominated one become nondominated.

Handling this special case is non-trivial and makes the computation more expensive, thus the default (`ignore_dominated=TRUE`) ignores all dominated points in the input, that is, their contribution is set to zero and their presence does not affect the contribution of any other point. Setting `ignore_dominated=FALSE` will consider dominated points according to the mathematical definition given above, but the computation will be slower.

When the input only consists of mutually nondominated points, the value of `ignore_dominated` does not change the result, but the default value is significantly faster.

Like the hypervolume, computing the hypervolume contribution is #P-hard (Bringmann and Friedrich 2012).

The current implementation uses a $O(n \log n)$ dimension-sweep algorithm for 2D. With `ignore_dominated=TRUE`, the 3D case uses the HVC3D algorithm (Guerreiro and Fonseca 2018), which has $O(n \log n)$ complexity. Otherwise, the implementation uses the naive algorithm that requires calculating the hypervolume $|X| + 1$ times.

Value

`numeric()`
A numerical vector

Author(s)

Manuel López-Ibáñez

References

Bringmann K, Friedrich T (2012). “Approximating the Least Hypervolume Contributor: NP-Hard in General, But Fast in Practice.” *Theoretical Computer Science*, **425**, 104–116. doi:[10.1016/j.tcs.2010.09.026](https://doi.org/10.1016/j.tcs.2010.09.026).

Guerreiro AP, Fonseca CM (2018). “Computing and Updating Hypervolume Contributions in Up to Four Dimensions.” *IEEE Transactions on Evolutionary Computation*, **22**(3), 449–463. doi:[10.1109/tevc.2017.2729550](https://doi.org/10.1109/tevc.2017.2729550).

See Also

[hypervolume\(\)](#)

Examples

```
x <- matrix(c(5,1, 1,5, 4,2, 4,4, 5,1), ncol=2, byrow=TRUE)
hv_contributions(x, reference=c(6,6))
# hvc[(5,1)] = 0 = duplicated
# hvc[(1,5)] = 3 = (4 - 1) * (6 - 5)
# hvc[(4,2)] = 3 = (5 - 4) * (5 - 2)
# hvc[(4,4)] = 0 = dominated
# hvc[(5,1)] = 0 = duplicated
hv_contributions(x, reference=c(6,6), ignore_dominated = FALSE)
# hvc[(5,1)] = 0 = duplicated
# hvc[(1,5)] = 3 = (4 - 1) * (6 - 5)
# hvc[(4,2)] = 2 = (5 - 4) * (4 - 2)
# hvc[(4,4)] = 0 = dominated
# hvc[(5,1)] = 0 = duplicated
data(SPEA2minstoptimeRichmond)
# The second objective must be maximized
# We calculate the hypervolume contribution of each point of the union of all sets.
```

```

hv_contributions(SPEA2minstoptimeRichmond[, 1:2], reference = c(250, 0),
                  maximise = c(FALSE, TRUE))

# Duplicated points show zero contribution above, even if not
# dominated. However, filter_dominated removes all duplicates except
# one. Hence, there are more points below with nonzero contribution.
hv_contributions(filter_dominated(SPEA2minstoptimeRichmond[, 1:2], maximise = c(FALSE, TRUE)),
                  reference = c(250, 0), maximise = c(FALSE, TRUE))

```

HybridGA

Results of Hybrid GA on Vanzyl and Richmond water networks

Description

Results of Hybrid GA on Vanzyl and Richmond water networks

Usage

HybridGA

Format

A list with two data frames, each of them with three columns, as produced by `read_datasets()`.

`$vanzyl` data frame of results on Vanzyl network

`$richmond` data frame of results on Richmond network. The second column is filled with NA

Source

López-Ibáñez M (2009). *Operational Optimisation of Water Distribution Networks*. Ph.D. thesis, School of Engineering and the Built Environment, Edinburgh Napier University, UK. <https://lopez-ibanez.eu/publications#LopezIbanezPhD..>

Examples

```

data(HybridGA)
print(HybridGA$vanzyl)
print(HybridGA$richmond)

```

hypervolume

*Hypervolume metric***Description**

Compute the hypervolume metric with respect to a given reference point assuming minimization of all objectives.

Usage

```
hypervolume(x, reference, maximise = FALSE)
```

Arguments

x	matrix() data.frame() Matrix or data frame of numerical values, where each row gives the coordinates of a point.
reference	numeric() Reference point as a vector of numerical values.
maximise	logical() Whether the objectives must be maximised instead of minimised. Either a single logical value that applies to all objectives or a vector of logical values, with one value per objective.

Details

The hypervolume of a set of multidimensional points $A \subset \mathbb{R}^m$, where m is the dimension of the points, with respect to a reference point $\vec{r} \in \mathbb{R}^m$ is the volume of the region dominated by the set and bounded by the reference point (Zitzler and Thiele 1998). Points in A that do not strictly dominate $\vec{r}$ do not contribute to the hypervolume value, thus, ideally, the reference point must be strictly dominated by all points in the true Pareto front.

More precisely, the hypervolume is the **Lebesgue measure** of the union of axis-aligned hyperrectangles (**orthotopes**), where each hyperrectangle is defined by one point from $\vec{a} \in A$ and the reference point. The union of axis-aligned hyperrectangles is also called an *orthogonal polytope*.

The hypervolume is compatible with Pareto-optimality (Knowles and Corne 2002; Zitzler et al. 2003), that is, $\nexists A, B \subset \mathbb{R}^m$, such that A is better than B according to Pareto-optimality and $\text{hyp}(A) \leq \text{hyp}(B)$. In other words, if the hypervolume of a set is larger than the hypervolume of another, then we know for sure that the latter set cannot be better than the former in terms of Pareto-optimality. The hypervolume is also the only unary metric known that is complete according to Pareto-optimality (Zitzler et al. 2003), that is, if A is better than B according to Pareto-optimality, then $\text{hyp}(A) > \text{hyp}(B)$.

Like most measures of unions of high-dimensional geometric objects, computing the hypervolume is #P-hard (Bringmann and Friedrich 2010), which means that the best possible algorithm takes an exponential time on the number of objectives or points, in the worst-case.

For 2D and 3D, the algorithms used (Fonseca et al. 2006; Beume et al. 2009) have $O(n \log n)$ complexity, where n is the number of input points. The 3D case uses the HV3D⁺ algorithm (Guerreiro

and Fonseca 2018), which has the same complexity as the HV3D algorithm (Fonseca et al. 2006; Beume et al. 2009), but it is faster in practice.

For 4D, the algorithm used is HV4D⁺ (Guerreiro and Fonseca 2018), which has $O(n^2)$ complexity. Compared to the **original implementation**, this implementation correctly handles weakly dominated points and has been further optimized for speed.

