Package 'IOHanalyzer'

March 1, 2024

Type Package

Title Data Analysis Part of 'IOHprofiler'

Version 0.1.8.10

Maintainer Diederick Vermetten <d.l.vermetten@liacs.leidenuniv.nl>

Description The data analysis module for the Iterative Optimization Heuristics Profiler ('IOHprofiler'). This module provides statistical analysis methods for the benchmark data generated by optimization heuristics, which can be visualized through a web-based interface. The benchmark data is usually generated by the experimentation module, called 'IOHexperimenter'. 'IOHanalyzer' also supports the widely used 'COCO' (Comparing Continuous Optimisers) data format for benchmarking.

License BSD_3_clause + file LICENSE

Encoding UTF-8

LazyData true

URL https://iohanalyzer.liacs.nl,
https://github.com/IOHprofiler/IOHAnalyzer

BugReports https://github.com/IOHprofiler/IOHAnalyzer/issues

Imports magrittr, dplyr, data.table, ggplot2, plotly, colorspace, RColorBrewer, shiny, reshape2, stringi, httr, knitr, methods, rjson, eaf, viridis

LinkingTo Rcpp

SystemRequirements C++

RoxygenNote 7.2.3

Suggests Rcpp, testthat, withr, ComplexHeatmap, grid, keyring, PlayerRatings, xtable, shinyjs, colourpicker, bsplus, DT, kableExtra, markdown, igraph, shinydashboard, RVCompare, reticulate

Depends R (>= 2.10)

NeedsCompilation yes

Author Hao Wang [aut] (<https://orcid.org/0000-0002-4933-5181>),
Diederick Vermetten [cre, aut]
(<https://orcid.org/0000-0003-3040-7162>),
Carola Doerr [aut] (<https://orcid.org/0000-0002-4981-3227>),
Thomas Bäck [aut] (<https://orcid.org/0000-0001-6768-1478>)

Repository CRAN

Date/Publication 2024-03-01 12:52:37 UTC

R topics documented:

==.DataSet *S3 generic == operator for DataSets*

Description

S3 generic == operator for DataSets

Usage

```
## S3 method for class 'DataSet'
dsL == dsR
```

Arguments

dsL	A 'DataSet' object
dsR	A 'DataSet' object

Value

True if the DataSets contain the same function, dimension and algorithm, and have the exact same attributes

Examples

```
dsl[[1]] == dsl[[2]]
```

arrange *S3 sort function for DataSetList*

Description

Sorts a DataSetList based on the custom specified attributes ('algId', 'DIM' or 'funcId'). Default is as ascending, can be made descending by adding a - in front of the attribute. Sorting accross multiple attributes is supported, in the order they are specified.

Usage

```
arrange(dsl, ...)

## S3 method for class 'DataSetList'
arrange(dsl, ...)
```

Arguments

dsl	The DataSetList to sort
...	attribute by which 'dsl' is sorted. Multiple attributes can be specified.

Examples

```
arrange(dsl, DIM, -funcId, algId)
```

`as.character.DataSet` *S3 generic as.character operator for DataSet*

Description

S3 generic as.character operator for DataSet

Usage

```
## S3 method for class 'DataSet'
as.character(x, verbose = F, ...)
```

Arguments

`x`	A DataSet object
`verbose`	Verbose mode, currently not implemented
`...`	Arguments passed to other methods

Value

A short description of the DataSet

Examples

```
as.character(dsl[[1]])
```

`AUC` *Area Under Curve (Empirical Cumulative Dsitribution Function)*

Description

Area Under Curve (Empirical Cumulative Dsitribution Function)

Usage

```
AUC(fun, from = NULL, to = NULL)

## S3 method for class 'ECDF'
AUC(fun, from = NULL, to = NULL)
```

Arguments

fun	A ECDF object.
from	double. Starting point of the area on x-axis
to	double. Ending point of the area on x-axis

Value

a object of type 'ECDF'

Examples

```
ecdf <- ECDF(dsl,c(12,14))
AUC(ecdf, 0, 100)
```

---

bootstrap_RT *Bootstrapping for running time samples*

Description

Bootstrapping for running time samples

Usage

```
bootstrap_RT(x, max_eval, bootstrap.size)
```

Arguments

x	A numeric vector. A sample of the running time.
max_eval	A numeric vector, containing the maximal running time in each run. It should have the same size as x
bootstrap.size	integer, the size of the bootstrapped sample

Value

A numeric vector of the bootstrapped running time sample

Examples

```
ds <- dsl[[1]]
x <- get_RT_sample(ds, ftarget = 16, output = 'long')
max_eval <- get_maxRT(dsl, output = 'long')
bootstrap_RT(x$RT, max_eval$maxRT, bootstrap.size = 30)
```

c.DataSet *S3 concatenation function for DataSet*

Description

Concatenation for DataSets. Combines multiple runs from separate DataSets into a single DataSet object if all provided arguments have the same dimension, function ID and algorithm ID, and each contains only a single run. Currently does not support parameter tracking

Usage

```
## S3 method for class 'DataSet'
c(...)
```

Arguments

...	The DataSets to concatenate

Value

A new DataSet

Examples

```
c(dsl[[1]], dsl[[1]])
```

c.DataSetList *S3 concatenation function for DataSetList*

Description

S3 concatenation function for DataSetList

Usage

```
## S3 method for class 'DataSetList'
c(...)
```

Arguments

...	The DataSetLists to concatenate

Value

A new DataSetList

Examples

```
c(dsl[1], dsl[3])
```

---

cat.DataSet *S3 generic cat operator for DataSet*

Description

S3 generic cat operator for DataSet

Usage

```
cat.DataSet(x)
```

Arguments

x	A DataSet object

Value

A short description of the DataSet

Examples

```
cat.DataSet(dsl[[1]])
```

---

change_id *Add unique identifiers to each DataSet in the provided DataSetList based on static attributes*

Description

Note that this function returns a new DataSetList object, since a split into new datasetlist has to be done to ensure each dataset has exactly one unique identifier. Note that only static attributes (see 'get_static_attributes') can be used to create unique identifiers.

Usage

```
change_id(dsl, attrs)
```

Arguments

dsl	The DataSetList
attrs	The list of attributes to combine into a unique identifier

Value

A new DataSetList object where the split has been done based on the provided attributes, and the unique identifier has been added.

Examples

```
change_id(dsl, c('instance'))
```

---

check_dsc_configured *Verify that the credentials for DSCtool have been set*

Description

This uses the keyring package to store and load credentials. If the keyring package does not exists, it will default to look for a config-file in the 'repository'-folder, under your home directory. This can be changed by setting the option IOHprofiler.config_dir If you already have an account, please call 'set_DSC_credentials' with the corresponding username and password. If you don't have an account, you can register for one using 'register_DSC'

Usage

```
check_dsc_configured()
```

Examples

```
check_dsc_configured()
```

---

check_format *Check the format of data*

Description

Throws a warning when multiple formats are found in the same folder.

Usage

```
check_format(path)
```

Arguments

path	The path to the folder to check

Value

The format of the data in the given folder. Either 'COCO', 'IOHprofiler', 'NEVERGRAD' or 'SOS'.

Examples

```
path <- system.file("extdata", "ONE_PLUS_LAMDA_EA", package = "IOHanalyzer")
check_format(path)
```

---

clean_DataSetList *Clean DataSetList object by concatenating DataSets*

Description

Concatenates all DataSets with the same ID, algid, function id and dimension

Usage

```
clean_DataSetList(dsList)
```

Arguments

dsList	The DataSetList object to clean

Examples

```
clean_DataSetList(dsl)
```

DataSet *Constructor of S3 class 'DataSet'*

Description

DataSet contains the following attributes * funId * DIM * algId * datafile * instance * maxEvals * finalFunEvals * comment * Additional attributes based on the original format

Usage

```
DataSet(
  info,
  verbose = F,
  maximization = NULL,
  format = IOHprofiler,
  subsampling = FALSE,
  full_sampling = FALSE
)
```

Arguments

info	A List. Contains a set of in a *.info file.
verbose	Logical.
maximization	Logical. Whether the underlying optimization algorithm performs a maximization? Set to NULL to determine automatically based on format
format	A character. The format of data source, either 'IOHProfiler', 'COCO' or 'TWO_COL"
subsampling	Logical. Whether *.cdat files are subsampled?
full_sampling	Logical. Whether the raw (unaligned) FV matrix should be stored. Currenlt only useful when a correlation plot between function values and parameters should be made

Value

A S3 object 'DataSet'

Examples

```
path <- system.file('extdata', 'ONE_PLUS_LAMDA_EA', package = 'IOHanalyzer')
info <- read_index_file(file.path(path, 'IOHprofiler_f1_i1.info'))
DataSet(info[[1]])
```

DataSetList *S3 constructor of the 'DataSetList'*

Description

Attributes funId DIM algId

Usage

```
DataSetList(
  path = NULL,
  verbose = T,
  print_fun = NULL,
  maximization = NULL,
  format = IOHprofiler,
  subsampling = FALSE,
  full_aggregation = TRUE
)
```

Arguments

path	Path to the data files. Will look for all .info-files in this directory and use the corresponding datafiles to create the DataSetList
verbose	Logical.
print_fun	Function used to print output when in verbose mode
maximization	Logical. Whether the underlying optimization algorithm performs a maximization?
format	A character. The format of data source, options are: • 'IOHProfiler' • 'COCO' • 'TWO_COL' • 'COCO_BIOBJ' • 'NEVERGRAD' • 'SOS' These formats are specified in more detail in our github wiki.
subsampling	Logical. Whether *.cdat files are subsampled?
full_aggregation	If True, individual DataSets are aggregated as much as possible: all DataSets with the same algorithmname, function id and dimension are combined together. This leads to information loss related to static variables, so only use if that information is not required.

Value

A DataSetList object

Examples

```
path <- system.file("extdata", "ONE_PLUS_LAMDA_EA", package = "IOHanalyzer")
DataSetList(path)
```

---

dsl *Example DataSetList used in tests / examples*

Description

A DataSetList containing DataSets on 2 IOHProfiler functions from 2 algorithms in 16D

Usage

```
dsl
```

Format

DataSetList

Examples

```
summary(dsl)
```

`dsl_large` *Larger example DataSetList used in tests / examples*

Description

A DataSetList containing DataSets on all IOHProfiler functions from 11 algorithms in 100D

Usage

```
dsl_large
```

Format

DataSetList

Examples

```
summary(dsl_large)
```

`ECDF` *Empirical Cumulative Dsitribution Function of Runtime of a single data set*

Description

Empirical Cumulative Dsitribution Function of Runtime of a single data set

Usage

```
ECDF(ds, ftarget, ...)

## S3 method for class 'DataSet'
ECDF(ds, ftarget, ...)

## S3 method for class 'DataSetList'
ECDF(ds, ftarget, ...)
```

Arguments

`ds`	A DataSet or DataSetList object.
`ftarget`	A Numerical vector. Function values at which runtime values are consumed
`...`	Arguments passed to other methods

Value

a object of type 'ECDF'

Examples

```
ECDF(dsl,c(12,14))
ECDF(dsl[[1]],c(12,14))
```

---

fast_RT_samples *Function to get just the RT samples needed, without any formatting to improve speed*

Description

Function to get just the RT samples needed, without any formatting to improve speed

Usage

```
fast_RT_samples(RT_mat, target, maximization = F)
```

Arguments

RT_mat	A matrix containing the RT-values of a dataset
target	Which target-value to use
maximization	Whether maximization is needed or not

---

generate_data.Aggr *Generate dataframe of a single function/dimension pair*

Description

This function generates a dataframe which can be easily plotted using the 'plot_general_data'-function

Usage

```
generate_data.Aggr(dsList, aggr_on = "funcId", targets = NULL, which = "by_RT")
```

Arguments

dsList	The DataSetList object
aggr_on	Which attribute to use for aggregation. Either 'funcId' or 'DIM'
targets	Optional list of target values (Runtime or target value)
which	Whether to use a fixed-target 'by_RT' perspective or fixed-budget 'by_FV'

Examples

```
generate_data.Aggr(dsl)
```

`generate_data.AUC` *Generate dataframe containing the AUC for any ECDF-curves*

Description

This function generates a dataframe which can be easily plotted using the 'plot_general_data'-function

Usage

```
generate_data.AUC(
  dsList,
  targets,
  scale_log = F,
  which = "by_RT",
  dt_ecdf = NULL,
  multiple_x = FALSE,
  normalize = T
)
```

Arguments

`dsList`	The DataSetList object
`targets`	A list or data.table containing the targets per function / dimension. If this is a data.table, it needs columns 'target', 'DIM' and 'funcId'
`scale_log`	Whether to use logarithmic scaling or not
`which`	Whether to use a fixed-target 'by_RT' perspective or fixed-budget 'by_FV'
`dt_ecdf`	A data table of the ECDF to avoid needless recomputations. Will take preference if it is provided together with dsList and targets
`multiple_x`	Boolean, whether to get only the total AUC or get stepwise AUC values
`normalize`	Whether to normalize the resulting AUC values to [0,1] or not

Examples

```
generate_data.AUC(dsl, get_ECDF_targets(dsl))
generate_data.AUC(NULL, NULL, dt_ecdf = generate_data.ECDF(dsl, get_ECDF_targets(dsl)))
```

`generate_data.CDP` *Generate data for the cumulative difference plot.*

Description

This function generates a dataframe that can be used to generate the ‘cumulative_difference_plot‘.

