Package ‘eaf’

August 20, 2024

Type Package

Title Plots of the Empirical Attainment Function

Version 2.5.1

Description Computation and visualization of the empirical attainment function (EAF) for the analysis of random sets in multi-criterion optimization. M. López-Ibáñez, L. Paquete, and T. Stützle (2010) <doi:10.1007/978-3-642-02538-9_9>.

Depends R (>= 3.2)

Imports modeltools, graphics, grDevices, matrixStats, Rdpack

Suggests extrafont, testthat (>= 3.0.0), withr, viridisLite, spelling

License GPL (>= 2)

BugReports `https://github.com/MLopez-Ibanez/eaf/issues`

URL `https://mlopez-ibanez.github.io/eaf/`,
`https://github.com/MLopez-Ibanez/eaf`

LazyLoad true

LazyData true

Encoding UTF-8

RoxygenNote 7.3.2

SystemRequirements GNU make, Gnu Scientific Library

RdMacros Rdpack

Config/testthat/edition 3

Language en-GB

NeedsCompilation yes

Author Manuel López-Ibáñez [aut, cre]
(<`https://orcid.org/0000-0001-9974-1295`>),
Marco Chiarandini [aut],
Carlos Fonseca [aut],
Luís Paquete [aut],
Thomas Stützle [aut],
Mickaël Binois [ctb]

Maintainer Manuel López-Ibáñez <manuel.lopez-ibanez@manchester.ac.uk>

Repository CRAN

Date/Publication 2024-08-19 22:20:02 UTC

Contents

`attsurf2df` *Convert a list of attainment surfaces to a data.frame*

Description

Convert a list of attainment surfaces to a single 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 `eafplot()` (and the internal function `compute_eaf_as_list`).

Value

A data.frame with as many columns as objectives and an additional column `percentiles`.

Examples

```
data(SPEA2relativeRichmond)
attsurfs <- eafplot (SPEA2relativeRichmond, percentiles = c(0,50,100),
                     xlab = expression(C[E]), ylab = "Total switches",
                     lty=0, pch=21, xlim = c(90, 140), ylim = c(0, 25))
attsurfs <- attsurf2df(attsurfs)
text(attsurfs[,1:2], labels = attsurfs[,3], adj = c(1.5,1.5))
```

`choose_eafdiffplot` *Interactively choose according to empirical attainment function differences*

Description

Creates the same plot as `eafdiffplot()` but waits for the user to click in one of the sides. Then it returns the rectangles the give the differences in favour of the chosen side. These rectangles may be used for interactive decision-making as shown in Diaz and López-Ibáñez (2021). The function `choose_eafdiff()` may be used in a non-interactive context.

Usage

```
choose_eafdiffplot(
  data.left,
  data.right,
  intervals = 5,
  maximise = c(FALSE, FALSE),
  title.left = deparse(substitute(data.left)),
  title.right = deparse(substitute(data.right)),
  ...
)

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

Arguments

`data.left`, `data.right`	Data frames corresponding to the input data of left and right sides, respectively. Each data frame has at least three columns, the third one being the set of each point. See also `read_datasets()`.
`intervals`	(`integer(1)`|`character()`) The absolute range of the differences $[0, 1]$ is partitioned into the number of intervals provided. If an integer is provided, then labels for each interval are computed automatically. If a character vector is provided, its length is taken as the number of intervals.
`maximise`	(`logical()` | `logical(1)`) 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.
`title.left`, `title.right`	Title for left and right panels, respectively.
`...`	Other graphical parameters are passed down to `eafdiffplot()`.
`x`	(`matrix()`) Matrix of rectangles representing EAF differences (returned by `eafdiff()` with `rectangles=TRUE`).
`left`	(`logical(1)`) With `left=TRUE` 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.

References

Juan Esteban Diaz, Manuel López-Ibáñez (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()`, `eafdiffplot()`, `whv_rect()`

Examples

```
extdata_dir <- system.file(package="eaf", "extdata")
A1 <- read_datasets(file.path(extdata_dir, "wrots_l100w10_dat"))
A2 <- read_datasets(file.path(extdata_dir, "wrots_l10w100_dat"))
if (interactive()) {
  rectangles <- choose_eafdiffplot(A1, A2, intervals = 5)
} else { # 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)
boxplot(list(A1=hv_A1, A2=hv_A2), main = "Hypervolume")

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)
boxplot(list(A1=whv_A1, A2=whv_A2), main = "Weighted hypervolume")
```

CPFs — *Conditional Pareto fronts obtained from Gaussian processes simulations.*

Description

The data has the only goal of providing an example of use of `vorobT()` and `vorobDev()`. 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

M Binois, D Ginsbourger, O Roustant (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.

Examples

```
data(CPFs)

res <- vorobT(CPFs, reference = c(2, 200))
eafplot(CPFs[,1:2], sets = CPFs[,3], percentiles = c(0, 20, 40, 60, 80, 100),
       col = gray(seq(0.8, 0.1, length.out = 6)^2), type = "area",
       legend.pos = "bottomleft", extra.points = res$VE, extra.col = "cyan")
```

`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 = c(FALSE, FALSE), rectangles = FALSE)
```

Arguments

`x, y`	Data frames corresponding to the input data of left and right sides, respectively. Each data frame has at least three columns, the third one being the set of each point. See also `read_datasets()`.
`intervals`	(`integer(1)`) The absolute range of the differences $[0, 1]$ is partitioned into the number of intervals provided.
`maximise`	(`logical()` | `logical(1)`) 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.
`rectangles`	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 and the last column. 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()`, `eafdiffplot()`

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)
print(d)

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

`eafdiffplot` *Plot empirical attainment function differences*

Description

Plot the differences between the empirical attainment functions (EAFs) of two data sets as a two-panel plot, where the left side shows the values of the left EAF minus the right EAF and the right side shows the differences in the other direction.