For 5D or higher and up to 12 points, the implementation uses the inclusion-exclusion algorithm (Wu and Azam 2001), which has $O(m2^n)$ time and $O(n \cdot m)$ space complexity, but it is very fast for such small sets. For larger number of points, it uses a recursive algorithm (Fonseca et al. 2006) that computes 4D contributions (Guerreiro and Fonseca 2018) as the base case, resulting in a $O(n^{m-2})$ time complexity and $O(n)$ space complexity in the worst-case. Experimental results show that the pruning techniques used may reduce the time complexity even further. The original proposal (Fonseca et al. 2006) had the HV3D algorithm as the base case, giving a time complexity of $O(n^{m-2} \log n)$. Andreia P. Guerreiro enhanced the numerical stability of the algorithm by avoiding floating-point comparisons of partial hypervolumes.

The hypervolume of 1D inputs is defined as $\max(\emptyset, \text{ref} - \min(x))$.

Value

`numeric(1)` A single numerical value.

Author(s)

Manuel López-Ibáñez

References

- Beume N, Fonseca CM, López-Ibáñez M, Paquete L, Vahrenhold J (2009). “On the complexity of computing the hypervolume indicator.” *IEEE Transactions on Evolutionary Computation*, **13**(5), 1075–1082. doi:10.1109/TEVC.2009.2015575.
- Bringmann K, Friedrich T (2010). “Approximating the volume of unions and intersections of high-dimensional geometric objects.” *Computational Geometry*, **43**(6–7), 601–610. doi:10.1016/j.comgeo.2010.03.004.
- Fonseca CM, Paquete L, López-Ibáñez M (2006). “An improved dimension-sweep algorithm for the hypervolume indicator.” In *Proceedings of the 2006 Congress on Evolutionary Computation (CEC 2006)*, 1157–1163. doi:10.1109/CEC.2006.1688440.
- Guerreiro AP, Fonseca CM (2018). “Computing and Updating Hypervolume Contributions in Up to Four Dimensions.” *IEEE Transactions on Evolutionary Computation*, **22**(3), 449–463. doi:10.1109/tevc.2017.2729550.
- Knowles JD, Corne D (2002). “On Metrics for Comparing Non-Dominated Sets.” In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC’02)*, 711–716.
- Wu J, Azam S (2001). “Metrics for Quality Assessment of a Multiobjective Design Optimization Solution Set.” *Journal of Mechanical Design*, **123**(1), 18–25. doi:10.1115/1.1329875.

Zitzler E, Thiele L (1998). “Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Case Study.” In Eiben AE, Bäck T, Schoenauer M, Schwefel H (eds.), *Parallel Problem Solving from Nature – PPSN V*, volume 1498 of *Lecture Notes in Computer Science*, 292–301. Springer, Heidelberg, Germany. doi:10.1007/BFb0056872.

Zitzler E, Thiele L, Laumanns M, Fonseca CM, Grunert da Fonseca V (2003). “Performance Assessment of Multiobjective Optimizers: an Analysis and Review.” *IEEE Transactions on Evolutionary Computation*, 7(2), 117–132. doi:10.1109/TEVC.2003.810758.

Examples

```
dat = matrix(c(5, 5, 4, 6, 2, 7, 7, 4), ncol=2, byrow=TRUE)
hypervolume(dat, ref=c(10, 10))
hypervolume(dat, ref=0, maximise=TRUE)

data(SPEA2minstoptimeRichmond)
# The second objective must be maximized
# We calculate the hypervolume of the union of all sets.
hypervolume(SPEA2minstoptimeRichmond[, 1:2], reference = c(250, 0),
             maximise = c(FALSE, TRUE))
```

igd	<i>Inverted Generational Distance (IGD and IGD+) and Averaged Hausdorff Distance</i>
-----	--

Description

Functions to compute the inverted generational distance (IGD and IGD+) and the averaged Hausdorff distance between nondominated sets of points.

Usage

```
igd(x, reference, maximise = FALSE)

igd_plus(x, reference, maximise = FALSE)

avg_hausdorff_dist(x, reference, maximise = FALSE, p = 1L)
```

Arguments

x	matrix() data.frame() Matrix or data frame of numerical values, where each row gives the coordinates of a point.
reference	matrix data.frame Reference set as a matrix or data.frame of numerical values.
maximise	logical() Whether the objectives must be maximised instead of minimised. Either a single logical value that applies to all objectives or a vector of logical values, with one value per objective.

p integer(1)
Hausdorff distance parameter (default: 1L).

Details

The generational distance (GD) of a set A is defined as the distance between each point $a \in A$ and the closest point r in a reference set R , averaged over the size of A . Formally,

$$GD_p(A, R) = \left(\frac{1}{|A|} \sum_{a \in A} \min_{r \in R} d(a, r)^p \right)^{\frac{1}{p}}$$

where the distance in our implementation is the Euclidean distance:

$$d(a, r) = \sqrt{\sum_{k=1}^m (a_k - r_k)^2}$$

The inverted generational distance (IGD) is calculated as $IGD_p(A, R) = GD_p(R, A)$.

The modified inverted generational distance (IGD+) was proposed by Ishibuchi et al. (2015) to ensure that IGD+ is weakly Pareto compliant, similarly to `epsilon_additive()` or `epsilon_mult()`. Formally,

$$IGD^+(A, R) = \left(\frac{1}{|R|} \sum_{r \in R} \min_{a \in A} d^+(r, a) \right)$$

Assuming minimization of all objectives, IGD+ uses the following modified distance:

$$d^+(r, a) = \sqrt{\sum_{k=1}^m (\max\{a_k - r_k, 0\})^2}$$

The average Hausdorff distance (Δ_p) was proposed by Schütze et al. (2012) and it is calculated as:

$$\Delta_p(A, R) = \max\{IGD_p(A, R), IGD_p(R, A)\}$$

IGDX (Zhou et al. 2009) is the application of IGD to decision vectors instead of objective vectors to measure closeness and diversity in decision space. One can use the functions `igd()` or `igd_plus()` (recommended) directly, just passing the decision vectors as data.

There are different formulations of the GD and IGD metrics in the literature that differ on the value of p , on the distance metric used and on whether the term $|A|^{-1}$ is inside (as above) or outside the exponent $1/p$. GD was first proposed by Van Veldhuizen and Lamont (1998) with $p = 2$ and the term $|A|^{-1}$ outside the exponent. Bosman and Thierens (2003) proposed the idea of IGD with $p = 1$ under the name D-metric. The name IGD seems to have been mentioned first by Coello Coello and Reyes-Sierra (2004). Schütze et al. (2012) proposed to place the term $|A|^{-1}$ inside the exponent, as in the formulation shown above. This has a significant effect for GD and less so for IGD given a constant reference set. IGD+ also follows this formulation. We refer to Ishibuchi et al. (2015) and Bezerra et al. (2017) for a more detailed historical perspective and a comparison of the various variants.