Usage

```
generate_data.CDP(
  dsList,
  runtime_or_target_value,
  isFixedBudget,
  alpha = 0.05,
  EPSILON = 1e-80,
  nOfBootstrapSamples = 1000
)
```

Arguments

`dsList`	The DataSetList object. Note that the ‘cumulative_difference_plot‘ can only compare two algorithms in a single problem of dimension one.
`runtime_or_target_value`	The target runtime or the target value
`isFixedBudget`	Should be TRUE when target runtime is used. False otherwise.
`alpha`	1 minus the confidence level of the confidence band.
`EPSILON`	If abs(x-y) < EPSILON, then we assume that x = y.
`nOfBootstrapSamples`	The number of bootstrap samples used in the estimation.

Value

A dataframe with the data to generate the cumulative difference plot.

Examples

```
dsl_sub <- subset(dsl, funcId == 1)
generate_data.CDP(dsl_sub, 15, TRUE, nOfBootstrapSamples = 10)
```

generate_data.EAF *Generate dataframe consisting of the levelsets of the EAF*

Description

This function generates a dataframe which can be easily plotted using the ‘plot_eaf_data‘-function

Usage

```
generate_data.EAF(
  dsList,
  n_sets = 11,
  subsampling = 100,
  scale_xlog = F,
  xmin = "",
  xmax = ""
)
```

Arguments

dsList	The DataSetList object
n_sets	The number of level sets to calculate
subsampling	Level of subsampling to use for runtime-values (number of runtimes to consider). Setting to 0 will make the calculations more precise at the cost of potentially much longer exectution times
scale_xlog	Only has effect when ‘subsampling‘ is True. The scaling of the subsampled runtimes When true, these are equally spaced in log-space, when false they are linearly spaced.
xmin	Minimum runtime value
xmax	Maximum runtime value

Examples

```
generate_data.EAF(subset(dsl, funcId == 1))
```

generate_data.EAF_Difference *Generate differences between two EAFs*

Description

This function uses the ’eaf’ package to calculate eaf differences

Usage

```
generate_data.EAF_Difference(dsList1, dsList2)
```

Arguments

dsList1	The first DataSetList object
dsList2	The second DataSetList object

Examples

```
generate_data.EAF_Difference(dsl[1], dsl[3])
```

---

generate_data.EAF_diff_Approximate

Generate EAF-differences between each function and the remaining portfolio

---

Description

This is an approximation of ", since the number of required polygons can quickly become problematic for plotly. This function uses discretized contour matrices instead, which trades off accuracy for scalability.

Usage

```
generate_data.EAF_diff_Approximate(
  dsList,
  xmin,
  xmax,
  ymin,
  ymax,
  x.log = T,
  y.log = T
)
```

Arguments

dsList	The DataSetList object, containing at least 2 IDs
xmin	Minimum runtime to consider
xmax	Maximum runtime to consider
ymin	Minimum f(x) to consider
ymax	Maximum f(x) to consider
x.log	Whether to scale the y-space logarithmically
y.log	Whether to scale the y-space logarithmically

Examples

```
generate_data.EAF_diff_Approximate(subset(dsl, funcId == 1), 1, 16, 1, 16)
```

`generate_data.ECDF` *Generate dataframe of a single function/dimension pair*

Description

This function generates a dataframe which can be easily plotted using the ‘plot_general_data‘-function

Usage

```
generate_data.ECDF(
  dsList,
  targets,
  scale_log = F,
  which = "by_RT",
  use_full_range = TRUE
)
```

Arguments

`dsList`	The DataSetList object
`targets`	A list or data.table containing the targets per function / dimension. If this is a data.table, it needs columns ’target’, ’DIM’ and ’funcId’
`scale_log`	Wheterh to use logarithmic scaling or not
`which`	Whether to use a fixed-target ’by_RT’ perspective or fixed-budget ’by_FV’
`use_full_range`	Whether or not to use the full range of the x-axis or cut it off as soon as all algorithms reach 98% success (+10% buffer). Only supported in the case of one function and dimension

Examples

```
generate_data.ECDF(subset(dsl, funcId == 1), c(10, 15, 16))
```

generate_data.ECDF_From_EAF

Generate dataframe consisting of the ECDF-equivalent based on the EAF

Description

This function uses EAF-data to calculate a target-independent version of the ECDF

Usage

```
generate_data.ECDF_From_EAF(
  eaf_table,
  min_val,
  max_val,
  maximization = F,
  scale_log = F,
  normalize = T
)
```

Arguments

eaf_table	Datatable resulting from the 'generate_data.EAF' function
min_val	Minimum value to use for y-space
max_val	Maximum value to use for y-space
maximization	Whether the data resulted from maximization or not
scale_log	Whether to use logarithmic scaling in y-space before calculating the partial integral
normalize	Whether to normalize the resulting integrals to [0,1] (Based on 'min_val' and 'max_va')

Examples

```
generate_data.ECDF_From_EAF(generate_data.EAF(subset(dsl, funcId == 1)), 1, 16, maximization = TRUE)
```

generate_data.ECDF_raw

Generate dataframe of a the unaggregated values of individual algorithms. Stripped-down version of

Description

This provides an unaggregated version of the function 'generate_data.ECDF'.

Usage

```
generate_data.ECDF_raw(dsList, targets, scale_log = F)
```

Arguments

dsList	The DataSetList object
targets	A list or data.table containing the targets per function / dimension. If this is a data.table, it needs columns 'target', 'DIM' and 'funcId'
scale_log	Whetherh to use logarithmic scaling or not

Examples

```
generate_data.ECDF_raw(subset(dsl, funcId == 1), c(10, 15, 16))
```

---

generate_data.Heatmaps

Nevergrad-dashboard based algorithm comparison

Description

This procedure calculates the fraction of times algorithm A is better than algorithm B according to their mean on each function,dimension,target tuple

Usage

```
generate_data.Heatmaps(dsList, which = "by_FV", target_dt = NULL)
```

Arguments

dsList	The DataSetList, can contain multiple functions and dimensions, but should have the same algorithms for all of them. For functions/dimensions where this is not the case, all algorithms are considered tied.
which	Whether to use fixed-target ('by_FV') or fixed-budget ('by_RT') perspective
target_dt	Custom data.table target value to use. When NULL, this is selected automatically.

Value

A matrix containing the pairwise win-ratios.

Examples

```
generate_data.Heatmaps(dsl)
generate_data.Heatmaps(dsl, which = 'by_RT')
```

generate_data.hist *Generate dataframe of a single function/dimension pair*

Description

This function generates a dataframe which can be easily plotted using the 'plot_general_data'-function

Usage

```
generate_data.hist(dsList, target, use.equal.bins = F, which = "by_RT")
```

Arguments

dsList	The DataSetList object
target	The target value (Runtime or target value)
use.equal.bins	Whether all bins should be equal size for each algorithm or not
which	Whether to use a fixed-target 'by_RT' perspective or fixed-budget 'by_FV'

Examples

```
generate_data.hist(subset(dsl, funcId == 1), target = 15, which = 'by_RT')
```

generate_data.Parameters *Generate dataframe of a single function/dimension pair*

Description

This function generates a dataframe which can be easily plotted using the 'plot_general_data'-function

Usage

```
generate_data.Parameters(dsList, which = "by_RT", scale_log = F)
```

Arguments

dsList	The DataSetList object
which	Whether to use a fixed-target 'by_RT' perspective or fixed-budget 'by_FV'
scale_log	Wheterh to use logarithmic scaling or not

Examples

```
generate_data.Parameters(subset(dsl, funcId == 1))
```

generate_data.Parameter_correlation

Generate dataframe of exactly 2 parameters, matched by running time

Description

This function generates a dataframe which can be easily plotted using the 'plot_general_data'-function

Usage

```
generate_data.Parameter_correlation(dsList, par1, par2)
```

Arguments

dsList	The DataSetList object
par1	The first parameter. Either a parameter name or 'f(x)'
par2	The second parameter. Either a parameter name or 'f(x)'

Examples

```
generate_data.Parameter_correlation(subset(dsl, funcId == 1), 'f(x)', 'f(x)')
```

generate_data.PMF

Generate dataframe of a single function/dimension pair for creating PDF or PMF plots

Description

This function generates a dataframe which can be easily plotted using the 'plot_general_data'-function

Usage

```
generate_data.PMF(dsList, target, which = "by_RT")
```

Arguments

dsList	The DataSetList object
target	The target value (Runtime or target value)
which	Whether to use a fixed-target 'by_RT' perspective or fixed-budget 'by_FV'

Examples

```
generate_data.PMF(subset(dsl, funcId == 1), target = 15, which = 'by_RT')
```

`generate_data.Single_Function` *Generate dataframe of a single function/dimension pair*

Description

This function generates a dataframe which can be easily plotted using the 'plot_general_data'-function

Usage

```
generate_data.Single_Function(
  dsList,
  start = NULL,
  stop = NULL,
  scale_log = F,
  which = "by_RT",
  include_opts = F,
  budget = NULL,
  include_geom_mean = F
)
```

Arguments

`dsList`	The DataSetList object
`start`	Optional start value (Runtime or target value)
`stop`	Optional end value (Runtime or target value)
`scale_log`	Wheterh to use logarithmic scaling or not
`which`	Whether to use a fixed-target 'by_RT' perspective or fixed-budget 'by_FV'
`include_opts`	Whether or not to also include the best value hit by each algorithm to the generated datapoints
`budget`	Optional; overwrites the budget of each individual algorithm when doing ERT calculations. Only works in fixed_target mode.
`include_geom_mean`	Boolean to indicate whether to include the geometric mean. Only works in fixed_budget mode. Negative values cause NaN, zeros cause output to be completely 0. Defaults to False.