Usage

```
eafdiffplot(
  data.left,
  data.right,
  col = c("#FFFFFF", "#808080", "#000000"),
  intervals = 5,
  percentiles = c(50),
  full.eaf = FALSE,
  type = "area",
  legend.pos = if (full.eaf) "bottomleft" else "topright",
  title.left = deparse(substitute(data.left)),
  title.right = deparse(substitute(data.right)),
  xlim = NULL,
  ylim = NULL,
  cex = par("cex"),
  cex.lab = par("cex.lab"),
  cex.axis = par("cex.axis"),
  maximise = c(FALSE, FALSE),
  grand.lines = TRUE,
  sci.notation = FALSE,
  left.panel.last = NULL,
  right.panel.last = NULL,
  ...
)
```

Arguments

`data.left, data.right`	Data frames corresponding to the input data of left and right sides, respectively. Each data frame has at least three columns, the third one being the set of each point. See also `read_datasets()`.
`col`	A character vector of three colors for the magnitude of the differences of 0, 0.5, and 1. Intermediate colors are computed automatically given the value of `intervals`. Alternatively, a function such as `viridisLite::viridis()` that generates a colormap given an integer argument.
`intervals`	(`integer(1)`|`character()`) The absolute range of the differences $[0, 1]$ is partitioned into the number of intervals provided. If an integer is provided, then labels for each interval are computed automatically. If a character vector is provided, its length is taken as the number of intervals.
`percentiles`	The percentiles of the EAF of each side that will be plotted as attainment surfaces. `NA` does not plot any. See `eafplot()`.
`full.eaf`	Whether to plot the EAF of each side instead of the differences between the EAFs.
`type`	Whether the EAF differences are plotted as points (‘`points`’) or whether to color the areas that have at least a certain value (‘`area`’).
`legend.pos`	The position of the legend. See `legend()`. A value of `"none"` hides the legend.
`title.left, title.right`	Title for left and right panels, respectively.
`xlim, ylim, cex, cex.lab, cex.axis`	Graphical parameters, see `plot.default()`.
`maximise`	(`logical()` | `logical(1)`) 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.
`grand.lines`	Whether to plot the grand-best and grand-worst attainment surfaces.

`sci.notation`	Generate prettier labels
`left.panel.last`, `right.panel.last`	An expression to be evaluated after plotting has taken place on each panel (left or right). This can be useful for adding points or text to either panel. Note that this works by lazy evaluation: passing this argument from other `plot` methods may well not work since it may be evaluated too early.
`...`	Other graphical parameters are passed down to `plot.default()`.

Details

This function calculates the differences between the EAFs of two data sets, and plots on the left the differences in favour of the left data set, and on the right the differences in favour of the right data set. By default, it also plots the grand best and worst attainment surfaces, that is, the 0%- and 100%-attainment surfaces over all data. These two surfaces delimit the area where differences may exist. In addition, it also plots the 50%-attainment surface of each data set.

With `type = "point"`, only the points where there is a change in the value of the EAF difference are plotted. This means that for areas where the EAF differences stays constant, the region will appear in white even if the value of the differences in that region is large. This explains "white holes" surrounded by black points.

With `type = "area"`, the area where the EAF differences has a certain value is plotted. The idea for the algorithm to compute the areas was provided by Carlos M. Fonseca. The implementation uses R polygons, which some PDF viewers may have trouble rendering correctly (See `https://cran.r-project.org/doc/FAQ/R-FAQ.html#Why-are-there-unwanted-borders`). Plots (should) look correct when printed.

Large differences that appear when using `type = "point"` may seem to disappear when using `type = "area"`. The explanation is the points size is independent of the axes range, therefore, the plotted points may seem to cover a much larger area than the actual number of points. On the other hand, the areas size is plotted with respect to the objective space, without any extra borders. If the range of an area becomes smaller than one-pixel, it won't be visible. As a consequence, zooming in or out certain regions of the plots does not change the apparent size of the points, whereas it affects considerably the apparent size of the areas.

Value

Returns a representation of the EAF differences (invisibly).