Following Ishibuchi et al. (2015), we always use $p = 1$ in our implementation of IGD and IGD+ because (1) it is the setting most used in recent works; (2) it makes irrelevant whether the term $|A|^{-1}$ is inside or outside the exponent $1/p$; and (3) the meaning of IGD becomes the average Euclidean distance from each reference point to its nearest objective vector. It is also slightly faster to compute.

GD should never be used directly to compare the quality of approximations to a Pareto front, because it is not weakly Pareto-compliant and often contradicts Pareto optimality.

IGD is still popular due to historical reasons, but we strongly recommend IGD+ instead of IGD, because IGD contradicts Pareto optimality in some cases (see examples below) whereas IGD+ is weakly Pareto-compliant.

The average Hausdorff distance $\Delta_p(A, R)$ is also not weakly Pareto-compliant, as shown in the examples below.

Value

numeric(1) A single numerical value.

Author(s)

Manuel López-Ibáñez

References

- Bezerra LCT, López-Ibáñez M, Stützle T (2017). “An Empirical Assessment of the Properties of Inverted Generational Distance Indicators on Multi- and Many-objective Optimization.” In Trautmann H, Rudolph G, Klamroth K, Schütze O, Wiecek MM, Jin Y, Grimme C (eds.), *Evolutionary Multi-criterion Optimization, EMO 2017*, volume 10173 of *Lecture Notes in Computer Science*, 31–45. Springer International Publishing, Cham, Switzerland. doi:10.1007/9783319541570_3.
- Bosman PAN, Thierens D (2003). “The balance between proximity and diversity in multiobjective evolutionary algorithms.” *IEEE Transactions on Evolutionary Computation*, 7(2), 174–188. doi:10.1109/TEVC.2003.810761.
- Coello Coello CA, Reyes-Sierra M (2004). “A Study of the Parallelization of a Coevolutionary Multi-objective Evolutionary Algorithm.” In Monroy R, Arroyo-Figueroa G, Sucar LE, Sossa H (eds.), *Proceedings of MICAI*, volume 2972 of *Lecture Notes in Artificial Intelligence*, 688–697. Springer, Heidelberg, Germany.
- Ishibuchi H, Masuda H, Tanigaki Y, Nojima Y (2015). “Modified Distance Calculation in Generational Distance and Inverted Generational Distance.” In Gaspar-Cunha A, Antunes CH, Coello Coello CA (eds.), *Evolutionary Multi-criterion Optimization, EMO 2015 Part I*, volume 9018 of *Lecture Notes in Computer Science*, 110–125. Springer, Heidelberg, Germany.
- Schütze O, Esquivel X, Lara A, Coello Coello CA (2012). “Using the Averaged Hausdorff Distance as a Performance Measure in Evolutionary Multiobjective Optimization.” *IEEE Transactions on Evolutionary Computation*, 16(4), 504–522. doi:10.1109/TEVC.2011.2161872.
- Van Veldhuizen DA, Lamont GB (1998). “Evolutionary Computation and Convergence to a Pareto

Front.” In Koza JR (ed.), *Late Breaking Papers at the Genetic Programming 1998 Conference*, 221–228.

Zhou A, Zhang Q, Jin Y (2009). “Approximating the set of Pareto-optimal solutions in both the decision and objective spaces by an estimation of distribution algorithm.” *IEEE Transactions on Evolutionary Computation*, **13**(5), 1167–1189. doi:10.1109/TEVC.2009.2021467.

Examples

```
# Example 4 from Ishibuchi et al. (2015)
ref <- matrix(c(10,0,6,1,2,2,1,6,0,10), ncol=2, byrow=TRUE)
A <- matrix(c(4,2,3,3,2,4), ncol=2, byrow=TRUE)
B <- matrix(c(8,2,4,4,2,8), ncol=2, byrow=TRUE)
if (requireNamespace("graphics", quietly = TRUE)) {
  plot(ref, xlab=expression(f[1]), ylab=expression(f[2]),
        panel.first=grid(nx=NULL), pch=23, bg="gray", cex=1.5)
  points(A, pch=1, cex=1.5)
  points(B, pch=19, cex=1.5)
  legend("topright", legend=c("Reference", "A", "B"), pch=c(23,1,19),
        pt.bg="gray", bg="white", bty = "n", pt.cex=1.5, cex=1.2)
}
cat("A is better than B in terms of Pareto optimality,\n however, IGD(A)=",
    igd(A, ref), "> IGD(B)=", igd(B, ref),
    "and AvgHausdorff(A)=", avg_hausdorff_dist(A, ref),
    "> AvgHausdorff(B)=", avg_hausdorff_dist(B, ref),
    ", which both contradict Pareto optimality.\nBy contrast, IGD+(A)=",
    igd_plus(A, ref), "< IGD+(B)=", igd_plus(B, ref), ", which is correct.\n")
# A less trivial example.
extdata_path <- system.file(package="moocore", "extdata")
path.A1 <- file.path(extdata_path, "ALG_1_dat.xz")
path.A2 <- file.path(extdata_path, "ALG_2_dat.xz")
A1 <- read_datasets(path.A1)[,1:2]
A2 <- read_datasets(path.A2)[,1:2]
ref <- filter_dominated(rbind(A1, A2))
igd(A1, ref)
igd(A2, ref)

# IGD+ (Pareto compliant)
igd_plus(A1, ref)
igd_plus(A2, ref)

# Average Hausdorff distance
avg_hausdorff_dist(A1, ref)
avg_hausdorff_dist(A2, ref)
```

Description

Identify nondominated points with `is_nondominated()` and remove dominated ones with `filter_dominated()`. `any_dominated()` quickly detects if a set contains any dominated point.

Usage

```
is_nondominated(x, maximise = FALSE, keep_weakly = FALSE)
```

```
filter_dominated(x, maximise = FALSE, keep_weakly = FALSE)
```

```
any_dominated(x, maximise = FALSE, keep_weakly = FALSE)
```

Arguments

x	matrix() data.frame() Matrix or data frame of numerical values, where each row gives the coordinates of a point.
maximise	logical() Whether the objectives must be maximised instead of minimised. Either a single logical value that applies to all objectives or a vector of logical values, with one value per objective.
keep_weakly	logical(1) If FALSE, return FALSE for any duplicates of nondominated points, except the last one.