Examples

```
generate_data.Single_Function(subset(dsl, funcId == 1), which = 'by_RT')
```

get_algId *Get all algorithm ids present in a DataSetList*

Description

Get all algorithm ids present in a DataSetList

Usage

```
get_algId(dsList)
```

Arguments

dsList	The DataSetLsit

Value

A sorted list of all unique algorithm ids which occur in the DataSetList

Examples

```
get_algId(dsl)
```

get_color_scheme *Get colors according to the current colorScheme of the IOHanalyzer*

Description

Get colors according to the current colorScheme of the IOHanalyzer

Usage

```
get_color_scheme(ids_in)
```

Arguments

ids_in	List of algorithms (or custom ids, see 'change_id') for which to get colors

Examples

```
get_color_scheme(get_algId(dsl))
```

get_color_scheme_dt *Get datatable of current color (and linestyle) scheme to file*

Description

Get datatable of current color (and linestyle) scheme to file

Usage

```
get_color_scheme_dt()
```

Value

data.table object with 3 columns: ids, colors, linestyles

Examples

```
get_color_scheme_dt()
```

get_default_ECDF_targets *Generate ECDF targets for a DataSetList*

Description

Generate ECDF targets for a DataSetList

Usage

```
get_default_ECDF_targets(data, format_func = as.integer)
```

Arguments

data	A DataSetList
format_func	function to format the targets

Value

a vector of targets

Examples

```
get_default_ECDF_targets(dsl)
```

get_dim *Get all dimensions present in a DataSetList*

Description

Get all dimensions present in a DataSetList

Usage

```
get_dim(dsList)
```

Arguments

dsList	The DataSetLsit

Value

A sorted list of all unique dimensions which occur in the DataSetList

Examples

```
get_dim(dsl)
```

get_dsc_omnibus *Perform omnibus statistical tests on the matrix of rankings from the DSCtool api*

Description

Perform omnibus statistical tests on the matrix of rankings from the DSCtool api

Usage

```
get_dsc_omnibus(res, method = NULL, alpha = 0.05)
```

Arguments

res	The result of a call to the ‘get_dsc_rank‘
method	Which method to use to do the tests. Has be be one of the allowed ones in ‘res$valid_methods‘. When NULL, the first valid option is chosen by default
alpha	Threshold value for statistical significance

Value

A named list containing the algorithm means

Examples

```
get_dsc_omnibus(get_dsc_rank(dsl, na.correction = 'PAR-10'))
```

`get_dsc_posthoc` *Perform post-hoc processing on data from DSCtool*

Description

Perform post-hoc processing on data from DSCtool

Usage

```
get_dsc_posthoc(
  omni_res,
  nr_algs,
  nr_problems,
  base_algorithm = NULL,
  method = "friedman",
  alpha = 0.05
)
```

Arguments

`omni_res`	The result from a call to ‘get_dsc_omnibus‘
`nr_algs`	The number of algorithms present in ‘omni_res‘
`nr_problems`	The number of problems present in ‘omni_res‘
`base_algorithm`	The base algorithm to which the other are compared. This has to be present in ‘omni_res$algorithm_means‘ as an ‘algorithm‘ property
`method`	Either ’friedman’ or ’friedman-aligned-rank’
`alpha`	Threshold value for statistical significance

Value

A named list containing 4 types of analyses: * Zvalue * UnadjustedPValue * Holm * Hochberg

Examples

```
get_dsc_posthoc(get_dsc_omnibus(get_dsc_rank(dsl, na.correction = 'PAR-10')), 2, 2)
```

get_dsc_rank *Get the matrix of rankings using the DSCtool api for a DataSetList*

Description

Get the matrix of rankings using the DSCtool api for a DataSetList

Usage

```
get_dsc_rank(
  dsList,
  targets = NULL,
  which = "by_RT",
  test_type = "AD",
  alpha = 0.05,
  epsilon = 0,
  monte_carlo_iterations = 0,
  na.correction = NULL
)
```

Arguments

dsList	The DataSetList object
targets	Optional list of target values (Runtime or target value)
which	Whether to use a fixed-target 'by_RT' perspective or fixed-budget 'by_FV'
test_type	Either 'AD' for Anderson-Darling or KS for Kolmogorov-Smirnov tests
alpha	Threshold value for statistical significance
epsilon	Minimum threshold to have practical difference between algorithms (eDSC)
monte_carlo_iterations	How many monte-carlo-simulations to perform (set to 0 to use regular DSC)
na.correction	How to deal with missing values. Only used in fixed-target perspective. Options are: - 'NULL': No correction is done. This will likely result in an error, as the DSCtool does not allow for na values - 'PAR-1' Replace missing values with Budget (budget taken from relevant DataSet) - 'PAR-10' Replace missing values with 10*Budget (budget taken from relevant DataSet) - 'ERT' Replace NA values with the Expected Running Time. If all values are NA, this reverts to nr_runs * budget - 'Remove-na' Removes all NA values

Value

A named list containing a ranked-matrix which has the rankin of each algorithm on each problem, as well as a list of which omnibus tests can be used to further process this data. This can be further analyzed using 'get_dsc_omnibus'

Examples

```
get_dsc_rank(dsl, na.correction = 'PAR-10')
```

get_ECDF_targets *Generation of default ECDF-targets*

Description

Generation of default ECDF-targets

Usage

```
get_ECDF_targets(dsList, type = "log-linear", number_targets = 10)
```

Arguments

dsList	The DataSetList object for which to generate the targets
type	The way to generate the targets. Either 'log-linear', 'linear' or 'bbob' (51 fixed targets, equal for all functions / dimensions)
number_targets	The amount of targets to generate

Value

A data.table with 3 columns: funcId, DIM and target

Examples

```
get_ECDF_targets(dsl, 'linear', 10)
```

get_ERT *Get Expected RunTime*

Description

Get Expected RunTime

Usage

```
get_ERT(ds, ftarget, budget, ...)

## S3 method for class 'DataSet'
get_ERT(ds, ftarget, budget = NULL, ...)

## S3 method for class 'DataSetList'
get_ERT(ds, ftarget, budget = NULL, algorithm = "all", ...)
```

Arguments

ds	A DataSet or DataSetList object
ftarget	The function target(s) for which to get the ERT
budget	Optional; overwrites the budget found in ds for ERT-calculation
...	Arguments passed to other methods
algorithm	DEPRECATED, will be removed in next release. Which algorithms in the DataSetList to consider.

Value

A data.table containing the runtime samples for each provided target function value

Examples

```
get_ERT(dsl, 14)
get_ERT(dsl[[1]], 14)
```

---

get_funcId *Get all function ids present in a DataSetList*

Description

Get all function ids present in a DataSetList

Usage

```
get_funcId(dsList)
```

Arguments

dsList	The DataSetLsit

Value

A sorted list of all unique function ids which occur in the DataSetList

Examples

```
get_funcId(dsl)
```

get_funcName *Get all function names present in a DataSetList*

Description

Get all function names present in a DataSetList

Usage

```
get_funcName(dsList)
```

Arguments

dsList	The DataSetLsit

Value

A list of all unique function names which occur in the DataSetList

Examples

```
get_funcName(dsl)
```

get_funvals *Get all function values present in a DataSetList*

Description

Get all function values present in a DataSetList

Usage

```
get_funvals(dsList)
```

Arguments

dsList	The DataSetLsit

Value

A list matrices of all function values which occur in the DataSetList

Examples

```
get_funvals(dsl)
```

get_FV *Get function value matrix of the used dataset.*

Description

To be used instead of accessing ds$FV directly, since in the case of constrained problems, the violation handling should be applied before using the function values Constraint penalty function should be set in global options, as IOHanalyzer.Violation_Function

Usage

```
get_FV(ds, ...)

## S3 method for class 'DataSet'
get_FV(ds, ...)
```

Arguments

ds	The DataSet
...	Arguments passed to other methods

Value

The matrix of FV values in the dataset, penalized if applicable.

Examples

```
get_FV(dsl[[1]])
```

get_FV_overview *Get Function Value condensed overview*

Description

Get Function Value condensed overview

Usage

```
get_FV_overview(ds, ...)

## S3 method for class 'DataSet'
get_FV_overview(ds, ...)

## S3 method for class 'DataSetList'
get_FV_overview(ds, algorithm = "all", ...)
```

Arguments

ds	A 'DataSet' or 'DataSetList' object
...	Arguments passed to other methods
algorithm	DEPRECATED, will be removed in next release. Which algorithms in the DataSetList to consider.

Value

A data.table containing the algorithm ID, best, worst and mean reached function values, the number of runs and available budget for the DataSet

Examples

```
get_FV_overview(dsl)
get_FV_overview(dsl[[1]])
get_FV_overview(dsl, algorithm = '(1+1)_greedy_hill_climber_1')
```

---

get_FV_sample *Get Funtion Value Samples*

Description

Get Funtion Value Samples

Usage

```
get_FV_sample(ds, ...)

## S3 method for class 'DataSet'
get_FV_sample(ds, runtime, output = "wide", ...)

## S3 method for class 'DataSetList'
get_FV_sample(ds, runtime, algorithm = "all", ...)
```

Arguments

ds	A DataSet or DataSetList object
...	Arguments passed to other methods
runtime	A Numerical vector. Runtimes at which function values are reached
output	A String. The format of the output data: 'wide' or 'long'
algorithm	DEPRECATED, will be removed in next release. Which algorithms in the DataSetList to consider.

Value

A data.table containing the function value samples for each provided target runtime

Examples

```
get_FV_sample(dsl, 100)
get_FV_sample(dsl[[1]], 100)
```

`get_FV_summary` *Get Function Value Summary*

Description

Get Function Value Summary

Usage

```
get_FV_summary(ds, ...)

## S3 method for class 'DataSet'
get_FV_summary(ds, runtime, include_geom_mean = F, ...)

## S3 method for class 'DataSetList'
get_FV_summary(ds, runtime, algorithm = "all", include_geom_mean = F, ...)
```

Arguments

`ds`	A DataSet or DataSetList object
`...`	Arguments passed to other methods
`runtime`	A Numerical vector. Runtimes at which function values are reached
`include_geom_mean`	Boolean to indicate whether to include the geometric mean. Only works in fixed_budget mode. Negative values cause NaN, zeros cause output to be completely 0. Defaults to False.
`algorithm`	DEPRECATED, will be removed in next release. Which algorithms in the DataSetList to consider.

Value

A data.table containing the function value statistics for each provided target runtime value

Examples

```
get_FV_summary(dsl, 100)
get_FV_summary(dsl[[1]], 100)
```

get_id *Get condensed overview of datasets*

Description

Get the unique identifiers for each DataSet in the provided DataSetList

Usage

```
get_id(ds, ...)

## S3 method for class 'DataSet'
get_id(ds, ...)

## S3 method for class 'DataSetList'
get_id(ds, ...)
```

Arguments

ds	The DataSetList
...	Arguments passed to other methods

Details

If no unique identifier is set (using ‘change_id‘ or done in DataSet construction from 1.6.0 onwards), this function falls back on returning the algorith id (from ‘get_aldId‘)to ensure backwards compatibility

Value

The list of unique identiefiers present in dsl

Examples

```
get_id(dsl)
get_id(dsl[[1]])
```

get_line_style *Get line styles according to the current styleScheme of the IOHanalyzer*

Description

Get line styles according to the current styleScheme of the IOHanalyzer

Usage

```
get_line_style(ids_in)
```

Arguments

ids_in	List of algorithms (or custom ids, see 'change_id') for which to get linestyles

Examples

```
get_line_style(get_algId(dsl))
```

get_marg_contrib_ecdf *Get the marginal contribution of an algorithm to a portfolio*

Description

Based on the contribution to the ECDF-curve of the VBS of the portfolio

Usage

```
get_marg_contrib_ecdf(id, perm, j, dt)
```

Arguments

id	The id for which to get the contribution
perm	The permutation of algorithms to which is being contributed
j	At which point in the permutation the contribution should be measured
dt	The datatable in which the raw ecdf-values are stored (see 'generate_data.ECDF_raw')

Examples

```
dt <- generate_data.ECDF_raw(dsl, get_ECDF_targets(dsl))
get_marg_contrib_ecdf(get_id(dsl)[[1]], get_id(dsl), 1, dt)
```

get_maxRT *Get the maximal running time*

Description

Get the maximal running time

Usage

```
get_maxRT(ds, ...)

## S3 method for class 'DataSet'
get_maxRT(ds, output = "wide", ...)

## S3 method for class 'DataSetList'
get_maxRT(ds, algorithm = "all", ...)
```

Arguments

ds	A DataSet or DataSetList object
...	Arguments passed to other methods
output	The format of the outputted table: 'wide' or 'long'
algorithm	DEPRECATED, will be removed in next release. Which algorithms in the DataSetList to consider.

Value

A data.table object containing the algorithm ID and the running time when the algorithm terminates in each run

Examples

```
get_maxRT(dsl)
get_maxRT(dsl[[1]])
```

get_ontology_data *Get the list of available options for data from the OPTION ontology*

Description

Get the list of available options for data from the OPTION ontology

Usage

```
get_ontology_data(
  datasource,
  fids,
  dims,
  algs,
  iids = NULL,
  funcsuites = NULL,
  min_target = NULL,
  max_target = NULL,
  min_budget = NULL,
  max_budget = NULL
)
```

Arguments

datasource	The datasource: either BBOB or Nevergrad
fids	The function names as given by ‘get_ontology_var‘
dims	The dimensionalities as given by ‘get_ontology_var‘
algs	The algorithm names as given by ‘get_ontology_var‘
iids	The instances as given by ‘get_ontology_var‘ (only for BBOB data)
funcsuites	The function suite as given by ‘get_ontology_var‘ (only for Nevergrad data)
min_target	The minimum target value for which to return data
max_target	The maximum target value for which to return data
min_budget	The minimum budget value for which to return data
max_budget	The maximum budget value for which to return data

Value

a DataSetList object matching the selected attributes.