See Also

`eafplot()` `pdf_crop()`

Examples

```
## NOTE: The plots in the website look squashed because of how pkgdown
## generates them. They should look fine when you generate them yourself.
extdata_dir <- system.file(package="eaf", "extdata")
A1 <- read_datasets(file.path(extdata_dir, "ALG_1_dat.xz"))
A2 <- read_datasets(file.path(extdata_dir, "ALG_2_dat.xz"))
# These take time
```

```
eafdiffplot(A1, A2, full.eaf = TRUE)
if (requireNamespace("viridisLite", quietly=TRUE)) {
  viridis_r <- function(n) viridisLite::viridis(n, direction=-1)
  eafdiffplot(A1, A2, type = "area", col = viridis_r)
} else {
  eafdiffplot(A1, A2, type = "area")
}
A1 <- read_datasets(file.path(extdata_dir, "wrots_l100w10_dat"))
A2 <- read_datasets(file.path(extdata_dir, "wrots_l10w100_dat"))
eafdiffplot(A1, A2, type = "point", sci.notation = TRUE, cex.axis=0.6)

# A more complex example
DIFF <- eafdiffplot(A1, A2, col = c("white", "blue", "red"), intervals = 5,
                    type = "point",
                    title.left=expression("W-RoTS," ~ lambda==100 * "," ~ omega==10),
                    title.right=expression("W-RoTS," ~ lambda==10 * "," ~ omega==100),
                    right.panel.last={
                      abline(a = 0, b = 1, col = "red", lty = "dashed")})
DIFF$right[,3] <- -DIFF$right[,3]

## Save the values to a file.
# write.table(rbind(DIFF$left,DIFF$right),
#             file = "wrots_l100w10_dat-wrots_l10w100_dat-diff.txt",
#             quote = FALSE, row.names = FALSE, col.names = FALSE)
```

eafplot *Plot the Empirical Attainment Function for two objectives*

Description

Computes and plots the Empirical Attainment Function, either as attainment surfaces for certain percentiles or as points.

Usage

```
eafplot(x, ...)

## Default S3 method:
eafplot(
  x,
  sets = NULL,
  groups = NULL,
  percentiles = c(0, 50, 100),
  attsurfs = NULL,
  xlab = NULL,
  ylab = NULL,
  xlim = NULL,
  ylim = NULL,
  log = "",
  type = "point",
  col = NULL,
  lty = c("dashed", "solid", "solid", "solid", "dashed"),
  lwd = 1.75,
  pch = NA,
  cex.pch = par("cex"),
  las = par("las"),
  legend.pos = "topright",
  legend.txt = NULL,
  extra.points = NULL,
  extra.legend = NULL,
  extra.pch = 4:25,
  extra.lwd = 0.5,
  extra.lty = NA,
  extra.col = "black",
  maximise = c(FALSE, FALSE),
  xaxis.side = "below",
  yaxis.side = "left",
  axes = TRUE,
  sci.notation = FALSE,
  ...
)

## S3 method for class 'formula'
eafplot(formula, data, groups = NULL, subset = NULL, ...)

## S3 method for class 'list'
eafplot(x, ...)
```

Arguments

`x`	Either a matrix of data values, or a data frame, or a list of data frames of exactly three columns.
`...`	Other graphical parameters to `plot.default()`.
`sets`	(numeric) Vector indicating which set each point belongs to.
`groups`	This may be used to plot profiles of different algorithms on the same plot.
`percentiles`	(`numeric()`) Vector indicating which percentile should be plot. The default is to plot only the median attainment curve.
`attsurfs`	TODO
`xlab`, `ylab`, `xlim`, `ylim`, `log`, `col`, `lty`, `lwd`, `pch`, `cex.pch`, `las`	Graphical parameters, see `plot.default()`.
`type`	(`character(1)`) string giving the type of plot desired. The following values are possible, '`points`' and '`area`'.
`legend.pos`	the position of the legend, see `legend()`. A value of `"none"` hides the legend.

`legend.txt`	a character or expression vector to appear in the legend. If `NULL`, appropriate labels will be generated.
`extra.points`	A list of matrices or data.frames with two-columns. Each element of the list defines a set of points, or lines if one of the columns is `NA`.
`extra.legend`	A character vector providing labels for the groups of points.
`extra.pch`, `extra.lwd`, `extra.lty`, `extra.col`	Control the graphical aspect of the points. See `points()` and `lines()`.
`maximise`	(`logical()` | `logical(1)`) 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.
`xaxis.side`	On which side that x-axis is drawn. Valid values are `"below"` and `"above"`. See `axis()`.
`yaxis.side`	On which side that y-axis is drawn. Valid values are `"left"` and `"right"`. See `axis()`.
`axes`	A logical value indicating whether both axes should be drawn on the plot.
`sci.notation`	Generate prettier labels
`formula`	A formula of the type: `time + cost ~ run | instance` will draw `time` on the x-axis and `cost` on the y-axis. If `instance` is present the plot is conditional to the instances.
`data`	Dataframe containing the fields mentioned in the formula and in groups.
`subset`	(`integer()` | `NULL`) A vector indicating which rows of the data should be used. If left to default `NULL` all data in the data frame are used.

Details

This function can be used to plot random sets of points like those obtained by different runs of biobjective stochastic optimisation algorithms. An EAF curve represents the boundary separating points that are known to be attainable (that is, dominated in Pareto sense) in at least a fraction (quantile) of the runs from those that are not. The median EAF represents the curve where the fraction of attainable points is 50%. In single objective optimisation the function can be used to plot the profile of solution quality over time of a collection of runs of a stochastic optimizer.

Value

Return (invisibly) the attainment surfaces computed.