Details

Given n points of dimension m , the current implementation always uses the best-known $O(n \log n)$ dimension-sweep algorithm (Kung et al. 1975) for $m \leq 3$. For $m \geq 4$, functions `is_nondominated()` and `filter_dominated()` use the best-known $O(n \log^{m-2} n)$ algorithm (Kung et al. 1975) when $n > 16$, and the naive $O(mn^2)$ algorithm otherwise. Function `any_dominated()` always uses the naive algorithm for $m \geq 4$.

Value

`is_nondominated()` returns a logical vector of the same length as the number of rows of data, where TRUE means that the point is not dominated by any other point.

`filter_dominated()` returns a matrix or data.frame with only mutually nondominated points.

`any_dominated()` returns TRUE if x contains any (weakly-)dominated points, FALSE otherwise.

Author(s)

Manuel López-Ibáñez

References

Kung HT, Luccio F, Preparata FP (1975). “On Finding the Maxima of a Set of Vectors.” *Journal of the ACM*, **22**(4), 469–476. doi:[10.1145/321906.321910](https://doi.org/10.1145/321906.321910).

See Also[pareto_rank\(\)](#)**Examples**

```

S = matrix(c(1,1,0,1,1,0,1,0), ncol = 2, byrow = TRUE)
is_nondominated(S)
is_nondominated(S, maximise = TRUE)
filter_dominated(S)
filter_dominated(S, keep_weakly = TRUE)
any_dominated(S)
any_dominated(S, keep_weakly = TRUE)
any_dominated(filter_dominated(S))
path_A1 <- file.path(system.file(package="moocore"), "extdata", "ALG_1_dat.xz")
set <- read_datasets(path_A1)[,1:2]
is_nondom <- is_nondominated(set)
cat("There are ", sum(is_nondom), " nondominated points\n")

if (requireNamespace("graphics", quietly = TRUE)) {
  plot(set, col = "blue", type = "p", pch = 20)
  ndset <- filter_dominated(set)
  points(ndset[order(ndset[,1]),], col = "red", pch = 21)
}

```

largest_eafdiff	<i>Identify largest EAF differences</i>
-----------------	---

Description

Given a list of datasets, return the indexes of the pair with the largest EAF differences according to the method proposed by Diaz and López-Ibáñez (2021).

Usage

```
largest_eafdiff(x, maximise = FALSE, intervals = 5L, reference, ideal = NULL)
```

Arguments

x	<code>list()</code> A list of matrices or data frames with at least 3 columns (last column indicates the set).
maximise	<code>logical()</code> Whether the objectives must be maximised instead of minimised. Either a single logical value that applies to all objectives or a vector of logical values, with one value per objective.
intervals	<code>integer(1)</code> The absolute range of the differences $[0, 1]$ is partitioned into the number of intervals provided.

reference	numeric() Reference point as a vector of numerical values.
ideal	numeric() Ideal point as a vector of numerical values. If NULL, it is calculated as minimum (or maximum if maximising that objective) of each objective in the input data.

Value

list()
A list with two components pair and value.

References

Diaz JE, López-Ibáñez M (2021). “Incorporating Decision-Maker’s Preferences into the Automatic Configuration of Bi-Objective Optimisation Algorithms.” *European Journal of Operational Research*, **289**(3), 1209–1222. doi:10.1016/j.ejor.2020.07.059.

Examples

```
# FIXME: This example is too large, we need a smaller one.
data(tpls50x20_1_MWT)
nadir <- apply(tpls50x20_1_MWT[,2:3], 2L, max)
x <- largest_eafdiff(split.data.frame(tpls50x20_1_MWT[,2:4], tpls50x20_1_MWT[, 1L]),
                    reference = nadir)

str(x)
```

normalise	<i>Normalise points</i>
-----------	-------------------------

Description

Normalise points per coordinate to a range, e.g., $c(1, 2)$, where the minimum value will correspond to 1 and the maximum to 2. If bounds are given, they are used for the normalisation.

Usage

```
normalise(x, to_range = c(1, 2), lower = NA, upper = NA, maximise = FALSE)
```

Arguments

x	matrix() data.frame() Matrix or data frame of numerical values, where each row gives the coordinates of a point.
to_range	numerical(2) Normalise values to this range. If the objective is maximised, it is normalised to $c(\text{to_range}[1], \text{to_range}[0])$ instead.

lower, upper	numerical() Bounds on the values. If NA, the maximum and minimum values of each coordinate are used.
maximise	logical() Whether the objectives must be maximised instead of minimised. Either a single logical value that applies to all objectives or a vector of logical values, with one value per objective.

Value

matrix()
A numerical matrix

Author(s)

Manuel López-Ibáñez

Examples

```
data(SPEA2minstoptimeRichmond)
# The second objective must be maximized
head(SPEA2minstoptimeRichmond[, 1:2])

head(normalise(SPEA2minstoptimeRichmond[, 1:2], maximise = c(FALSE, TRUE)))

head(normalise(SPEA2minstoptimeRichmond[, 1:2], to_range = c(0,1), maximise = c(FALSE, TRUE)))
```

pareto_rank	<i>Rank points according to Pareto-optimality (nondominated sorting).</i>
-------------	---

Description

`pareto_rank()` is meant to be used like `rank()`, but it assigns ranks according to Pareto dominance, where rank 1 indicates those solutions not dominated by any other solution in the input set. Duplicated points are assigned the same rank. The resulting ranking can be used to partition points into a list of matrices, each matrix representing a nondominated front (Deb et al. 2002) (see examples below).

Usage

```
pareto_rank(x, maximise = FALSE)
```

Arguments

<code>x</code>	<code>matrix() data.frame()</code> Matrix or data frame of numerical values, where each row gives the coordinates of a point.
<code>maximise</code>	<code>logical()</code> Whether the objectives must be maximised instead of minimised. Either a single logical value that applies to all objectives or a vector of logical values, with one value per objective.

Details

Given $x, y \in \mathbb{R}^m$, let $x \prec y$ denote that x dominates y according to Pareto optimality.

Nondominated sorting, also called the *layers-of-maxima problem* (Buchsbaum and Goodrich 2004), partitions a finite set of points $X \subset \mathbb{R}^m$ into an ordered sequence of *fronts*, $F_1, F_2, \dots, F_k$, where $1 \leq k \leq |X|$, satisfying the following conditions:

1. All points are allocated to a front:

$$\bigcup_{i=1}^k F_i = X$$

2. Each point is allocated to only one front:

$$F_i \cap F_j = \emptyset, \forall i, j \in \{1, 2, \dots, k\}, i \neq j$$

3. Points within a front are mutually nondominated:

$$\nexists x, y \in F_i, x \prec y, \forall i \in \{1, 2, \dots, k\}$$

4. The first front contains points not dominated by any other point:

$$\forall y \in F_1, \nexists x \in X, x \prec y$$

5. Every point in front i is dominated by at least one point in front $i - 1$:

$$\forall y \in F_i, \exists x \in F_{i-1}, x \prec y, \forall i \in \{2, 3, \dots, k\}$$

The rank returned by `pareto_rank()` is the index $i \in \{1, 2, \dots, k\}$ of the front F_i allocated to each point. If all points are mutually nondominated, there is only one front ($k = 1$). If each front contains one point, then $k = n = |X|$.