Examples

```
get_ontology_data("BBOB", "f5", 5, "IPOP400D", 1)
```

---

get_ontology_var *Get the list of available options for data from the OPTION ontology*

Description

Get the list of available options for data from the OPTION ontology

Usage

```
get_ontology_var(varname, datasource = NULL, study = NULL, algs = NULL, ...)
```

Arguments

varname	The variable for which to get the options. Restricted to [Fid, Iid, DIM, AlgId, Suite]
datasource	The datasource for which to get the attributes. Either BBOB or Nevergrad, or NULL if looking at a specific ‘study‘ argument
study	Which study to load the requested variables for (NULL if no study is considered)
algs	Which algorithms to get the requested variables for. Required for varnames in [Fid, Iid, DIM]
...	Additional arguments to the OPTION call. Currently only supports 'Suite' for nevergrad.

Value

the options of varname given the specified datasource

Examples

```
get_ontology_var("Fid", "BBOB")
```

---

get_overview *Get condensed overview of datasets*

Description

Get condensed overview of datasets

Usage

```
get_overview(ds, ...)

## S3 method for class 'DataSet'
get_overview(ds, ...)

## S3 method for class 'DataSetList'
get_overview(ds, ...)
```

Arguments

ds	A DataSet or DataSetList object
...	Arguments passed to other methods

Value

A data.table containing some basic information about the provided DataSet(List)

Examples

```
get_overview(dsl)
get_overview(dsl[[1]])
```

`get_parId` *Get all parameter ids present in a DataSetList*

Description

Get all parameter ids present in a DataSetList

Usage

```
get_parId(dsList, which = "by_FV")
```

Arguments

`dsList`	The DataSetList
`which`	A string takes values in 'c('by_FV', 'by_RT')'. To choose the parameters aligned by the running time (RT) or the function value (FV). Note that parameters in each case are not necessary the same.

Value

A sorted list of all unique parameter ids which occur in the DataSetList

Examples

```
get_parId(dsl)
```

`get_PAR_name` *Get the parameter names of the algorithm*

Description

Get the parameter names of the algorithm

Usage

```
get_PAR_name(ds, which)

## S3 method for class 'DataSet'
get_PAR_name(ds, which = "by_FV")
```

Arguments

ds	A DataSet object
which	a string takes it value in ‘c(’by_FV’, ’by_RT’)‘, indicating the parameters aligned against the running time (RT) or function value (FV). ‘’by_FV’‘ is the default value.

Value

a character list of paramter names, if recorded in the data set

Examples

```
get_PAR_name(dsl[[1]])
```

---

get_PAR_sample *Get Parameter Value Samples*

Description

Get Parameter Value Samples

Usage

```
get_PAR_sample(ds, idxValue, ...)

## S3 method for class 'DataSet'
get_PAR_sample(
  ds,
  idxValue,
  parId = "all",
  which = "by_FV",
  output = "wide",
  ...
)

## S3 method for class 'DataSetList'
get_PAR_sample(ds, idxValue, algorithm = "all", ...)
```

Arguments

ds	A DataSet or DataSetList object
idxValue	A Numerical vector. Index values at which parameter values are observed. The index value can either take its value in the range of running times, or function values. Such a value type is signified by ‘which‘ parameter.
...	Arguments passed to other methods
parId	A character vector. Either ’all’ or the name of parameters to be retrieved

which	A string takes values in ‘c(’by_FV’, ’by_RT’)‘, indicating the parameters to be retrieved are aligned against the running time (RT) or function value (FV). ‘’by_FV’‘ is the default value.
output	A character. The format of the output data: ’wide’ or ’long’
algorithm	DEPRECATED, will be removed in next release. Which algorithms in the DataSetList to consider.

Value

A data.table object containing parameter values aligned at each given target value

Examples

```
get_PAR_sample(dsl, 14)
get_PAR_sample(dsl[[1]], 14)
```

---

get_PAR_summary *Get Parameter Value Summary*

Description

Get Parameter Value Summary

Usage

```
get_PAR_summary(ds, idxValue, ...)

## S3 method for class 'DataSet'
get_PAR_summary(ds, idxValue, parId = "all", which = "by_FV", ...)

## S3 method for class 'DataSetList'
get_PAR_summary(ds, idxValue, algorithm = "all", ...)
```

Arguments

ds	A DataSet or DataSetList object
idxValue	A Numerical vector. Index values at which parameter values are observed. The index value can either take its value in the range of running times, or function values. Such a value type is signified by ‘which‘ parameter.
...	Arguments passed to other methods
parId	A character vector. Either ’all’ or the name of parameters to be retrieved
which	A string takes values in ‘c(’by_FV’, ’by_RT’)‘, indicating the parameters to be retrieved are aligned against the running time (RT) or function value (FV). ‘’by_FV’‘ is the default value.
algorithm	DEPRECATED, will be removed in next release. Which algorithms in the DataSetList to consider.

Value

A data.table object containing basic statistics of parameter values aligned at each given target value

Examples

```
get_PAR_summary(dsl, 14)
get_PAR_summary(dsl[[1]], 14)
```

---

get_position_dsl *Extract the position information from a datasetlist object*

Description

Extract the position information from a datasetlist object

Usage

```
get_position_dsl(dsList, iid)
```

Arguments

dsList	The DataSetList object
iid	the Instance Id from which to get the position history (can be a list)

Examples

```
get_position_dsl(subset(dsl, funcId == 1), 1)
```

---

get_RT *Get runtime matrix of the used dataset.*

Description

To be used instead of accessing ds$RT directly, since in the case of constrained problems, the violation handling should be applied before using the function values Constraint penalty function should be set in global options, as IOHanalyzer.Violation_Function

Usage

```
get_RT(ds, ...)

## S3 method for class 'DataSet'
get_RT(ds, ...)
```

Arguments

ds	The DataSet
...	Arguments passed to other methods

Value

The matrix of FV values in the dataset, penalized if applicable.

Examples

```
get_RT(dsl[[1]])
```

---

get_RT_overview *Get Runtime Value condensed overview*

Description

Get Runtime Value condensed overview

Usage

```
get_RT_overview(ds, ...)

## S3 method for class 'DataSet'
get_RT_overview(ds, ...)

## S3 method for class 'DataSetList'
get_RT_overview(ds, algorithm = "all", ...)
```

Arguments

ds	A DataSet or DataSetList object
...	Arguments passed to other methods
algorithm	DEPRECATED, will be removed in next release. Which algorithms in the DataSetList to consider.

Value

A data.table containing the algorithm ID, minimum and maximum used evaluations, number of runs and available budget for the DataSet

Examples

```
get_RT_overview(dsl)
get_RT_overview(dsl[[1]])
```

get_RT_sample *Get RunTime Sample*

Description

Get RunTime Sample

Usage

```
get_RT_sample(ds, ftarget, ...)

## S3 method for class 'DataSet'
get_RT_sample(ds, ftarget, output = "wide", ...)

## S3 method for class 'DataSetList'
get_RT_sample(ds, ftarget, algorithm = "all", ...)
```

Arguments

ds	A DataSet or DataSetList object
ftarget	A Numerical vector. Function values at which runtime values are consumed
...	Arguments passed to other methods
output	A character determining the format of output data.table: 'wide' or 'long'
algorithm	DEPRECATED, will be removed in next release. Which algorithms in the DataSetList to consider.

Value

A data.table containing the runtime samples for each provided target function value

Examples

```
get_RT_sample(dsl, 14)
get_RT_sample(dsl[[1]], 14)
```

get_RT_summary *Get RunTime Summary*

Description

Get RunTime Summary

Usage

```
get_RT_summary(ds, ftarget, budget, ...)

## S3 method for class 'DataSet'
get_RT_summary(ds, ftarget, budget = NULL, ...)

## S3 method for class 'DataSetList'
get_RT_summary(ds, ftarget, budget = NULL, ...)
```

Arguments

ds	A DataSet or DataSetList object
ftarget	The function target(s) for which to get the runtime summary
budget	Optional; overwrites the budget found in ds for ERT-calculation
...	Arguments passed to other methods

Value

A data.table containing the runtime statistics for each provided target function value

Examples

```
get_RT_summary(dsl, 14)
get_RT_summary(dsl[[1]], 14)
```

---

get_runtimes *Get all runtime values present in a DataSetList*

Description

Get all runtime values present in a DataSetList

Usage

```
get_runtimes(dsList)
```

Arguments

dsList	The DataSetLsit

Value

A list matrices of all runtime values which occur in the DataSetList

Examples

```
get_runtimes(dsl)
```

get_shapley_values *Get the shapley-values of a portfolio of algorithms*

Description

Based on the contribution to the ECDF-curve of the VBS of the portfolio

Usage

```
get_shapley_values(
  dsList,
  targets,
  scale.log = T,
  group_size = 5,
  max_perm_size = 10,
  normalize = T
)
```

Arguments

dsList	The DataSetList object
targets	A list or data.table containing the targets per function / dimension. If this is a data.table, it needs columns 'target', 'DIM' and 'funcId'
scale.log	Whether to use logarithmic scaling for the runtimes at which the ecdf will be sampled or not
group_size	How many permutation groups will be considered
max_perm_size	The maximum limit for permutations to be considered
normalize	Whether or not to ensure the resulting values will be in [0,1]

Examples

```
dsl_sub <- subset(dsl, funcId == 1)
get_shapley_values(dsl_sub, get_ECDF_targets(dsl_sub), group_size = 2)
```

get_static_attributes *Get all attributes which can be used to subset a DataSetList*

Description

Get all attributes which can be used to subset a DataSetList

Usage

```
get_static_attributes(dsl)
```

Arguments

dsl	The DataSetList

Value

The list of available attributes

Examples

```
get_static_attributes(dsl)
```

---

get_static_attribute_values

Get all options for a specific attribute which can be used to subset a DataSetList

---

Description

This is a more generic version of the existing ‘get_dim‘, ‘get_funcId‘ and ‘get_algId‘ functions. Note the only attributes returned by ‘get_static_attributes‘ are supported in this funcion

Usage

```
get_static_attribute_values(dsl, attribute)
```

Arguments

dsl	The DataSetList
attribute	the name of the attribute for which to get the available options in dsl

Value

The list of options for the specified attribute

Examples

```
get_static_attribute_values(dsl, 'funcId')
```

get_target_dt *Generate datatables of runtime or function value targets for a DataSetList*

Description

Only one target is generated per (function, dimension)-pair, as opposed to the function 'get_default_ECDF_targets', which generates multiple targets.

Usage

```
get_target_dt(dsList, which = "by_RT")
```

Arguments

dsList	A DataSetList
which	Whether to generate fixed-target ('by_FV') or fixed-budget ('by_RT') targets

Value

a data.table of targets

Examples

```
get_target_dt(dsl)
```

glicko2_ranking *Glicko2 raning of algorithms*

Description

This procedure ranks algorithms based on a glicko2-procedure. Every round (total nr_rounds), for every function and dimension of the datasetlist, each pair of algorithms competes. This competition samples a random runtime for the provided target (defaults to best achieved target). Whichever algorithm has the lower runtime wins the game. Then, from these games, the glicko2-rating is determined.

Usage

```
glicko2_ranking(dsl, nr_rounds = 100, which = "by_FV", target_dt = NULL)
```

Arguments

dsl	The DataSetList, can contain multiple functions and dimensions, but should have the same algorithms for all of them
nr_rounds	The number of rounds to run. More rounds leads to a more accurate ranking.
which	Whether to use fixed-target ('by_FV') or fixed-budget ('by_RT') perspective
target_dt	Custom data.table target value to use. When NULL, this is selected automatically.