Methods (by class)

- `eafplot(default)`: Main function
- `eafplot(formula)`: Formula interface
- `eafplot(list)`: List interface for lists of data.frames or matrices

See Also

eafdiffplot() pdf_crop()

Examples

```
data(gcp2x2)
tabucol <- subset(gcp2x2, alg != "TSinN1")
tabucol$alg <- tabucol$alg[drop=TRUE]
eafplot(time + best ~ run, data = tabucol, subset = tabucol$inst=="DSJC500.5")

# These take time
eafplot(time + best ~ run | inst, groups=alg, data=gcp2x2)
eafplot(time + best ~ run | inst, groups=alg, data=gcp2x2,
percentiles=c(0,50,100), cex.axis = 0.8, lty = c(2,1,2), lwd = c(2,2,2),
     col = c("black","blue","grey50"))

extdata_path <- system.file(package = "eaf", "extdata")
A1 <- read_datasets(file.path(extdata_path, "ALG_1_dat.xz"))
A2 <- read_datasets(file.path(extdata_path, "ALG_2_dat.xz"))
eafplot(A1, percentiles = 50, sci.notation = TRUE, cex.axis=0.6)
# The attainment surfaces are returned invisibly.
attsurfs <- eafplot(list(A1 = A1, A2 = A2), percentiles = 50)
str(attsurfs)

## Save as a PDF file.
# dev.copy2pdf(file = "eaf.pdf", onefile = TRUE, width = 5, height = 4)


## Using extra.points

data(HybridGA)
data(SPEA2relativeVanzyl)
eafplot(SPEA2relativeVanzyl, percentiles = c(25, 50, 75),
        xlab = expression(C[E]), ylab = "Total switches", xlim = c(320, 400),
        extra.points = HybridGA$vanzyl, extra.legend = "Hybrid GA")

data(SPEA2relativeRichmond)
eafplot (SPEA2relativeRichmond, percentiles = c(25, 50, 75),
         xlab = expression(C[E]), ylab = "Total switches",
         xlim = c(90, 140), ylim = c(0, 25),
         extra.points = HybridGA$richmond, extra.lty = "dashed",
         extra.legend = "Hybrid GA")

eafplot (SPEA2relativeRichmond, percentiles = c(25, 50, 75),
         xlab = expression(C[E]), ylab = "Total switches",
         xlim = c(90, 140), ylim = c(0, 25), type = "area",
         extra.points = HybridGA$richmond, extra.lty = "dashed",
         extra.legend = "Hybrid GA", legend.pos = "bottomright")

data(SPEA2minstoptimeRichmond)
SPEA2minstoptimeRichmond[,2] <- SPEA2minstoptimeRichmond[,2] / 60
eafplot (SPEA2minstoptimeRichmond, xlab = expression(C[E]),
         ylab = "Minimum idle time (minutes)", maximise = c(FALSE, TRUE),
         las = 1, log = "y", main = "SPEA2 (Richmond)",
         legend.pos = "bottomright")
```

eafs — *Exact computation of the EAF in 2D or 3D*

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

```
eafs(points, sets, groups = NULL, percentiles = NULL)
```

Arguments

`points`	Either a matrix or a data frame of numerical values, where each row gives the coordinates of a point.
`sets`	A vector indicating which set each point belongs to.
`groups`	Indicates that the EAF must be computed separately for data belonging to different groups.
`percentiles`	(`numeric()`) Vector indicating which percentiles are computed. `NULL` computes all.

Value

A data frame (`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="eaf","extdata")`. The current implementation only supports two and three dimensional points.

Author(s)

Manuel López-Ibáñez

References

Viviane Grunert da Fonseca, Carlos M. Fonseca, Andreia O. Hall (2001). “Inferential Performance Assessment of Stochastic Optimisers and the Attainment Function.” In Eckart Zitzler, Kalyanmoy Deb, Lothar Thiele, Carlos A. Coello Coello, David Corne (eds.), *Evolutionary Multi-criterion Optimization, EMO 2001*, volume 1993 of *Lecture Notes in Computer Science*, 213–225. Springer, Heidelberg, Germany. doi: 10.1007/3540447199_15.

Carlos M. Fonseca, Andreia P. Guerreiro, Manuel López-Ibáñez, Luís Paquete (2011). “On the Computation of the Empirical Attainment Function.” In R H C Takahashi, others (eds.), *Evolutionary Multi-criterion Optimization, EMO 2011*, volume 6576 of *Lecture Notes in Computer Science*, 106–120. Springer, Heidelberg . doi: 10.1007/9783642198939_8.

See Also

`read_datasets()`

Examples

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

x <- read_datasets(file.path(extdata_path, "example1_dat"))
# Compute full EAF
str(eafs(x[,1:2], x[,3]))

# Compute only best, median and worst
str(eafs(x[,1:2], 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 <- eafs(x[,1:3], sets = x[,4], groups = x[,5], percentiles = 50)
str(z)
# library(plotly)
# plot_ly(z, x = ~X1, y = ~X2, z = ~X3, color = ~groups,
#         colors = c('#BF382A', '#0C4B8E')) %>% add_markers()
```

`epsilon` *Epsilon metric*

Description

Computes the epsilon metric, either additive or multiplicative.

Usage

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

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

Arguments

`data`	(`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()` | `logical(1)`) 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 with respect to a reference set R is defined as

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

where a and b are objective vectors and, in the case of minimization of objective i, $epsilon(a_i, b_i)$ is computed as a_i/b_i for the multiplicative variant (respectively, $a_i - b_i$ for the additive variant), whereas in the case of maximization of objective i, $epsilon(a_i, b_i) = b_i/a_i$ for the multiplicative variant (respectively, $b_i - a_i$ 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 computation of the multiplicative version for negative values doesn't make sense.

Computation of the epsilon indicator requires $O(n \cdot |A| \cdot |R|)$, where n is the number of objectives (dimension of vectors).

Value

A single numerical value.

Author(s)

Manuel López-Ibáñez

References

Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M. Fonseca, Viviane Grunert da Fonseca (2003). "Performance Assessment of Multiobjective Optimizers: an Analysis and Review." *IEEE Transactions on Evolutionary Computation*, **7**(2), 117–132.

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)

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

# A more realistic example
extdata_path <- system.file(package="eaf","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)
```