With $m = 2$, i.e., `ncol(data)=2`, the code uses the best-known $O(n \log n)$ algorithm by Jensen (2003). When $m \geq 3$, it uses the naive algorithm that identifies one *front* at a time, which requires $O(n^2 \log n)$ for $m = 3$, and $O(n^2 \log^{m-2} n)$ for $m \geq 4$.

Value

An integer vector of the same length as the number of rows of the input `x` with values within `[1, nrow(points)]`, where each value gives the rank of each point (lower is better).

References

Buchsbaum AL, Goodrich MT (2004). “Three-Dimensional Layers of Maxima.” *Algorithmica*, **39**, 275–289. doi:10.1007/s0045300410825.

Deb K, Pratap A, Agarwal S, Meyarivan T (2002). “A fast and elitist multi-objective genetic algorithm: NSGA-II.” *IEEE Transactions on Evolutionary Computation*, **6**(2), 182–197. doi:10.1109/4235.996017.

Jensen MT (2003). “Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms.” *IEEE Transactions on Evolutionary Computation*, **7**(5), 503–515. doi:10.1109/TEVC.2003.817234.

See Also

[is_nondominated\(\)](#)

Examples

```
three_fronts = matrix(c(1, 2, 3,
                        3, 1, 2,
                        2, 3, 1,
                        10, 20, 30,
                        30, 10, 20,
                        20, 30, 10,
                        100, 200, 300,
                        300, 100, 200,
                        200, 300, 100), ncol=3, byrow=TRUE)

pareto_rank(three_fronts)

split.data.frame(three_fronts, pareto_rank(three_fronts))
path_A1 <- file.path(system.file(package="moocore"), "extdata", "ALG_1_dat.xz")
set <- read_datasets(path_A1)[1:2]
ranks <- pareto_rank(set)
str(ranks)
if (requireNamespace("graphics", quietly = TRUE)) {
  colors <- colorRampPalette(c("red", "yellow", "springgreen", "royalblue"))(max(ranks))
  plot(set, col = colors[ranks], type = "p", pch = 20)
}
```

r2_exact

Exact R2 indicator

Description

Computes the exact R2 indicator with respect to a given ideal/utopian reference point assuming minimization of all objectives.

Usage

```
r2_exact(x, reference, maximise = FALSE)
```

Arguments

x	matrix() data.frame() Matrix or data frame of numerical values, where each row gives the coordinates of a point.
reference	numeric() Reference (ideal) point as a vector of numerical values.
maximise	logical() Whether the objectives must be maximised instead of minimised. Either a single logical value that applies to all objectives or a vector of logical values, with one value per objective.

Details

The unary R2 indicator is a quality indicator for a set $A \subset \mathbb{R}^m$ w.r.t. an ideal or utopian reference point $\vec{r} \in \mathbb{R}^m$. It was originally proposed by Hansen and Jaszekiewicz (1998) and is defined as the expected Tchebycheff utility under a uniform distribution of weight vectors (w.l.o.g. assuming minimization):

$$R2(A) := \int_{w \in W} \min_{a \in A} \left\{ \max_{i=1, \dots, m} w_i (a_i - r_i) \right\} dw,$$

where W denotes the uniform distribution across weights:

$$W = \{w \in \mathbb{R}^m \mid w_i \geq 0, \sum_{i=1}^m w_i = 1\}.$$

The R2 indicator is to be minimized and has an optimal value of 0 when $\vec{r} \in A$.

The exact R2 indicator is strongly Pareto-compliant, i.e., compatible with Pareto-optimality:

$$\forall A, B \subset \mathbb{R}^m : A \prec B \Rightarrow R2(A) < R2(B).$$

Given an ideal or utopian reference point, which is available in most scenarios, all non-dominated solutions always contribute to the value of the exact R2 indicator. However, it is scale-dependent and care should be taken such that all objectives contribute approximately equally to the indicator, e.g., by normalizing the Pareto front to the unit hypercube.

The current implementation exclusively supports bi-objective solution sets and runs in $O(n \log n)$ following Schäpermeier and Kerschke (2025).

Value

numeric(1) A single numerical value.

References

Hansen MP, Jaszkiewicz A (1998). “Evaluating the quality of approximations to the non-dominated set.” Technical Report IMM-REP-1998-7, Institute of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark.

Schäpermeier L, Kerschke P (2025). “R2 v2: The Pareto-compliant R2 Indicator for Better Benchmarking in Bi-objective Optimization.” *Evolutionary Computation*, 1–17. doi:10.1162/evco.a.366.

Examples

```
dat <- matrix(c(5, 5, 4, 6, 2, 7, 7, 4), ncol = 2, byrow = TRUE)
r2_exact(dat, reference = c(0, 0))

# This function assumes minimisation by default. We can easily specify maximisation:
r2_exact(dat, reference = c(10, 10), maximise = TRUE)

# Merge all the sets of a dataset by removing the set number column:
extdata_path <- system.file(package="moocore", "extdata")
dat <- read_datasets(file.path(extdata_path, "example1_dat"))[, 1:2]
nrow(dat)

# Dominated points are ignored, so this:
r2_exact(dat, reference = 0)

# gives the same exact R2 value as this:
dat <- filter_dominated(dat)
nrow(dat)
r2_exact(dat, reference = 0)
```

rbind_datasets	<i>Combine datasets x and y by row taking care of making all sets unique.</i>
----------------	---

Description

Combine datasets x and y by row taking care of making all sets unique.

Usage

```
rbind_datasets(x, y)
```

Arguments

x, y	matrix data.frame() Each dataset has at least three columns, the last one is the set of each point. See also read_datasets() .
------	---

Value

matrix()|data.frame()
A dataset.

Examples

```
x <- data.frame(f1 = 5:10, f2 = 10:5, set = 1:6)
y <- data.frame(f1 = 15:20, f2 = 20:15, set = 1:6)
rbind_datasets(x,y)
```

read_datasets	<i>Read several data sets</i>
---------------	-------------------------------

Description

Reads a text file in table format and creates a matrix from it. The file may contain several sets, separated by empty lines. Lines starting by '#' are considered comments and treated as empty lines. The function adds an additional column set to indicate to which set each row belongs.