Value

A dataframe containing the glicko2-ratings and some additional info

Examples

```
glicko2_ranking(dsl, nr_round = 25)
glicko2_ranking(dsl, nr_round = 25, which = 'by_RT')
```

IOHanalyzer *IOHanalyzer: Data Analysis Part of IOHprofiler*

Description

The data analysis module for the Iterative Optimization Heuristics Profiler (IOHprofiler). This module provides statistical analysis methods for the benchmark data generated by optimization heuristics, which can be visualized through a web-based interface. The benchmark data is usually generated by the experimentation module, called IOHexperimenter. IOHanalyzer also supports the widely used COCO (Comparing Continuous Optimisers) data format for benchmarking.

Functions

The IOHanalyzer consists of 3 main functionalities:

- Reading and alligning data from different heuristics, such as IOHExperimenter. This is done using the DataSet and DataSetList functions
- Processing and summarizing this data
- Creating various plots

Author(s)

Maintainer: Diederick Vermetten <d.l.vermetten@liacs.leidenuniv.nl> (ORCID)

Authors:

- Hao Wang <h.wang@liacs.leidenuniv.nl> (ORCID)
- Carola Doerr <Carola.Doerr@mpi-inf.mpg.de> (ORCID)
- Thomas Bäck <t.h.w.baeck@liacs.leidenuniv.nl> (ORCID)

See Also

Useful links:

- https://iohanalyzer.liacs.nl
- https://github.com/IOHprofiler/IOHAnalyzer
- Report bugs at https://github.com/IOHprofiler/IOHAnalyzer/issues

Examples

```
path <- system.file("extdata", "ONE_PLUS_LAMDA_EA", package="IOHanalyzer")
dsList <- DataSetList(path)
summary(dsList)
Plot.RT.Single_Func(dsList[1])

## Not run:
runServer()

## End(Not run)
```

IOH_plot_ly_default *Template for creating plots in the IOHanalyzer-style*

Description

Template for creating plots in the IOHanalyzer-style

Usage

```
IOH_plot_ly_default(title = NULL, x.title = NULL, y.title = NULL)
```

Arguments

title	Title for the plot
x.title	X-axis label
y.title	Y-axis label

Examples

```
IOH_plot_ly_default("Example plot","x-axis","y-axis")
```

`limit.data` *Reduce the size of the data set by evenly subsampling the records*

Description

Reduce the size of the data set by evenly subsampling the records

Usage

```
limit.data(df, n)
```

Arguments

`df`	The data to subsample
`n`	The amount of samples

Value

A smaller data.frame

`max_ERTs` *Get the ERT-values for all DataSets in a DataSetList at certain targets*

Description

Get the ERT-values for all DataSets in a DataSetList at certain targets

Usage

```
max_ERTs(dsList, aggr_on = "funcId", targets = NULL, maximize = T)

## S3 method for class 'DataSetList'
max_ERTs(dsList, aggr_on = "funcId", targets = NULL, maximize = T)
```

Arguments

`dsList`	The DataSetLsit
`aggr_on`	Whether to aggregate on 'funcId' or 'DIM'.
`targets`	Predifined target function-values. Should be one for each function/dimension
`maximize`	Whether the DataSetList is from a maximization or minimization problem

Value

A data.table containing ERT-values

Examples

```
max_ERTs(dsl)
```

`mean_FVs` *Get the expected function-values for all DataSets in a DataSetList at certain runtimes*

Description

Get the expected function-values for all DataSets in a DataSetList at certain runtimes

Usage

```
mean_FVs(dsList, aggr_on = "funcId", runtimes = NULL)

## S3 method for class 'DataSetList'
mean_FVs(dsList, aggr_on = "funcId", runtimes = NULL)
```

Arguments

`dsList`	The DataSetLsit
`aggr_on`	Whether to aggregate on 'funcId' or 'DIM'.
`runtimes`	Predifined target runtimes-values. Should be one for each function/dimension

Value

A data.table containing expected fucntion-values

Examples

```
mean_FVs(dsl)
```

`pairwise.test` *Performs a pairwise Kolmogorov-Smirnov test on the bootstrapped running times among a data set*

Description

This function performs a Kolmogorov-Smirnov test on each pair of algorithms in the input x to determine which algorithm gives a significantly smaller running time. The resulting p-values are arranged in a matrix, where each cell (i, j) contains a p-value from the test with alternative hypothesis: the running time of algorithm i is smaller (thus better) than that of j.

Usage

```
pairwise.test(x, ...)

## S3 method for class 'list'
pairwise.test(x, max_eval, bootstrap.size = 30, ...)

## S3 method for class 'DataSetList'
pairwise.test(x, ftarget, bootstrap.size = 0, which = "by_FV", ...)
```

Arguments

x	either a list that contains running time sample for each algorithm as sub-lists, or a DataSetList object
...	all other options
max_eval	list that contains the maximal running time for each algorithm as sub-lists
bootstrap.size	integer, the size of the bootstrapped sample. Set to 0 to disable bootstrapping
ftarget	float, the target value used to determine the running / hitting
which	wheter to do fixed-target ('by_FV') or fixed-budget ('by_RT') comparison time

Value

A matrix containing p-values of the test

Examples

```
pairwise.test(subset(dsl, funcId == 1), 16)
```

Plot.Comparison.Heatmap

Plot a heatmap according to the specifications from the Nevergrad dashboard

---

Description

Plot a heatmap according to the specifications from the Nevergrad dashboard

Usage

```
Plot.Comparison.Heatmap(dsList, target_dt, which = "by_FV")

## S3 method for class 'DataSetList'
Plot.Comparison.Heatmap(dsList, target_dt = NULL, which = "by_FV")
```

Arguments

dsList	A DataSetList (should consist of only one function and dimension).
target_dt	A data-table containing the targets to condider on each function/dimension pair
which	Whether to use fixed-target ('by_FV') or fixed-budget ('by_RT') perspective

Value

A heatmap showing the fraction of times algorithm A beats algorithm B

Examples

```
Plot.Comparison.Heatmap(dsl)
```

Plot.cumulative_difference_plot

Plot the cumulative difference plot given a DataSetList.

Description

Plot the cumulative difference plot given a DataSetList.

Usage

```
Plot.cumulative_difference_plot(
  dsList,
  runtime_or_target_value,
  isFixedBudget,
  alpha = 0.05,
  EPSILON = 1e-80,
  nOfBootstrapSamples = 1000,
  dataAlreadyComputed = FALSE,
  precomputedData = NULL
)
```

Arguments

dsList	A DataSetList (should consist of only one function and dimension and two algorithms).
runtime_or_target_value	The target runtime or the target value
isFixedBudget	Should be TRUE when target runtime is used. False otherwise.
alpha	1 minus the confidence level of the confidence band.
EPSILON	If abs(x-y) < EPSILON, then we assume that x = y.
nOfBootstrapSamples	The number of bootstrap samples used in the estimation.

dataAlreadyComputed
: If false, ‘generate_data.CDP‘ will be called to process the data.

precomputedData
: only needed when dataAlreadyComputed=TRUE. The result of ‘generate_data.CDP‘.

Value

A cumulative difference plot.

Examples

```
dsl
dsl_sub <- subset(dsl, funcId == 1)
target <- 15

Plot.cumulative_difference_plot(dsl_sub, target, FALSE)
```

Plot.FV.Aggregated *Plot expected function value-based comparison over multiple functions or dimensions*

Description

Plot expected function value-based comparison over multiple functions or dimensions

Usage

```
Plot.FV.Aggregated(
  dsList,
  aggr_on = "funcId",
  runtimes = NULL,
  plot_mode = "radar",
  use_rank = F,
  scale.ylog = T,
  fvs = NULL
)

## S3 method for class 'DataSetList'
Plot.FV.Aggregated(
  dsList,
  aggr_on = "funcId",
  runtimes = NULL,
  plot_mode = "radar",
  use_rank = F,
  scale.ylog = T,
  fvs = NULL
)
```

Arguments

dsList	A DataSetList (should consist of only one function OR dimension).
aggr_on	Whether to compare on functions ('funcId') or dimensions ('DIM')
runtimes	Custom list of function-value targets, one for each function or dimension.
plot_mode	How the plots should be created. Can be 'line' or 'radar'
use_rank	Wheter to use a ranking system. If False, the actual expected function- values will be used.
scale.ylog	Whether or not to scale the y-axis logaritmically
fvs	Pre-calculated expected function-values for the provided runtimes Created by the max_ERTs function of DataSetList. Can be provided to prevent needless computation in recalculating ERTs when recreating this plot.

Value

A plot of expected function value-based comparison on the provided functions or dimensions of the DataSetList

Examples

```
Plot.FV.Aggregated(dsl)
```

Plot.FV.ECDF_AUC *Radarplot of the area under the aggregated ECDF-curve of a DataSetList.*

Description

Radarplot of the area under the aggregated ECDF-curve of a DataSetList.

Usage

```
Plot.FV.ECDF_AUC(dsList, rt_min = NULL, rt_max = NULL, rt_step = NULL)

## S3 method for class 'DataSetList'
Plot.FV.ECDF_AUC(dsList, rt_min = NULL, rt_max = NULL, rt_step = NULL)
```

Arguments

dsList	A DataSetList (should consist of only one function and dimension).
rt_min	The starting runtime
rt_max	The final runtime
rt_step	The spacing between starting and final runtimes

Value

A radarplot of the area under the aggregated ECDF-curve of the DataSetList

Examples

```
Plot.FV.ECDF_AUC(subset(dsl, funcId == 1))
```

`Plot.FV.ECDF_Per_Target` *Plot the empirical cumulative distriburtion as a function of the target values of a DataSetList at certain target runtimes*

Description

Plot the empirical cumulative distriburtion as a function of the target values of a DataSetList at certain target runtimes

Usage

```
Plot.FV.ECDF_Per_Target(dsList, runtimes, scale.xlog = F, scale.reverse = F)

## S3 method for class 'DataSetList'
Plot.FV.ECDF_Per_Target(dsList, runtimes, scale.xlog = F, scale.reverse = F)
```

Arguments

`dsList`	A DataSetList (should consist of only one function and dimension).
`runtimes`	The target runtimes
`scale.xlog`	Whether or not to scale the x-axis logaritmically
`scale.reverse`	Whether or not to reverse the x-axis (when using minimization)

Value

A plot of the empirical cumulative distriburtion as a function of the fucntion values of the DataSetList at the target runtimes

Examples

```
Plot.FV.ECDF_Per_Target(subset(dsl, funcId == 1), 10)
```

`Plot.FV.ECDF_Single_Func` *Plot the aggregated empirical cumulative distriburtion as a function of the function values of a DataSetList.*

Description

Plot the aggregated empirical cumulative distriburtion as a function of the function values of a DataSetList.

Usage

```
Plot.FV.ECDF_Single_Func(
  dsList,
  rt_min = NULL,
  rt_max = NULL,
  rt_step = NULL,
  scale.xlog = F,
  show.per_target = F,
  scale.reverse = F
)

## S3 method for class 'DataSetList'
Plot.FV.ECDF_Single_Func(
  dsList,
  rt_min = NULL,
  rt_max = NULL,
  rt_step = NULL,
  scale.xlog = F,
  show.per_target = F,
  scale.reverse = F
)
```

Arguments

`dsList`	A DataSetList (should consist of only one function and dimension).
`rt_min`	The starting runtime
`rt_max`	The final runtime
`rt_step`	The spacing between starting and final runtimes
`scale.xlog`	Whether or not to scale the x-axis logaritmically
`show.per_target`	Whether or not to show the individual ECDF-curves for each runtime
`scale.reverse`	Whether or not to reverse the x-axis (when using minimization)

Value

A plot of the empirical cumulative distriburtion as a function of the function values of the DataSetList

Examples

```
Plot.FV.ECDF_Single_Func(subset(dsl, funcId == 1))
```

`Plot.FV.Histogram` *Plot histograms of the function values of a DataSetList at a certain target runtime*

Description

Plot histograms of the function values of a DataSetList at a certain target runtime

Usage

```
Plot.FV.Histogram(dsList, runtime, plot_mode = "overlay", use.equal.bins = F)

## S3 method for class 'DataSetList'
Plot.FV.Histogram(dsList, runtime, plot_mode = "overlay", use.equal.bins = F)
```

Arguments

`dsList`	A DataSetList (should consist of only one function and dimension).
`runtime`	The target runtime
`plot_mode`	How to plot the different hisograms for each algorithm. Can be either 'overlay' to show all algorithms on one plot, or 'subplot' to have one plot per algorithm.
`use.equal.bins`	Whether to determine one bin size for all plots or have individual bin sizes for each algorithm

Value

A plot of the histograms of the function values at a the target runtime of the DataSetList

Examples

```
Plot.FV.Histogram(subset(dsl, funcId == 1), 100)
```

Plot.FV.Multi_Func *Plot FV-plots for multiple functions or dimensions*

Description

Plot FV-plots for multiple functions or dimensions

Usage

```
Plot.FV.Multi_Func(dsList, scale.xlog = F, scale.ylog = F, backend = NULL)

## S3 method for class 'DataSetList'
Plot.FV.Multi_Func(dsList, scale.xlog = F, scale.ylog = F, backend = NULL)
```

Arguments

dsList	A DataSetList (should consist of only one function OR dimension).
scale.xlog	Whether or not to scale the x-axis logaritmically
scale.ylog	Whether or not to scale the y-axis logaritmically
backend	Which plotting library to use. Either 'plotly' or 'ggplot2'.