`gcp2x2` *Metaheuristics for solving the Graph Vertex Coloring Problem*

Description

Two metaheuristic algorithms, TabuCol (Hertz et al., 1987) and simulated annealing (Johnson et al. 1991), to find a good approximation of the chromatic number of two random graphs. The data here has the only goal of providing an example of use of eafplot for comparing algorithm performance with respect to both time and quality when modelled as two objectives in trade off.

Usage

```
gcp2x2
```

Format

A data frame with 3133 observations on the following 6 variables.

`alg` a factor with levels `SAKempeFI` and `TSinN1` `inst` a factor with levels `DSJC500.5` and `DSJC500.9`. Instances are taken from the DIMACS repository.

`run` a numeric vector indicating the run to which the observation belong.

`best` a numeric vector indicating the best solution in number of colors found in the corresponding run up to that time.

`time` a numeric vector indicating the time since the beginning of the run for each observation. A rescaling is applied.

`titer` a numeric vector indicating iteration number corresponding to the observations.

Details

Each algorithm was run 10 times per graph registering the time and iteration number at which a new best solution was found. A time limit corresponding to 500*10^5 total iterations of TabuCol was imposed. The time was then normalized on a scale from 0 to 1 to make it instance independent.

Source

Marco Chiarandini (2005). *Stochastic Local Search Methods for Highly Constrained Combinatorial Optimisation Problems*. Ph.D. thesis, FB Informatik, TU Darmstadt, Germany. (page 138)

References

A. Hertz and D. de Werra. Using Tabu Search Techniques for Graph Coloring. Computing, 1987, 39(4), 345-351.

David S. Johnson, Cecilia R. Aragon, Lyle A. McGeoch, Catherine Schevon (1991). "Optimization by Simulated Annealing: An Experimental Evaluation: Part II, Graph Coloring and Number Partitioning." *Operations Research*, **39**(3), 378–406.

Examples

```
data(gcp2x2)
```

`hv_contributions` *Hypervolume contribution of a set of points*

Description

Computes the hypervolume contribution of each point given a set of points with respect to a given reference point assuming minimization of all objectives. Dominated points have zero contribution. Duplicated points have zero contribution even if not dominated, because removing one of them does not change the hypervolume dominated by the remaining set.

Usage

```
hv_contributions(data, reference, maximise = FALSE)
```

Arguments

`data`	(`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()` | `logical(1)`) 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

(numeric) A numerical vector

Author(s)

Manuel López-Ibáñez

References

Carlos M. Fonseca, Luís Paquete, Manuel López-Ibáñez (2006). "An improved dimension-sweep algorithm for the hypervolume indicator." In *Proceedings of the 2006 Congress on Evolutionary Computation (CEC 2006)*, 1157–1163. IEEE Press, Piscataway, NJ. doi: 10.1109/CEC.2006.1688440.

Nicola Beume, Carlos M. Fonseca, Manuel López-Ibáñez, Luís Paquete, Jan Vahrenhold (2009). "On the complexity of computing the hypervolume indicator." *IEEE Transactions on Evolutionary Computation*, **13**(5), 1075–1082. doi: 10.1109/TEVC.2009.2015575.

See Also

`hypervolume`

Examples

```
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

The data has the only goal of providing an example of use of eafplot.

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

Manuel López-Ibáñez (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

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

Usage

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

Arguments

`data`	(`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()` | `logical(1)`) 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 algorithm has $O(n^{d-2} \log n)$ time and linear space complexity in the worst-case, but experimental results show that the pruning techniques used may reduce the time complexity even further.

Value

A single numerical value.

Author(s)

Manuel López-Ibáñez

References

Carlos M. Fonseca, Luís Paquete, Manuel López-Ibáñez (2006). "An improved dimension-sweep algorithm for the hypervolume indicator." In *Proceedings of the 2006 Congress on Evolutionary Computation (CEC 2006)*, 1157–1163. IEEE Press, Piscataway, NJ. doi: 10.1109/CEC.2006.1688440.

Nicola Beume, Carlos M. Fonseca, Manuel López-Ibáñez, Luís Paquete, Jan Vahrenhold (2009). "On the complexity of computing the hypervolume indicator." *IEEE Transactions on Evolutionary Computation*, **13**(5), 1075–1082. doi: 10.1109/TEVC.2009.2015575.

Examples

```
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 — *Inverted Generational Distance (IGD and IGD+) and Averaged Hausdorff Distance*

Description

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

Usage

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

igd_plus(data, reference, maximise = FALSE)

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

Arguments

`data`	(`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()` | `logical(1)`) 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 distanced (IGD+) was proposed by Ishibuchi et al. (2015) to ensure that IGD+ is weakly Pareto compliant, similarly to `epsilon_additive()` or `epsilon_mult()`. It modifies the distance measure as:

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

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. IGD seems to have been mentioned first by Coello Coello and Reyes-Sierra (2004), however, some people also used the name D-metric for the same concept with $p = 1$ and later papers have often used IGD/GD with $p = 1$. 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, as it often contradicts Pareto optimality (it is not weakly Pareto-compliant). We recommend IGD+ instead of IGD, since the latter contradicts Pareto optimality in some cases (see examples below) whereas IGD+ is weakly Pareto-compliant, but we implement IGD here because it is still popular due to historical reasons.