Usage

```
read_datasets(file, col_names, text)
```

Arguments

file	character() Filename that contains the data. Each row of the table appears as one line of the file. If it does not contain an <i>absolute</i> path, the file name is <i>relative</i> to the current working directory, <code>base::getwd()</code> . Tilde-expansion is performed where supported. Files compressed with xz are supported.
col_names	character() Vector of optional names for the variables. The default is to use "V" followed by the column number.
text	character() If file is not supplied and this is, then data are read from the value of text via a text connection. Notice that a literal string can be used to include (small) data sets within R code.

Value

matrix()
A numerical matrix of the data in the file. An extra column set is added to indicate to which set each row belongs.

Warning

A known limitation is that the input file must use newline characters native to the host system, otherwise they will be, possibly silently, misinterpreted. In GNU/Linux the program dos2unix may be used to fix newline characters.

Note

There are several examples of data sets in `system.file(package="moocore", "extdata")`.

Author(s)

Manuel López-Ibáñez

See Also

`utils::read.table()`

Examples

```
extdata_path <- system.file(package="moocore", "extdata")
A1 <- read_datasets(file.path(extdata_path, "ALG_1_dat.xz"))
str(A1)

read_datasets(text="1 2\n3 4\n\n5 6\n7 8\n", col_names=c("obj1", "obj2"))
```

SPEA2minstoptimeRichmond

Results of SPEA2 when minimising electrical cost and maximising the minimum idle time of pumps on Richmond water network.

Description

Results of SPEA2 when minimising electrical cost and maximising the minimum idle time of pumps on Richmond water network.

Usage

SPEA2minstoptimeRichmond

Format

A data frame as produced by `read_datasets()`. The second column measures time in seconds and corresponds to a maximisation problem.

Source

López-Ibáñez M (2009). *Operational Optimisation of Water Distribution Networks*. Ph.D. thesis, School of Engineering and the Built Environment, Edinburgh Napier University, UK. <https://lopez-ibanez.eu/publications#LopezIbanezPhD>.

Examples

```
data(SPEA2minstoptimeRichmond)
str(SPEA2minstoptimeRichmond)
```

SPEA2relativeRichmond	<i>Results of SPEA2 with relative time-controlled triggers on Richmond water network.</i>
-----------------------	---

Description

Results of SPEA2 with relative time-controlled triggers on Richmond water network.

Usage

```
SPEA2relativeRichmond
```

Format

A data frame as produced by `read_datasets()`.

Source

López-Ibáñez M (2009). *Operational Optimisation of Water Distribution Networks*. Ph.D. thesis, School of Engineering and the Built Environment, Edinburgh Napier University, UK. <https://lopez-ibanez.eu/publications#LopezIbanezPhD>.

Examples

```
data(SPEA2relativeRichmond)
str(SPEA2relativeRichmond)
```

SPEA2relativeVanzyl	<i>Results of SPEA2 with relative time-controlled triggers on Vanzyl's water network.</i>
---------------------	---

Description

Results of SPEA2 with relative time-controlled triggers on Vanzyl's water network.

Usage

```
SPEA2relativeVanzyl
```

Format

An object of class `data.frame` with 107 rows and 3 columns.

Source

López-Ibáñez M (2009). *Operational Optimisation of Water Distribution Networks*. Ph.D. thesis, School of Engineering and the Built Environment, Edinburgh Napier University, UK. <https://lopez-ibanez.eu/publications#LopezIbanezPhD>.

Examples

```
data(SPEA2relativeVanzyl)
str(SPEA2relativeVanzyl)
```

tpls50x20_1_MWT	<i>Various strategies of Two-Phase Local Search applied to the Permutation Flowshop Problem with Makespan and Weighted Tardiness objectives.</i>
-----------------	--

Description

Various strategies of Two-Phase Local Search applied to the Permutation Flowshop Problem with Makespan and Weighted Tardiness objectives.

Usage

```
tpls50x20_1_MWT
```

Format

A data frame with 1511 observations of 4 variables:

algorithm TPLS search strategy

Makespan first objective values.

WeightedTardiness second objective values.

run index of the run.

Source

Dubois-Lacoste J, López-Ibáñez M, Stützle T (2011). “Improving the Anytime Behavior of Two-Phase Local Search.” *Annals of Mathematics and Artificial Intelligence*, **61**(2), 125–154. doi:[10.1007/s1047201192350](https://doi.org/10.1007/s1047201192350).

Examples

```
data(tpls50x20_1_MWT)
str(tpls50x20_1_MWT)
```

transform_maximise	<i>Transform matrix according to maximise parameter</i>
--------------------	---

Description

Transform matrix according to maximise parameter

Usage

```
transform_maximise(x, maximise)
```

Arguments

x	matrix() data.frame() Matrix or data frame of numerical values, where each row gives the coordinates of a point.
maximise	logical() Whether the objectives must be maximised instead of minimised. Either a single logical value that applies to all objectives or a vector of logical values, with one value per objective.

Value

x transformed such that every column where maximise is TRUE is multiplied by -1.

Examples

```
x <- data.frame(f1=1:10, f2=101:110)
rownames(x) <- letters[1:10]
transform_maximise(x, maximise=c(FALSE,TRUE))
transform_maximise(x, maximise=TRUE)
x <- as.matrix(x)
transform_maximise(x, maximise=c(FALSE,TRUE))
transform_maximise(x, maximise=TRUE)
```

vorob_t	<i>Vorob'ev threshold, expectation and deviation</i>
---------	--

Description

Compute Vorob'ev threshold, expectation and deviation. Also, displaying the symmetric deviation function is possible. The symmetric deviation function is the probability for a given target in the objective space to belong to the symmetric difference between the Vorob'ev expectation and a realization of the (random) attained set.

Usage

```
vorob_t(x, sets, reference, maximise = FALSE)
```

```
vorob_dev(x, sets, reference, ve = NULL, maximise = FALSE)
```

Arguments

x	matrix() data.frame() Matrix or data frame of numerical values that represents multiple sets of points, where each row represents a point. If sets is missing, the last column of x gives the sets.
sets	integer() Vector that indicates the set of each point in x. If missing, the last column of x is used instead.
reference	numeric() Reference point as a vector of numerical values.
maximise	logical() Whether the objectives must be maximised instead of minimised. Either a single logical value that applies to all objectives or a vector of logical values, with one value per objective.
ve	matrix() Vorob'ev expectation, e.g., as returned by <code>vorob_t()</code> .

Details

Let $\mathcal{A} = \{A_1, \dots, A_n\}$ be a multi-set of n sets $A_i \subset \mathbb{R}^d$ of mutually nondominated vectors, with finite (but not necessarily equal) cardinality. If bounded by a reference point $\vec{r}$ that is strictly dominated by any point in any set, then these sets can be seen as samples from a random closed set (Molchanov 2005).