Value

A plot of Function-values of the DataSetList

Examples

```
Plot.FV.Multi_Func(dsl)
```

Plot.FV.Parameters *Plot the parameter values recorded in a DataSetList (aligned by budget)*

Description

Plot the parameter values recorded in a DataSetList (aligned by budget)

Usage

```
Plot.FV.Parameters(
  dsList,
  rt_min = NULL,
  rt_max = NULL,
  algids = "all",
  par_name = NULL,
  scale.xlog = F,
  scale.ylog = F,
  show.mean = T,
  show.median = F,
  show.CI = F
)

## S3 method for class 'DataSetList'
Plot.FV.Parameters(
  dsList,
  rt_min = NULL,
  rt_max = NULL,
  algids = "all",
  par_name = NULL,
  scale.xlog = F,
  scale.ylog = F,
  show.mean = T,
  show.median = F,
  show.CI = F
)
```

Arguments

dsList	A DataSetList (should consist of only one function and dimension).
rt_min	The starting budget value.
rt_max	The final budget value.
algids	Which algorithms from dsList to use
par_name	Which parameters to create plots for; set to NULL to use all parameters found in dsList.
scale.xlog	Whether or not to scale the x-axis logaritmically
scale.ylog	Whether or not to scale the y-axis logaritmically
show.mean	Whether or not to show the mean parameter values
show.median	Whether or not to show the median parameter values
show.CI	Whether or not to show the standard deviation

Value

A plot of for every recorded parameter in the DataSetList

Examples

```
Plot.FV.Parameters(subset(dsl, funcId == 1))
```

`Plot.FV.PDF` *Plot probablity density function of the function values of a DataSetList at a certain target runtime*

Description

Plot probablity density function of the function values of a DataSetList at a certain target runtime

Usage

```
Plot.FV.PDF(dsList, runtime, show.sample = F, scale.ylog = F)

## S3 method for class 'DataSetList'
Plot.FV.PDF(dsList, runtime, show.sample = F, scale.ylog = F)
```

Arguments

`dsList`	A DataSetList (should consist of only one function and dimension).
`runtime`	The target runtime
`show.sample`	Whether or not to show the individual function value samples
`scale.ylog`	Whether or not to scale the y-axis logaritmically

Value

A plot of the probablity density function of the runtimes at a the target function value of the DataSetList

Examples

```
Plot.FV.PDF(subset(dsl, funcId == 1), 100)
```

`Plot.FV.Single_Func` *Plot lineplot of the expected function values of a DataSetList*

Description

Plot lineplot of the expected function values of a DataSetList

Usage

```
Plot.FV.Single_Func(
  dsList,
  RTstart = NULL,
  RTstop = NULL,
  show.CI = F,
  show.mean = T,
  show.median = F,
  backend = NULL,
  scale.xlog = F,
  scale.ylog = F,
  scale.reverse = F
)

## S3 method for class 'DataSetList'
Plot.FV.Single_Func(
  dsList,
  RTstart = NULL,
  RTstop = NULL,
  show.CI = F,
  show.mean = T,
  show.median = F,
  backend = NULL,
  scale.xlog = F,
  scale.ylog = F,
  scale.reverse = F
)
```

Arguments

`dsList`	A DataSetList (should consist of only one function and dimension).
`RTstart`	The starting runtime value.
`RTstop`	The final runtime value.
`show.CI`	Whether or not to show the standard deviations
`show.mean`	Whether or not to show the mean runtimes
`show.median`	Whether or not to show the median runtimes
`backend`	Which plotting library to use. Can be 'plotly' or 'ggplot2'

scale.xlog	Whether or not to scale the x-axis logaritmically
scale.ylog	Whether or not to scale the y-axis logaritmically
scale.reverse	Wheter or not to reverse the x-axis (when using minimization)

Value

A plot of ERT-values of the DataSetList

Examples

```
Plot.FV.Single_Func(subset(dsl, funcId == 1))
```

Plot.Performviz *Create the PerformViz plot*

Description

From the paper:

Usage

```
Plot.Performviz(DSC_rank_result)
```

Arguments

DSC_rank_result	The result from a call to DSCtool rank service (‘get_dsc_rank‘)

Value

A performviz plot

Examples

```
## Not run:
Plot.Performviz(get_dsc_rank(dsl))

## End(Not run)
```

`Plot.RT.Aggregated` *Plot ERT-based comparison over multiple functions or dimensions*

Description

Plot ERT-based comparison over multiple functions or dimensions

Usage

```
Plot.RT.Aggregated(
  dsList,
  aggr_on = "funcId",
  targets = NULL,
  plot_mode = "radar",
  use_rank = F,
  scale.ylog = T,
  maximize = T,
  erts = NULL,
  inf.action = "overlap"
)

## S3 method for class 'DataSetList'
Plot.RT.Aggregated(
  dsList,
  aggr_on = "funcId",
  targets = NULL,
  plot_mode = "radar",
  use_rank = F,
  scale.ylog = T,
  maximize = T,
  erts = NULL,
  inf.action = "overlap"
)
```

Arguments

`dsList`	A DataSetList (should consist of only one function OR dimension).
`aggr_on`	Whether to compare on functions ('funcId') or dimensions ('DIM')
`targets`	Custom list of function-value targets, one for each function or dimension.
`plot_mode`	How the plots should be created. Can be 'line' or 'radar'
`use_rank`	Wheter to use a ranking system. If False, the actual ERT-values will be used.
`scale.ylog`	Whether or not to scale the y-axis logaritmically
`maximize`	Wheter or not to the data is of a maximization problem
`erts`	Pre-calculated ERT-values for the provided targets. Created by the max_ERTs function of DataSetList. Can be provided to prevent needless computation in recalculating ERTs when recreating this plot.

inf.action	How to handle infinite ERTs ('overlap' or 'jitter')

Value

A plot of ERT-based comparison on the provided functions or dimensions of the DataSetList

Examples

```
Plot.RT.Aggregated(dsl)
```

Plot.RT.ECDF_AUC *Radarplot of the area under the aggregated ECDF-curve of a DataSetList.*

Description

Radarplot of the area under the aggregated ECDF-curve of a DataSetList.

Usage

```
Plot.RT.ECDF_AUC(
  dsList,
  fstart = NULL,
  fstop = NULL,
  fstep = NULL,
  fval_formatter = as.integer
)

## S3 method for class 'DataSetList'
Plot.RT.ECDF_AUC(
  dsList,
  fstart = NULL,
  fstop = NULL,
  fstep = NULL,
  fval_formatter = as.integer
)
```

Arguments

dsList	A DataSetList (should consist of only one function and dimension).
fstart	The starting function value
fstop	The final function value
fstep	The spacing between starting and final function values
fval_formatter	Function to format the function-value labels

Value

A radarplot of the area under the aggregated ECDF-curve of the DataSetList

Examples

```
Plot.RT.ECDF_AUC(subset(dsl, funcId == 1))
```

`Plot.RT.ECDF_Multi_Func` *Plot the aggregated empirical cumulative distriburtion as a function of the running times of a DataSetList. Aggregated over multiple functions or dimensions.*

Description

Plot the aggregated empirical cumulative distriburtion as a function of the running times of a DataSetList. Aggregated over multiple functions or dimensions.

Usage

```
Plot.RT.ECDF_Multi_Func(dsList, targets = NULL, scale.xlog = F)

## S3 method for class 'DataSetList'
Plot.RT.ECDF_Multi_Func(dsList, targets = NULL, scale.xlog = F)
```

Arguments

`dsList`	A DataSetList.
`targets`	The target function values. Specified in a data.frame, as can be generated
`scale.xlog`	Whether or not to scale the x-axis logaritmically by the function 'get_ECDF_targets'

Value

A plot of the empirical cumulative distriburtion as a function of the running times of the DataSetList

Examples

```
Plot.RT.ECDF_Multi_Func(dsl)
```

Plot.RT.ECDF_Per_Target

Plot the empirical cumulative distriburtion as a function of the running times of a DataSetList at certain target function values

Description

Plot the empirical cumulative distriburtion as a function of the running times of a DataSetList at certain target function values

Usage

```
Plot.RT.ECDF_Per_Target(dsList, ftargets, scale.xlog = F)

## S3 method for class 'DataSetList'
Plot.RT.ECDF_Per_Target(dsList, ftargets, scale.xlog = F)
```

Arguments

dsList	A DataSetList (should consist of only one function and dimension).
ftargets	The target function values
scale.xlog	Whether or not to scale the x-axis logaritmically

Value

A plot of the empirical cumulative distriburtion as a function of the running times of the DataSetList at the target function values

Examples

```
Plot.RT.ECDF_Per_Target(subset(dsl, funcId == 1), 14)
```

Plot.RT.ECDF_Single_Func

Plot the aggregated empirical cumulative distriburtion as a function of the running times of a DataSetList.

Description

Plot the aggregated empirical cumulative distriburtion as a function of the running times of a DataSetList.

Usage

```
Plot.RT.ECDF_Single_Func(
  dsList,
  fstart = NULL,
  fstop = NULL,
  fstep = NULL,
  show.per_target = F,
  scale.xlog = F
)

## S3 method for class 'DataSetList'
Plot.RT.ECDF_Single_Func(
  dsList,
  fstart = NULL,
  fstop = NULL,
  fstep = NULL,
  show.per_target = F,
  scale.xlog = F
)
```

Arguments

dsList	A DataSetList (should consist of only one function and dimension).
fstart	The starting function value
fstop	The final function value
fstep	The spacing between starting and final function values
show.per_target	Whether or not to show the individual ECDF-curves for each target
scale.xlog	Whether or not to scale the x-axis logaritmically

Value

A plot of the empirical cumulative distriburtion as a function of the running times of the DataSetList

Examples

```
Plot.RT.ECDF_Single_Func(subset(dsl, funcId == 1))
```

Plot.RT.Histogram *Plot histograms of the runtimes of a DataSetList at a certain target function value*

Description

Plot histograms of the runtimes of a DataSetList at a certain target function value

Usage

```
Plot.RT.Histogram(dsList, ftarget, plot_mode = "overlay", use.equal.bins = F)

## S3 method for class 'DataSetList'
Plot.RT.Histogram(dsList, ftarget, plot_mode = "overlay", use.equal.bins = F)
```

Arguments

dsList	A DataSetList (should consist of only one function and dimension).
ftarget	The target function value.
plot_mode	How to plot the different hisograms for each algorithm. Can be either 'overlay' to show all algorithms on one plot, or 'subplot' to have one plot per algorithm.
use.equal.bins	Whether to determine one bin size for all plots or have individual bin sizes for each algorithm

Value

A plot of the histograms of the runtimes at a the target function value of the DataSetList

Examples

```
Plot.RT.Histogram(subset(dsl, funcId == 1), 14)
```

Plot.RT.Multi_Func *Plot ERT-plots for multiple functions or dimensions*

Description

Plot ERT-plots for multiple functions or dimensions

Usage

```
Plot.RT.Multi_Func(
  dsList,
  scale.xlog = F,
  scale.ylog = F,
  scale.reverse = F,
  backend = NULL
)

## S3 method for class 'DataSetList'
Plot.RT.Multi_Func(
  dsList,
  scale.xlog = F,
  scale.ylog = F,
  scale.reverse = F,
  backend = NULL
)
```

Arguments

dsList	A DataSetList (should consist of only one function OR dimension).
scale.xlog	Whether or not to scale the x-axis logaritmically
scale.ylog	Whether or not to scale the y-axis logaritmically
scale.reverse	Wheter or not to reverse the x-axis (when using minimization)
backend	Which plotting library to use. Either 'plotly' or 'ggplot2'.