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

Leonardo C. T. Bezerra, Manuel López-Ibáñez, Thomas Stützle (2017). "An Empirical Assessment of the Properties of Inverted Generational Distance Indicators on Multi- and Many-objective Optimization." In Heike Trautmann, Günter Rudolph, Kathrin Klamroth, Oliver Schütze, Margaret M. Wiecek, Yaochu Jin, Christian Grimme (eds.), *Evolutionary Multi-criterion Optimization, EMO 2017*, Lecture Notes in Computer Science, 31–45. Springer International Publishing, Cham, Switzerland. doi: 10.1007/9783319541570_3.

Carlos A. Coello Coello, Margarita Reyes-Sierra (2004). "A Study of the Parallelization of a Coevolutionary Multi-objective Evolutionary Algorithm." In Raúl Monroy, Gustavo Arroyo-Figueroa, Luis Enrique Sucar, Humberto Sossa (eds.), *Proceedings of MICAI*, volume 2972 of *Lecture Notes in Artificial Intelligence*, 688–697. Springer, Heidelberg, Germany.

Hisao Ishibuchi, Hiroyuki Masuda, Yuki Tanigaki, Yusuke Nojima (2015). "Modified Distance Calculation in Generational Distance and Inverted Generational Distance." In António Gaspar-Cunha, Carlos Henggeler Antunes, Carlos A. Coello Coello (eds.), *Evolutionary Multi-criterion Optimization, EMO 2015 Part I*, volume 9018 of *Lecture Notes in Computer Science*, 110–125. Springer, Heidelberg, Germany.

Oliver Schütze, X Esquivel, A Lara, Carlos A. Coello Coello (2012). "Using the Averaged Hausdorff Distance as a Performance Measure in Evolutionary Multiobjective Optimization." *IEEE Transactions on Evolutionary Computation*, **16**(4), 504–522.

David A. Van Veldhuizen, Gary B. Lamont (1998). "Evolutionary Computation and Convergence to a Pareto Front." In John R. Koza (ed.), *Late Breaking Papers at the Genetic Programming 1998 Conference*, 221–228.

A Zhou, Qingfu Zhang, Yaochu Jin (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)
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(A)=", 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="eaf","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 Haussdorff distance
avg_hausdorff_dist(A1, ref)
avg_hausdorff_dist(A2, ref)
```

is_nondominated — *Identify, remove and rank dominated points according to Pareto optimality*

Description

Identify nondominated points with `is_nondominated` and remove dominated ones with `filter_dominated`.

`pareto_rank()` ranks points according to Pareto-optimality, which is also called nondominated sorting (Deb et al. 2002).

Usage

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

filter_dominated(data, maximise = FALSE, keep_weakly = FALSE)

pareto_rank(data, maximise = FALSE)
```

Arguments

`data`	(`matrix` | `data.frame`) Matrix or data frame of numerical values, where each row gives the coordinates of a point.
`maximise`	(`logical()` | `logical(1)`) 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`	If `FALSE`, return `FALSE` for any duplicates of nondominated points.

Details

pareto_rank() is meant to be used like rank(), but it assigns ranks according to Pareto dominance. Duplicated points are kept on the same front. When ncol(data) == 2, the code uses the $O(n \log n)$ algorithm by Jensen (2003).

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.

pareto_rank() returns an integer vector of the same length as the number of rows of data, where each value gives the rank of each point.

Author(s)

Manuel López-Ibáñez

References

Kalyanmoy Deb, A Pratap, S Agarwal, T Meyarivan (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.

M T Jensen (2003). "Reducing the run-time complexity of multiobjective EAs: The NSGA-II and other algorithms." *IEEE Transactions on Evolutionary Computation*, **7**(5), 503–515.

Examples

```
path_A1 <- file.path(system.file(package="eaf"),"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")

plot(set, col = "blue", type = "p", pch = 20)
ndset <- filter_dominated(set)
points(ndset[order(ndset[,1]),], col = "red", pch = 21)

ranks <- pareto_rank(set)
colors <- colorRampPalette(c("red","yellow","springgreen","royalblue"))(max(ranks))
plot(set, col = colors[ranks], type = "p", pch = 20)
```