Let the β -quantile be the subset of the empirical attainment function $\mathcal{Q}_\beta = \{\vec{z} \in \mathbb{R}^d : \hat{\alpha}_{\mathcal{A}}(\vec{z}) \geq \beta\}$.

The Vorob'ev *expectation* is the β^* -quantile set $\mathcal{Q}_{\beta^*}$ such that the mean value hypervolume of the sets is equal (or as close as possible) to the hypervolume of $\mathcal{Q}_{\beta^*}$, that is, $\text{hyp}(\mathcal{Q}_\beta) \leq \mathbb{E}[\text{hyp}(\mathcal{A})] \leq \text{hyp}(\mathcal{Q}_{\beta^*})$, $\forall \beta > \beta^*$. Thus, the Vorob'ev expectation provides a definition of the notion of *mean* nondominated set.

The value $\beta^* \in [0, 1]$ is called the Vorob'ev *threshold*. Large differences from the median quantile (0.5) indicate a skewed distribution of $\mathcal{A}$.

The Vorob'ev *deviation* is the mean hypervolume of the symmetric difference between the Vorob'ev expectation and any set in $\mathcal{A}$, that is, $\mathbb{E}[\text{hyp}(\mathcal{Q}_{\beta^*} \ominus \mathcal{A})]$, where the symmetric difference is defined as $A \ominus B = (A \setminus B) \cup (B \setminus A)$. Low deviation values indicate that the sets are very similar, in terms of the location of the weakly dominated space, to the Vorob'ev expectation.

For more background, see Binois et al. (2015); Molchanov (2005); Chevalier et al. (2013).

Value

`vorob_t` returns a list with elements `threshold`, `ve`, and `avg_hyp` (average hypervolume)

`vorob_dev` returns the Vorob'ev deviation.

Author(s)

Mickael Binois

References

Binois M, Ginsbourger D, Roustant O (2015). “Quantifying uncertainty on Pareto fronts with Gaussian process conditional simulations.” *European Journal of Operational Research*, **243**(2), 386–394. doi:10.1016/j.ejor.2014.07.032.

Chevalier C, Ginsbourger D, Bect J, Molchanov I (2013). “Estimating and Quantifying Uncertainties on Level Sets Using the Vorob’ev Expectation and Deviation with Gaussian Process Models.” In Uciniski D, Atkinson AC, Patan M (eds.), *mODa 10—Advances in Model-Oriented Design and Analysis*, 35–43. Springer International Publishing, Heidelberg, Germany. doi:10.1007/9783319-002187_5.

Molchanov I (2005). *Theory of Random Sets*. Springer.

Examples

```
data(CPFs)
res <- vorob_t(CPFs, reference = c(2, 200))
res$threshold
res$avg_hyp
# Now print Vorob'ev deviation
vd <- vorob_dev(CPFs, ve = res$ve, reference = c(2, 200))
vd
```

whv_hype

Approximation of the (weighted) hypervolume by Monte-Carlo sampling (2D only)

Description

Return an estimation of the hypervolume of the space dominated by the input data following the procedure described by Auger et al. (2009). A weight distribution describing user preferences may be specified.

Usage

```
whv_hype(
  x,
  reference,
  ideal,
  maximise = FALSE,
  nsamples = 100000L,
  seed = NULL,
  dist = "uniform",
  mu = NULL
)
```

Arguments

x	<code>matrix()</code> <code>data.frame()</code> Matrix or data frame of numerical values, where each row gives the coordinates of a point.
reference	<code>numeric()</code> Reference point as a vector of numerical values.
ideal	<code>numeric()</code> Ideal point as a vector of numerical values.
maximise	<code>logical()</code> Whether the objectives must be maximised instead of minimised. Either a single logical value that applies to all objectives or a vector of logical values, with one value per objective.
nsamples	<code>integer(1)</code> Number of samples for Monte-Carlo sampling. Higher values typically produce more accurate approximations of the true hypervolume, but require more time.
seed	<code>integer(1)</code> Random seed.
dist	<code>character(1)</code> Weight distribution type. See Details.
mu	<code>numeric()</code> Parameter of the weight distribution. See Details.

Details

The current implementation only supports 2 objectives.

A weight distribution (Auger et al. 2009) can be provided via the `dist` argument. The ones currently supported are:

- "uniform" corresponds to the default hypervolume (unweighted).
- "point" describes a goal in the objective space, where the parameter `mu` gives the coordinates of the goal. The resulting weight distribution is a multivariate normal distribution centred at the goal.
- "exponential" describes an exponential distribution with rate parameter $1/\mu$, i.e., $\lambda = \frac{1}{\mu}$.

Value

A single numerical value.

References

Auger A, Bader J, Brockhoff D, Zitzler E (2009). "Articulating User Preferences in Many-Objective Problems by Sampling the Weighted Hypervolume." In Rothlauf F (ed.), *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2009*, 555–562. ACM Press, New York, NY. doi:10.1145/1569901.1569979.

See Also

[read_datasets\(\)](#), [eafdiff\(\)](#), [whv_rect\(\)](#)

Examples

```
whv_hype(matrix(2, ncol=2), reference = 4, ideal = 1, seed = 42)
whv_hype(matrix(c(3,1), ncol=2), reference = 4, ideal = 1, seed = 42)
whv_hype(matrix(2, ncol=2), reference = 4, ideal = 1, seed = 42,
  dist = "exponential", mu=0.2)
whv_hype(matrix(c(3,1), ncol=2), reference = 4, ideal = 1, seed = 42,
  dist = "exponential", mu=0.2)
whv_hype(matrix(2, ncol=2), reference = 4, ideal = 1, seed = 42,
  dist = "point", mu=c(2.9,0.9))
whv_hype(matrix(c(3,1), ncol=2), reference = 4, ideal = 1, seed = 42,
  dist = "point", mu=c(2.9,0.9))
```

whv_rect

Compute (total) weighted hypervolume given a set of rectangles

Description

Calculates the hypervolume weighted by a set of rectangles (with zero weight outside the rectangles). The function [total_whv_rect\(\)](#) calculates the total weighted hypervolume as [hypervolume\(\)](#) + scalefactor * abs(prod(reference - ideal)) * whv_rect(). The details of the computation are given by Diaz and López-Ibáñez (2021).