Value

A plot of ERT-values of the DataSetList

Examples

```
Plot.RT.Multi_Func(dsl)
```

Plot.RT.Parameters *Plot the parameter values recorded in a DataSetList (aligned by funcion value)*

Description

Plot the parameter values recorded in a DataSetList (aligned by funcion value)

Usage

```
Plot.RT.Parameters(
  dsList,
  f_min = NULL,
  f_max = NULL,
  algids = "all",
  par_name = NULL,
  scale.xlog = F,
  scale.ylog = F,
  show.mean = T,
  show.median = F,
  show.CI = F
)

## S3 method for class 'DataSetList'
Plot.RT.Parameters(
  dsList,
  f_min = NULL,
  f_max = NULL,
  algids = "all",
  par_name = NULL,
  scale.xlog = F,
```

```
  scale.ylog = F,
  show.mean = T,
  show.median = F,
  show.CI = F
)
```

Arguments

dsList	A DataSetList (should consist of only one function and dimension).
f_min	The starting function value.
f_max	The final function value.
algids	Which algorithms from dsList to use
par_name	Which parameters to create plots for; set to NULL to use all parameters found in dsList.
scale.xlog	Whether or not to scale the x-axis logaritmically
scale.ylog	Whether or not to scale the y-axis logaritmically
show.mean	Whether or not to show the mean parameter values
show.median	Whether or not to show the median parameter values
show.CI	Whether or not to show the standard deviation

Value

A plot of for every recorded parameter in the DataSetList

Examples

```
Plot.RT.Parameters(subset(dsl, funcId == 1))
```

Plot.RT.PMF *Plot probablity mass function of the runtimes of a DataSetList at a certain target function value*

Description

Plot probablity mass function of the runtimes of a DataSetList at a certain target function value

Usage

```
Plot.RT.PMF(dsList, ftarget, show.sample = F, scale.ylog = F, backend = NULL)

## S3 method for class 'DataSetList'
Plot.RT.PMF(dsList, ftarget, show.sample = F, scale.ylog = F, backend = NULL)
```

Arguments

dsList	A DataSetList (should consist of only one function and dimension).
ftarget	The target function value.
show.sample	Whether or not to show the individual runtime samples
scale.ylog	Whether or not to scale the y-axis logaritmically
backend	Which plotting library to use. Can be 'plotly' or 'ggplot2'

Value

A plot of the probablity mass function of the runtimes at a the target function value of the DataSetList

Examples

```
Plot.RT.PMF(subset(dsl, funcId == 1), 14)
```

Plot.RT.Single_Func *Plot lineplot of the ERTs of a DataSetList*

Description

Plot lineplot of the ERTs of a DataSetList

Usage

```
Plot.RT.Single_Func(
  dsList,
  Fstart = NULL,
  Fstop = NULL,
  show.ERT = T,
  show.CI = F,
  show.mean = F,
  show.median = F,
  backend = NULL,
  scale.xlog = F,
  scale.ylog = F,
  scale.reverse = F,
  includeOpts = F,
  p = NULL
)

## S3 method for class 'DataSetList'
Plot.RT.Single_Func(
  dsList,
  Fstart = NULL,
  Fstop = NULL,
  show.ERT = T,
```

```
  show.CI = T,
  show.mean = F,
  show.median = F,
  backend = NULL,
  scale.xlog = F,
  scale.ylog = F,
  scale.reverse = F,
  includeOpts = F,
  p = NULL
)
```

Arguments

dsList	A DataSetList (should consist of only one function and dimension).
Fstart	The starting function value.
Fstop	The final function value.
show.ERT	Whether or not to show the ERT-values
show.CI	Whether or not to show the standard deviations
show.mean	Whether or not to show the mean hitting times
show.median	Whether or not to show the median hitting times
backend	Which plotting library to use. Can be 'plotly' or 'ggplot2'
scale.xlog	Whether or not to scale the x-axis logaritmically
scale.ylog	Whether or not to scale the y-axis logaritmically
scale.reverse	Wheter or not to reverse the x-axis (when using minimization)
includeOpts	Whether or not to include all best points reached by each algorithm
p	Existing plot to which to add the current data

Value

A plot of ERT-values of the DataSetList

Examples

```
Plot.RT.Single_Func(subset(dsl, funcId == 1))
```

Plot.Stats.Glicko2_Candlestick

Create a candlestick plot of Glicko2-rankings

Description

Create a candlestick plot of Glicko2-rankings

Usage

```
Plot.Stats.Glicko2_Candlestick(
  dsList,
  nr_rounds = 100,
  glicko2_rank_df = NULL,
  which = "by_FV",
  target_dt = NULL
)

## S3 method for class 'DataSetList'
Plot.Stats.Glicko2_Candlestick(
  dsList,
  nr_rounds = 100,
  glicko2_rank_df = NULL,
  which = "by_FV",
  target_dt = NULL
)
```

Arguments

dsList	A DataSetList
nr_rounds	The number of rounds in the tournament
glicko2_rank_df	Optional. Dataframe containing the glicko2 rating to avoid needless recalculation.
which	Whether to use fixed-target ('by_FV') or fixed-budget ('by_RT') perspective
target_dt	Optional: data table containing the targets for each function and dimension

Examples

```
Plot.Stats.Glicko2_Candlestick(dsl, nr_rounds=2)
```

Plot.Stats.Significance_Graph

Plot a network graph showing the statistically different algorithms

---

Description

Plot a network graph showing the statistically different algorithms

Usage

```
Plot.Stats.Significance_Graph(
  dsList,
  ftarget,
  alpha = 0.01,
```

```
  bootstrap.size = 30,
  which = "by_FV"
)

## S3 method for class 'DataSetList'
Plot.Stats.Significance_Graph(
  dsList,
  ftarget,
  alpha = 0.01,
  bootstrap.size = 30,
  which = "by_FV"
)
```

Arguments

dsList	A DataSetList (should consist of only one function and dimension).
ftarget	The target function value to use
alpha	The cutoff for statistical significance
bootstrap.size	The amound of bootstrapped samples used
which	Whether to use fixed-target ('by_FV') or fixed-budget ('by_RT') perspective

Value

A graph showing the statistical significance between algorithms

Examples

```
Plot.Stats.Significance_Graph(subset(dsl, funcId == 2), 16)
```

Plot.Stats.Significance_Heatmap

Plot a heatmap showing the statistically different algorithms

---

Description

Plot a heatmap showing the statistically different algorithms

Usage

```
Plot.Stats.Significance_Heatmap(
  dsList,
  ftarget,
  alpha = 0.01,
  bootstrap.size = 30,
  which = "by_FV"
)
```

```
## S3 method for class 'DataSetList'
Plot.Stats.Significance_Heatmap(
  dsList,
  ftarget,
  alpha = 0.01,
  bootstrap.size = 30,
  which = "by_FV"
)
```

Arguments

dsList	A DataSetList (should consist of only one function and dimension).
ftarget	The target function value to use
alpha	The cutoff for statistical significance
bootstrap.size	The amound of bootstrapped samples used
which	Whether to use fixed-target ('by_FV') or fixed-budget ('by_RT') perspective

Value

A heatmap showing the statistical significance between algorithms

Examples

```
Plot.Stats.Significance_Heatmap(subset(dsl, funcId == 2), 16)
```

---

plot_eaf_data *Create EAF-based polygon plots*

Description

Create EAF-based polygon plots

Usage

```
plot_eaf_data(
  df,
  maximization = F,
  scale.xlog = F,
  scale.ylog = F,
  scale.reverse = F,
  p = NULL,
  x_title = NULL,
  xmin = NULL,
  xmax = NULL,
  ymin = NULL,
```

```
  ymax = NULL,
  y_title = NULL,
  plot_title = NULL,
  subplot_attr = NULL,
  show.colorbar = F,
  subplot_shareX = F,
  dt_overlay = NULL,
  ...
)
```

Arguments

df	The dataframe containing the data to plot. This should come from 'generate_data.EAF'
maximization	Whether the data comes from maximization or minimization
scale.xlog	Logarithmic scaling of x-axis
scale.ylog	Logarithmic scaling of y-axis
scale.reverse	Decreasing or increasing x-axis
p	A previously existing plot on which to add traces. If NULL, a new canvas is created
x_title	Title of x-axis. Defaults to x_attr
xmin	Minimum value for the x-axis
xmax	Maximum value for the x-axis
ymin	Minimum value for the y-axis
ymax	Maximum value for the y-axis
y_title	Title of x-axis. Defaults to x_attr
plot_title	Title of x-axis. Defaults to no title
subplot_attr	Which attribute of the dataframe to use for creating subplots
show.colorbar	Whether or not to include a colorbar
subplot_shareX	Whether or not to share X-axis when using subplots
dt_overlay	Dataframe containing additional data (e.g. quantiles) to plot on top of the EAF. This should have a column labeled 'runtime'. The other columsn will all be plotted as function values.
...	Additional parameters for the add_trace function

Value

An EAF plot

Examples

```
## Not run:
plot_eaf_data(generate_data.EAF(subset(dsl, ID==get_id(dsl)[[1]])), maximization=T)

## End(Not run)
```

plot_eaf_differences *Create EAF-difference contour plots*

Description

Create EAF-difference contour plots

Usage

```
plot_eaf_differences(
  matrices,
  scale.xlog = T,
  scale.ylog = F,
  zero_transparant = F,
  show_negatives = F
)
```

Arguments

matrices	The dataframes containing the data to plot. This should come from 'generate_data.EAF_diff_Approximate'
scale.xlog	Logarithmic scaling of x-axis
scale.ylog	Logarithmic scaling of y-axis
zero_transparant	Whether values of 0 should be made transparant or not
show_negatives	Whether to also show negative values or not

Value

EAF difference plots

Examples

```
## Not run:
plot_eaf_differences(generate_data.EAF_diff_Approximate(subset(dsl, funcId == 1), 1, 50, 1, 16))

## End(Not run)
```

plot_general_data *General function for plotting within IOHanalyzer*

Description

General function for plotting within IOHanalyzer

Usage

```
plot_general_data(
  df,
  x_attr = "ID",
  y_attr = "vals",
  type = "violin",
  legend_attr = "ID",
  scale.xlog = F,
  scale.ylog = F,
  scale.reverse = F,
  p = NULL,
  x_title = NULL,
  y_title = NULL,
  plot_title = NULL,
  upper_attr = NULL,
  lower_attr = NULL,
  subplot_attr = NULL,
  show.legend = F,
  inf.action = "none",
  violin.showpoints = F,
  frame_attr = "frame",
  symbol_attr = "run_nr",
  subplot_shareX = F,
  line.step = F,
  ...
)
```

Arguments

df	The dataframe containing the data to plot. It should contain at least two columns: 'x_attr' and 'y_attr'
x_attr	The column to specify the x_axis. Default is 'algId'
y_attr	The column to specify the y_axis
type	The type of plot to use. Currently available: 'violin', 'line', 'radar', 'bar', hist' and 'ribbon'
legend_attr	Default is 'algId' This is also used for the selection of colorschemes
scale.xlog	Logarithmic scaling of x-axis

scale.ylog	Logarithmic scaling of y-axis
scale.reverse	Decreasing or increasing x-axis
p	A previously existing plot on which to add traces. If NULL, a new canvas is created
x_title	Title of x-axis. Defaults to x_attr
y_title	Title of x-axis. Defaults to x_attr
plot_title	Title of x-axis. Defaults to no title
upper_attr	When using ribbon-plot, this can be used to create a shaded area. Only works in combination with‘lower_attr‘ and ‘type‘ == ’ribbon’
lower_attr	When using ribbon-plot, this can be used to create a shaded area. Only works in combination with‘upper_attr‘ and ‘type‘ == ’ribbon’
subplot_attr	Which attribute of the dataframe to use for creating subplots
show.legend	Whether or not to include a legend
inf.action	How to deal with infinite values. Can be ’none’, ’overlap’ or ’jitter’
violin.showpoints	Wheteher or not to show individual points when making a violinplot
frame_attr	Which attribute of the dataframe to use for the time element of the animation
symbol_attr	Which attribute of the dataframe to use for the scatter symbol
subplot_shareX	Whether or not to share X-axis when using subplots
line.step	Whether to plot lines as a step-function (T) or as linear interpolation (F, default)
...	Additional parameters for the add_trace function

`print.DataSet` *S3 generic print operator for DataSet*

Description

S3 generic print operator for DataSet

Usage

```
## S3 method for class 'DataSet'
print(x, ...)
```

Arguments

x	A DataSet object
...	Arguments passed to other methods

Value

A short description of the DataSet

Examples

```
print(dsl[[1]])
```

print.DataSetList *S3 print function for DataSetList*

Description

S3 print function for DataSetList

Usage

```
## S3 method for class 'DataSetList'
print(x, ...)
```

Arguments

x	The DataSetList to print
...	Arguments for underlying print function?