`largest_eafdiff` *Identify largest EAF differences*

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(data, maximise = FALSE, intervals = 5, reference, ideal = NULL)
```

Arguments

`data`	(`list(1)`) A list of matrices with at least 3 columns
`maximise`	(`logical()` | `logical(1)`) 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`	(`integer(1)`) 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 (resp. maximum if maximising that objective) of each objective in `data`.

Value

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

References

Juan Esteban Diaz, Manuel López-Ibáñez (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.
files <- c("wrots_l100w10_dat","wrots_l10w100_dat")
data <- lapply(files, function(x)
               read_datasets(file.path(system.file(package="eaf"),
                             "extdata", x)))
nadir <- apply(do.call(rbind, data)[,1:2], 2, max)
x <- largest_eafdiff(data, reference = nadir)
```

```
str(x)
```

normalise *Normalise points*

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

Arguments

`data`	(`matrix` | `data.frame`) Matrix or data frame of numerical values, where each row gives the coordinates of a point.
`to_range`	Normalise values to this range. If the objective is maximised, it is normalised to `c(to_range[1], to_range[0])` instead.
`lower`, `upper`	Bounds on the values. If NA, the maximum and minimum values of each coordinate are used.
`maximise`	(`logical()` | `logical(1)`) 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

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)))
```

pdf_crop — *Remove whitespace margins from a PDF file (and maybe embed fonts)*

Description

Remove whitespace margins using `https://ctan.org/pkg/pdfcrop` and optionally embed fonts using `grDevices::embedFonts()`. You may install `pdfcrop` using TinyTeX (`https://cran.r-project.org/package=tinytex`) with `tinytex::tlmgr_install('pdfcrop')`.

Usage

```
pdf_crop(
  filename,
  mustWork = FALSE,
  pdfcrop = Sys.which("pdfcrop"),
  embed_fonts = FALSE
)
```

Arguments

`filename`	Filename of a PDF file to crop. The file will be overwritten.
`mustWork`	If `TRUE`, then give an error if the file cannot be cropped.
`pdfcrop`	Path to the `pdfcrop` utility.
`embed_fonts`	(`logical(1)`) If `TRUE`, use `grDevices::embedFonts()` to embed fonts.

Details

You may also wish to consider `extrafont::embed_fonts()` (`https://cran.r-project.org/package=extrafont`).

```
library(extrafont)
# If you need to specify the path to Ghostscript (probably not needed in Linux)
Sys.setenv(R_GSCMD = "C:/Program Files/gs/gs9.56.1/bin/gswin64c.exe")
embed_fonts("original.pdf", outfile = "new.pdf")
```

As an alternative, saving the PDF with `grDevices::cairo_pdf()` should already embed the fonts.

Value

Nothing

See Also

`grDevices::embedFonts()` `extrafont::embed_fonts()` `grDevices::cairo_pdf()`

Examples

```
## Not run:
extdata_path <- system.file(package = "eaf", "extdata")
A1 <- read_datasets(file.path(extdata_path, "wrots_l100w10_dat"))
A2 <- read_datasets(file.path(extdata_path, "wrots_l10w100_dat"))
pdf(file = "eaf.pdf", onefile = TRUE, width = 5, height = 4)
eafplot(list(A1 = A1, A2 = A2), percentiles = 50, sci.notation=TRUE)
dev.off()
pdf_crop("eaf.pdf")

## End(Not run)
```

`read_datasets` *Read several data sets*

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)

read.data.sets(file, col.names)
```

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 *absolute* path, the file name is *relative* to the current working directory, `getwd()`. Tilde-expansion is performed where supported. Files compressed with `xz` are supported.
`col_names`, `col.names`	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()`) containing a representation 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="eaf","extdata")`.

`read.data.sets()` is a deprecated alias. It will be removed in the next major release.

Author(s)

Manuel López-Ibáñez

See Also

`read.table`, `eafplot()`, `eafdiffplot()`

Examples

```
extdata_path <- system.file(package="eaf","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

The data has the only goal of providing an example of use of eafplot.

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

Manuel López-Ibáñez (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)
data(SPEA2minstoptimeRichmond)
SPEA2minstoptimeRichmond[,2] <- SPEA2minstoptimeRichmond[,2] / 60
eafplot (SPEA2minstoptimeRichmond, xlab = expression(C[E]),
         ylab = "Minimum idle time (minutes)", maximise = c(FALSE, TRUE),
         las = 1, log = "y", legend.pos = "bottomright")
```

`SPEA2relativeRichmond` *Results of SPEA2 with relative time-controlled triggers on Richmond water network.*

Description

The data has the only goal of providing an example of use of eafplot.

Usage

```
SPEA2relativeRichmond
```

Format

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

Source

Manuel López-Ibáñez (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)
data(SPEA2relativeRichmond)
eafplot (SPEA2relativeRichmond, percentiles = c(25, 50, 75),
        xlab = expression(C[E]), ylab = "Total switches",
        xlim = c(90, 140), ylim = c(0, 25),
        extra.points = HybridGA$richmond, extra.lty = "dashed",
        extra.legend = "Hybrid GA")
```

SPEA2relativeVanzyl — *Results of SPEA2 with relative time-controlled triggers on Vanzyl's water network.*

Description

The data has the only goal of providing an example of use of eafplot.

Usage

```
SPEA2relativeVanzyl
```

Format

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

Source

Manuel López-Ibáñez (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)
data(SPEA2relativeVanzyl)
eafplot(SPEA2relativeVanzyl, percentiles = c(25, 50, 75),
        xlab = expression(C[E]), ylab = "Total switches", xlim = c(320, 400),
        extra.points = HybridGA$vanzyl, extra.legend = "Hybrid GA")
```

vorobT — *Vorob'ev computations*

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

```
vorobT(x, reference)

vorobDev(x, VE, reference)

symDifPlot(
  x,
  VE,
  threshold,
  nlevels = 11,
  ve.col = "blue",
  xlim = NULL,
  ylim = NULL,
  legend.pos = "topright",
  main = "Symmetric deviation function",
  col.fun = function(n) gray(seq(0, 0.9, length.out = n)^2)
)
```