Usage

```
whv_rect(x, rectangles, reference, maximise = FALSE)
```

```
total_whv_rect(
  x,
  rectangles,
  reference,
  maximise = FALSE,
  ideal = NULL,
  scalefactor = 0.1
)
```

Arguments

x	matrix() data.frame() Matrix or data frame of numerical values, where each row gives the coordinates of a point.
rectangles	matrix() Weighted rectangles that will bias the computation of the hypervolume. Maybe generated by eafdiff() with rectangles=TRUE or by choose_eafdiff() .

reference	numeric() Reference point as a vector of numerical values.
maximise	logical() Whether the objectives must be maximised instead of minimised. Either a single logical value that applies to all objectives or a vector of logical values, with one value per objective.
ideal	numeric() Ideal point as a vector of numerical values. If NULL, it is calculated as minimum (or maximum if maximising that objective) of each objective in the input data.
scalefactor	numeric(1) Real value within (0, 1] that scales the overall weight of the differences. This is parameter ψ (ψ) in Diaz and López-Ibáñez (2021).

Details

TODO

Value

numeric(1) A single numerical value.

References

Diaz JE, López-Ibáñez M (2021). “Incorporating Decision-Maker’s Preferences into the Automatic Configuration of Bi-Objective Optimisation Algorithms.” *European Journal of Operational Research*, **289**(3), 1209–1222. doi:10.1016/j.ejor.2020.07.059.

See Also

[read_datasets\(\)](#), [eafdiff\(\)](#), [choose_eafdiff\(\)](#), [whv_hype\(\)](#)

Examples

```
rectangles <- as.matrix(read.table(header=FALSE, text='
1.0 3.0 2.0 Inf 1
2.0 3.5 2.5 Inf 2
2.0 3.0 3.0 3.5 3
'))
whv_rect(matrix(2, ncol=2), rectangles, reference = 6)
whv_rect(matrix(c(2, 1), ncol=2), rectangles, reference = 6)
whv_rect(matrix(c(1, 2), ncol=2), rectangles, reference = 6)
total_whv_rect(matrix(2, ncol=2), rectangles, reference = 6, ideal = c(1,1))
total_whv_rect(matrix(c(2, 1), ncol=2), rectangles, reference = 6, ideal = c(1,1))
total_whv_rect(matrix(c(1, 2), ncol=2), rectangles, reference = 6, ideal = c(1,1))
```

write_datasets	<i>Write data sets</i>
----------------	------------------------

Description

Write data sets to a file in the same format as [read_datasets\(\)](#).

Usage

```
write_datasets(x, file = "")
```

Arguments

x	<code>matrix data.frame()</code> Dataset with at least three columns, the last one is the set of each point. See also read_datasets() .
file	Either a character string naming a file or a connection open for writing. "" indicates output to the console.

Value

No return value, called for side effects

See Also

[utils::write.table\(\)](#), [read_datasets\(\)](#)

Examples

```
x <- read_datasets(text="1 2\n3 4\n\n5 6\n7 8\n", col_names=c("obj1", "obj2"))
write_datasets(x)
```

Index

- * **datasets**
 - CPFs, [7](#)
 - HybridGA, [22](#)
 - SPEA2minstoptimeRichmond, [38](#)
 - SPEA2relativeRichmond, [39](#)
 - SPEA2relativeVanzyl, [39](#)
 - tpls50x20_1_MWT, [40](#)
- * **dominance**
 - is_nondominated, [28](#)
 - pareto_rank, [32](#)
- * **eaf**
 - attsurf2df, [3](#)
 - choose_eafdiff, [4](#)
 - compute_eaf_call, [5](#)
 - compute_eafdiff_call, [6](#)
 - eaf, [7](#)
 - eaf_as_list, [10](#)
 - eafdiff, [11](#)
 - largest_eafdiff, [30](#)
 - vorob_t, [41](#)
- * **file**
 - read_datasets, [37](#)
 - write_datasets, [47](#)
- * **metrics**
 - epsilon, [12](#)
 - hv_approx, [17](#)
 - hv_contributions, [20](#)
 - hypervolume, [23](#)
 - igd, [25](#)
 - r2_exact, [34](#)
 - whv_hype, [43](#)
 - whv_rect, [45](#)
- any_dominated(is_nondominated), [28](#)
- any_dominated(), [29](#)
- as_double_matrix, [3](#)
- as_double_matrix(), [6](#)
- attsurf2df, [3](#)
- attsurf2df(), [10](#)
- avg_hausdorff_dist(igd), [25](#)
- base::getwd(), [37](#)
- base::storage.mode(), [3](#)
- choose_eafdiff, [4](#)
- choose_eafdiff(), [45](#), [46](#)
- compute_eaf_call, [5](#)
- compute_eafdiff_call, [6](#)
- CPFs, [7](#)
- eaf, [7](#)
- eaf(), [5](#), [10](#)
- eaf_as_list, [10](#)
- eaf_as_list(), [3](#), [4](#)
- eafdiff, [11](#)
- eafdiff(), [4](#), [6](#), [45](#), [46](#)
- epsilon, [12](#)
- epsilon_additive(epsilon), [12](#)
- epsilon_additive(), [26](#)
- epsilon_mult(epsilon), [12](#)
- epsilon_mult(), [26](#)
- filter_dominated(is_nondominated), [28](#)
- filter_dominated(), [29](#)
- gamma(), [18](#)
- generate_ndset, [14](#)
- hv_approx, [17](#)
- hv_contributions, [20](#)
- HybridGA, [22](#)
- hypervolume, [23](#)
- hypervolume(), [20](#), [21](#), [45](#)
- igd, [25](#)
- igd_plus(igd), [25](#)
- IGDX(igd), [25](#)
- is_nondominated, [28](#)
- is_nondominated(), [29](#), [34](#)
- largest_eafdiff, [30](#)

normalise, [31](#)
normalise(), [15](#)

pareto_rank, [32](#)
pareto_rank(), [30](#), [32](#), [33](#)

r2_exact, [34](#)
rbind_datasets, [36](#)
read_datasets, [37](#)
read_datasets(), [6](#), [9](#), [11](#), [22](#), [36](#), [38](#), [39](#),
[45–47](#)

SPEA2minstoptimeRichmond, [38](#)
SPEA2relativeRichmond, [39](#)
SPEA2relativeVanzyl, [39](#)

total_whv_rect(whv_rect), [45](#)
total_whv_rect(), [45](#)
tpls50x20_1_MWT, [40](#)
transform_maximise, [41](#)
transform_maximise(), [6](#)

utils::read.table(), [38](#)
utils::write.table(), [47](#)

vorob_dev(vorob_t), [41](#)
vorob_dev(), [7](#)
vorob_t, [41](#)
vorob_t(), [7](#), [42](#)

whv_hype, [43](#)
whv_hype(), [46](#)
whv_rect, [45](#)
whv_rect(), [45](#)
write_datasets, [47](#)