Examples

```
print(dsl)
```

read_index_file *Read .info files and extract information*

Description

Read .info files and extract information

Usage

```
read_index_file(fname)
```

Arguments

fname	The path to the .info file

Value

The data contained in the .info file

Examples

```
path <- system.file("extdata", "ONE_PLUS_LAMDA_EA", package="IOHanalyzer")
info <- read_index_file(file.path(path,"IOHprofiler_f1_i1.info"))
```

read_IOH_v1plus *Read Nevergrad data*

Description

Read .csv files in arbitrary format

Usage

```
read_IOH_v1plus(info, full_sampling = FALSE)
```

Arguments

info	A List containing all meta-data about the dataset to create
full_sampling	Logical. Whether the raw (unaligned) FV matrix should be stored. Currently only useful when a correlation plot between function values and parameters should be made

Value

The DataSetList extracted from the .csv file provided

read_pure_csv *Read Nevergrad data*

Description

Read .csv files in arbitrary format

Usage

```
read_pure_csv(
  path,
  neval_name,
  fval_name,
  fname_name,
  algname_name,
  dim_name,
  run_name,
  maximization = F,
  static_attrs = NULL
)
```

Arguments

path	The path to the .csv file
neval_name	The name of the column to use for the evaluation count. If NULL, will be assumed to be sequential
fval_name	The name of the column to use for the function values
fname_name	The name of the column to use for the function name
algname_name	The name of the column to use for the algorithm name
dim_name	The name of the column to use for the dimension
run_name	The name of the column to use for the run number
maximization	Boolean indicating whether the data is resulting from maximization or minimization
static_attrs	Named list containing the static values for missing columns. When a parameter is not present in the csv file, its name-parameter should be set to NULL, and the static value should be added to this static_attrs list.

Value

The DataSetList extracted from the .csv file provided

---

register_DSC *Register an account to the DSCtool API*

Description

This uses the keyring package to store and load credentials. If you already have an account, please call ‘set_DSC_credentials‘ instead

Usage

```
register_DSC(name, username, affiliation, email, password = NULL)
```

Arguments

name	Your name
username	A usename to be identified with. Will be stored on keyring under 'DSCtool_name'
affiliation	Your affiliation (university / company)
email	Your email adress
password	The password to use. If NULL, this will be generated at random. Will be stored on keyring under 'DSCtool'

Examples

```
## Not run:
register_DSC('John Doe', 'jdoe', 'Sample University', "j.doe.sample.com")

## End(Not run)
```

runServer *Create a shiny-server GUI to interactively use the IOHanalyzer*

Description

Create a shiny-server GUI to interactively use the IOHanalyzer

Usage

```
runServer(port = getOption("shiny.port"), open_browser = TRUE, orca_gpu = TRUE)
```

Arguments

port	Optional; which port the server should be opened at. Defaults to the option set for 'shiny.port'
open_browser	Whether or not to open a browser tab with the IOHanalyzer GUI. Defaults to TRUE.
orca_gpu	Whether or not orca will be allowed to use gpu-accelleration for saving figures to file.

Examples

```
## Not run:
runServer(6563, TRUE)

## End(Not run)
```

save_plotly *Save plotly figure in multiple format*

Description

NOTE: This function requires orca to be installed

Usage

```
save_plotly(p, file, width = NULL, height = NULL, ...)
```

Arguments

p	plotly object. The plot to be saved
file	String. The name of the figure file, with the extension of the required file-format
width	Optional. Width of the figure
height	Optional. Height of the figure
...	Additional arguments for orca

Examples

```
## Not run:
p <- Plot.RT.Single_Func(dsl[1])
save_plotly(p, 'example_file.png')

## End(Not run)
```

`save_table` *Save DataTable in multiple formats*

Description

Save DataTable in multiple formats

Usage

```
save_table(df, file, format = NULL)
```

Arguments

`df`	The DataTable to store
`file`	String. The name of the figure file, with the extension of the required file-format
`format`	Optional, string. Overwrites the extension of the ‘file‘ parameter. If not specified while file does not have an extension, it defaults to csv

Examples

```
df <- generate_data.Single_Function(subset(dsl, funcId == 1), which = 'by_RT')
save_table(df, tempfile(fileext = ".md"))
```

`scan_index_file` *Scan *.info files for IOHProfiler or COCO*

Description

Scan *.info files for IOHProfiler or COCO

Usage

```
scan_index_file(folder)
```

Arguments

`folder`	The folder containing the .info or .json files

Value

The paths to all found .info and .json-files

Note

This automatically filetrs our files of size 0

Examples

```
path <- system.file("extdata", "ONE_PLUS_LAMDA_EA", package="IOHanalyzer")
scan_index_file(path)
```

---

seq_FV *Function for generating sequences of function values*

Description

Function for generating sequences of function values

Usage

```
seq_FV(
  FV,
  from = NULL,
  to = NULL,
  by = NULL,
  length.out = NULL,
  scale = NULL,
  force_limits = FALSE
)
```

Arguments

FV	A list of function values
from	Starting function value. Will be replaced by min(FV) if it is NULL or too small
to	Stopping function value. Will be replaced by max(FV) if it is NULL or too large
by	Stepsize of the sequence. Will be replaced if it is too small
length.out	Number of values in the sequence. 'by' takes preference if both it and length.out are provided.
scale	Scaling of the sequence. Can be either 'linear' or 'log', indicating a linear or log-linear spacing respectively. If NULL, the scale will be predicted based on FV
force_limits	Whether the from and to values are hard, or should be modified based on detected FV values (default False)

Value

A sequence of function values

Examples

```
FVall <- get_runtimes(dsl)
seq_FV(FVall, 10, 16, 1, scale='linear')
```

`seq_RT` *Function for generating sequences of runtime values*

Description

Function for generating sequences of runtime values

Usage

```
seq_RT(
  RT,
  from = NULL,
  to = NULL,
  by = NULL,
  length.out = NULL,
  scale = "linear"
)
```

Arguments

`RT`	A list of runtime values
`from`	Starting runtime value. Will be replaced by min(RT) if it is NULL or too small
`to`	Stopping runtime value. Will be replaced by max(RT) if it is NULL or too large
`by`	Stepsize of the sequence. Will be replaced if it is too small
`length.out`	Number of values in the sequence. 'by' takes preference if both it and length.out are provided.
`scale`	Scaling of the sequence. Can be either 'linear' or 'log', indicating a linear or log-linear spacing respectively.

Value

A sequence of runtime values

Examples

```
RTall <- get_runtimes(dsl)
seq_RT(RTall, 0, 500, length.out=10, scale='log')
```

set_color_scheme *Set the colorScheme of the IOHanalyzer plots*

Description

Set the colorScheme of the IOHanalyzer plots

Usage

```
set_color_scheme(schemename, ids, path = NULL)
```

Arguments

schemename	Three default colorschemes are implemented: • Default • Variant 1 • Variant 2 • Variant 3 And it is also possible to select "Custom", which allows uploading of a custom set of colors
ids	The names of the algorithms (or custom ids, see ‘change_id‘) for which to set the colors
path	The path to the file containing the colors to use. Only used if schemename is "Custom"

Examples

```
set_color_scheme("Default", get_algId(dsl))
```

set_DSC_credentials *Register an account to the DSCtool API*

Description

This uses the keyring package to store and load credentials. If you already have an account, please call ‘add_DSC_credentials‘ instead

Usage

```
set_DSC_credentials(username, password)
```

Arguments

username	The usename you use on DSCtool. Will be stored on keyring under 'DSC-tool_name'
password	The password you use on DSCtool. Will be stored on keyring under 'DSCtool'

Examples

```
## Not run: set_DSC_credentials('jdoe', 'monkey123')
```

SP *Estimator 'SP' for the Expected Running Time (ERT)*

Description

Estimator 'SP' for the Expected Running Time (ERT)

Usage

```
SP(data, max_runtime)
```

Arguments

data	A dataframe or matrix. Each row stores the runtime sample points from several runs
max_runtime	The budget to use for calculating ERT. If this is a vector, the largest value is taken. Using this as a vector is being deprecated, and will be removed in a future update

Value

A list containing ERTs, number of succesfull runs and the succes rate

Examples

```
SP(dsl[[1]]$RT, max(dsl[[1]]$RT))
```

subset.DataSet *S3 subset function for DataSet*

Description

Subset for DataSets. Based on the provided mask, the relevant data is taken from the given DataSet and turned into a new DataSet object.

Usage

```
## S3 method for class 'DataSet'
subset(x, mask, ...)
```

Arguments

x	The DataSet from which to get a subset
mask	The mask (as boolean list) to use when subsetting. The length should be equal to the number of runs present in the provided dataset object x.
...	Arguments passed to underlying subset method (not yet supported)

Value

A new DataSet

Examples

```
subset(dsl[[1]], c(0,1,1,1,0,0,0,0,0,0,0))
```

subset.DataSetList *Filter a DataSetList by some criteria*

Description

Filter a DataSetList by some criteria

Usage

```
## S3 method for class 'DataSetList'
subset(x, ...)
```

Arguments

x	The DataSetList
...	The conditions to filter on. Can be any expression which assigns True or False to a DataSet object, such as DIM == 625 or funcId == 2. Usage of && and || is only supported on default attributes (funcId, algId, DIM), not on combinations of with other attributes (e.g. instance). In those cases, & and | should be used respectively. Alternatively, this can be used as a keyword argument named 'text', with the condition as a string to be parsed. This allows exectution of subset commands on arbitrary variables in code.

Value

The filtered DataSetList

Examples

```
subset(dsl, funcId == 1)
subset(dsl, funcId == 1 && DIM == 16) # Can use && and || for default attributes
subset(dsl, instance == 1)
subset(dsl, instance == 1 & funcId == 1) # Can use & and | for all attributes
subset(dsl, instance == 1, funcId == 1) # Comma-seperated conditions are treated as AND
```

---

summary.DataSet *S3 generic summary operator for DataSet*

Description

S3 generic summary operator for DataSet

Usage

```
## S3 method for class 'DataSet'
summary(object, ...)
```

Arguments

object	A DataSet object
...	Arguments passed to other methods

Value

A summary of the DataSet containing both function-value and runtime based statistics.

Examples

```
summary(dsl[[1]])
```

summary.DataSetList *S3 summary function for DataSetList*

Description

Prints the Function ID, Dimension, Algorithm Id, datafile location and comment for every DataSet in the DataSetList

Usage

```
## S3 method for class 'DataSetList'
summary(object, ...)
```

Arguments

object	The DataSetList to print
...	Arguments for underlying summary function?

Examples

```
summary(dsl)
```

[.DataSetList *S3 extraction function for DataSetList*

Description

S3 extraction function for DataSetList

Usage

```
## S3 method for class 'DataSetList'
x[i, drop = FALSE]
```

Arguments

x	The DataSetList to use
i	The indices to extract
drop	Currently unused parameter

Value

The DataSetList of the DataSets at indices i of DataSetList x

Examples

```
dsl[c(1, 3)]
```

Index