Arguments

`x`	Either a matrix of data values, or a data frame, or a list of data frames of exactly three columns. The third column gives the set (run, sample, ...) identifier.
`reference`	(`numeric()`) Reference point as a vector of numerical values.
`VE, threshold`	Vorob'ev expectation and threshold, e.g., as returned by `vorobT()`.
`nlevels`	number of levels in which is divided the range of the symmetric deviation.
`ve.col`	plotting parameters for the Vorob'ev expectation.
`xlim, ylim, main`	Graphical parameters, see `plot.default()`.
`legend.pos`	the position of the legend, see `legend()`. A value of `"none"` hides the legend.
`col.fun`	function that creates a vector of `n` colors, see `heat.colors()`.

Value

`vorobT` returns a list with elements `threshold`, `VE`, and `avg_hyp` (average hypervolume)

`vorobDev` returns the Vorob'ev deviation.

Author(s)

Mickael Binois

References

M Binois, D Ginsbourger, O Roustant (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.

C. Chevalier (2013), Fast uncertainty reduction strategies relying on Gaussian process models, University of Bern, PhD thesis.

I. Molchanov (2005), Theory of random sets, Springer.

Examples

```
data(CPFs)
res <- vorobT(CPFs, reference = c(2, 200))
print(res$threshold)

## Display Vorob'ev expectation and attainment function
# First style
eafplot(CPFs[,1:2], sets = CPFs[,3], percentiles = c(0, 25, 50, 75, 100, res$threshold),
        main = substitute(paste("Empirical attainment function, ",beta,"* = ", a, "%"),
                          list(a = formatC(res$threshold, digits = 2, format = "f"))))

# Second style
eafplot(CPFs[,1:2], sets = CPFs[,3], percentiles = c(0, 20, 40, 60, 80, 100),
        col = gray(seq(0.8, 0.1, length.out = 6)^0.5), type = "area",
        legend.pos = "bottomleft", extra.points = res$VE, extra.col = "cyan",
        extra.legend = "VE", extra.lty = "solid", extra.pch = NA, extra.lwd = 2,
        main = substitute(paste("Empirical attainment function, ",beta,"* = ", a, "%"),
                          list(a = formatC(res$threshold, digits = 2, format = "f"))))

# Now print Vorob'ev deviation
VD <- vorobDev(CPFs, res$VE, reference = c(2, 200))
print(VD)
# Now display the symmetric deviation function.
symDifPlot(CPFs, res$VE, res$threshold, nlevels = 11)
# Levels are adjusted automatically if too large.
symDifPlot(CPFs, res$VE, res$threshold, nlevels = 200, legend.pos = "none")

# Use a different palette.
symDifPlot(CPFs, res$VE, res$threshold, nlevels = 11, col.fun = heat.colors)
```

---

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(
  data,
  reference,
  ideal,
  maximise = FALSE,
  dist = list(type = "uniform"),
  nsamples = 100000L
)
```

Arguments

`data`	(`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.
`ideal`	(`numeric()`) Ideal point as a vector of numerical values.
`maximise`	(`logical()` | `logical(1)`) 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.
`dist`	(`list()`) weight distribution. See Details.
`nsamples`	(`integer(1)`) number of samples for Monte-Carlo sampling.

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:

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

Value

A single numerical value.

References

Anne Auger, Johannes Bader, Dimo Brockhoff, Eckart Zitzler (2009). "Articulating User Preferences in Many-Objective Problems by Sampling the Weighted Hypervolume." In Franz Rothlauf (ed.), *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2009*, 555–562. ACM Press, New York, NY.

See Also

`read_datasets()`, `eafdiff()`, `whv_rect()`

Examples

```
whv_hype (matrix(2, ncol=2), reference = 4, ideal = 1)

whv_hype (matrix(c(3,1), ncol=2), reference = 4, ideal = 1)

whv_hype (matrix(2, ncol=2), reference = 4, ideal = 1,
          dist = list(type="exponential", mu=0.2))

whv_hype (matrix(c(3,1), ncol=2), reference = 4, ideal = 1,
          dist = list(type="exponential", mu=0.2))

whv_hype (matrix(2, ncol=2), reference = 4, ideal = 1,
          dist = list(type="point", mu=c(1,1)))

whv_hype (matrix(c(3,1), ncol=2), reference = 4, ideal = 1,
          dist = list(type="point", mu=c(1,1)))
```

`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(data, rectangles, reference, maximise = FALSE)

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

Arguments

`data`	(`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()` | `logical(1)`) 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 (resp. maximum if maximising that objective) of each objective in `data`.
`scalefactor`	(`numeric(1)`) real value within $(0, 1]$ that scales the overall weight of the differences. This is parameter psi (ψ) in Diaz and López-Ibáñez (2021).

Details

TODO

Value

A single numerical value.

References

Juan Esteban Diaz, Manuel López-Ibáñez (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 *Write data sets*

Description

Write data sets to a file in the same format as `read_datasets()`.

Usage

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

Arguments

`x`	The data set to write. The last column must be the set number.
`file`	either a character string naming a file or a connection open for writing. `""` indicates output to the console.

See Also

`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