

# Package ‘IOHanalyzer’

October 21, 2021

**Type** Package

**Title** Data Analysis Part of 'IOHprofiler'

**Version** 0.1.6.1

**Author** Hao Wang [cre, aut],  
Diederick Vermetten [aut],  
Carola Doerr [aut],  
Thomas Bäck [aut]

**Maintainer** Hao Wang <h.wang@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** <http://iohprofiler.liacs.nl>,  
<https://github.com/IOHprofiler/IOHanalyzer>

**BugReports** <https://github.com/IOHprofiler/IOHanalyzer/issues>

**Imports** magrittr, dplyr, data.table, ggplot2, plotly, colorspace,  
colorRamps, RColorBrewer, shiny, markdown, reshape2, shinyjs,  
colourpicker, bsplus, DT, kableExtra, stringi, httr, knitr,  
methods, shinydashboard

**LinkingTo** Rcpp

**SystemRequirements** C++11

**RoxygenNote** 7.1.2

**Suggests** Rcpp, testthat, withr, ComplexHeatmap, grid, keyring,  
PlayerRatings, xtable, igraph

**Depends** R (>= 2.10)

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2021-10-21 12:30:02 UTC

## R topics documented:

==.DataSet	4
arrange	5
as.character.DataSet	5
AUC	6
bootstrap_RT	6
c.DataSet	7
c.DataSetList	8
cat.DataSet	8
change_id	9
check_dsc_configured	9
check_format	10
clean_DataSetList	10
DataSet	11
DataSetList	11
dsl	12
dsl_large	13
ECDF	13
fast_RT_samples	14
generate_data.Aggr	14
generate_data.AUC	15
generate_data.ECDF	15
generate_data.ECDF_raw	16
generate_data.hist	17
generate_data.Parameters	17
generate_data.PMF	18
generate_data.Single_Function	18
get_algId	19
get_color_scheme	19
get_color_scheme_dt	20
get_default_ECDF_targets	20
get_dim	21
get_dsc_omnibus	21
get_dsc_posthoc	22
get_dsc_rank	23
get_ECDF_targets	24
get_ERT	24
get_funcId	25
get_funcName	26
get_funvals	26
get_FV_overview	27
get_FV_sample	27

get_FV_summary . . . . .	28
get_id . . . . .	29
get_line_style . . . . .	30
get_marg_contrib_ecdf . . . . .	30
get_maxRT . . . . .	31
get_overview . . . . .	31
get_parId . . . . .	32
get_PAR_name . . . . .	33
get_PAR_sample . . . . .	33
get_PAR_summary . . . . .	34
get_RT_overview . . . . .	35
get_RT_sample . . . . .	36
get_RT_summary . . . . .	37
get_runtimes . . . . .	37
get_shapley_values . . . . .	38
get_static_attributes . . . . .	39
get_static_attribute_values . . . . .	39
get_target_dt . . . . .	40
glicko2_ranking . . . . .	40
IOHanalyzer . . . . .	41
IOH_plot_ly_default . . . . .	42
limit.data . . . . .	42
max_ERTs . . . . .	43
mean_FVs . . . . .	43
pairwise.test . . . . .	44
Plot.FV.Aggregated . . . . .	45
Plot.FV.ECDF_AUC . . . . .	46
Plot.FV.ECDF_Per_Target . . . . .	46
Plot.FV.ECDF_Single_Func . . . . .	47
Plot.FV.Histogram . . . . .	48
Plot.FV.Multi_Func . . . . .	49
Plot.FV.Parameters . . . . .	50
Plot.FV.PDF . . . . .	51
Plot.FV.Single_Func . . . . .	51
Plot.Performviz . . . . .	52
Plot.RT.Aggregated . . . . .	53
Plot.RT.ECDF_AUC . . . . .	54
Plot.RT.ECDF_Multi_Func . . . . .	55
Plot.RT.ECDF_Per_Target . . . . .	55
Plot.RT.ECDF_Single_Func . . . . .	56
Plot.RT.Histogram . . . . .	57
Plot.RT.Multi_Func . . . . .	58
Plot.RT.Parameters . . . . .	59
Plot.RT.PMF . . . . .	60
Plot.RT.Single_Func . . . . .	60
Plot.Stats.Glicko2_Candlestick . . . . .	62
Plot.Stats.Significance_Graph . . . . .	62
Plot.Stats.Significance_Heatmap . . . . .	63

plot_general_data . . . . .	64
print.DataSet . . . . .	65
print.DataSetList . . . . .	66
read_index_file . . . . .	66
register_DSC . . . . .	67
runServer . . . . .	67
save_plotly . . . . .	68
save_table . . . . .	69
scan_index_file . . . . .	69
seq_FV . . . . .	70
seq_RT . . . . .	71
set_color_scheme . . . . .	71
set_DSC_credentials . . . . .	72
SP . . . . .	73
subset.DataSet . . . . .	73
subset.DataSetList . . . . .	74
summary.DataSet . . . . .	75
summary.DataSetList . . . . .	75
[.DataSetList . . . . .	76

## Index 77

---

==.DataSet	<i>S3 generic == operator for DataSets</i>
------------	--

---

### 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

```
ds1[[1]] == ds1[[2]]
```

---

arrange	<i>S3 sort function for DataSetList</i>
---------	---

---

**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	<i>S3 generic as.character operator for DataSet</i>
----------------------	---

---

**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(ds1[[1]])
```

---

AUC *Area Under Curve (Empirical Cumulative Distribution Function)*

---

**Description**

Area Under Curve (Empirical Cumulative Distribution 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(ds1,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	<i>Add unique identifiers to each DataSet in the provided DataSetList based on static attributes</i>
-----------	--

---

**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	<i>Verify that the credentials for DSCtool have been set</i>
----------------------	--

---

**Description**

This uses the keyring package to store and load credentials. If the keyring package does not exist, 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	<i>Check the format of data</i>
--------------	---------------------------------

---

**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	<i>Clean DataSetList object by concatenating DataSets</i>
-------------------	---

---

**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)
```

### 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?

### 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: <ul style="list-style-type: none"> <li>• 'IOHProfiler'</li> <li>• 'COCO'</li> <li>• 'TWO_COL'</li> <li>• 'COCO_BIOBJ'</li> <li>• 'NEVERGRAD'</li> <li>• 'SOS'</li> </ul> <p>These formats are specified in more detail in our github wiki.</p>
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 Dstribution Function of Runtime of a single data set*


---

**Description**

Empirical Cumulative Dstribution 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(ds1,c(12,14))
ECDF(ds1[[1]],c(12,14))
```

---

fast_RT_samples	<i>Function to get just the RT samples needed, without any formatting to improve speed</i>
-----------------	--

---

**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	<i>Generate dataframe of a single function/dimension pair</i>
--------------------	---

---

**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(ds1)
```

---

```
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)
```

**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

**Examples**

```
generate_data.AUC(ds1, get_ECDF_targets(ds1))
generate_data.AUC(NULL, NULL, dt_ecdf = generate_data.ECDF(ds1, get_ECDF_targets(ds1)))
```

---

```
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	Whether 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(ds1, funcId == 1), c(10, 15, 16))
```

---

```
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	Whether to use logarithmic scaling or not

**Examples**

```
generate_data.ECDF_raw(subset(ds1, funcId == 1), c(10, 15, 16))
```



---

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	Whether to use logarithmic scaling or not

### Examples

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

---

generate_data.PMF	<i>Generate dataframe of a single function/dimension pair for creating PDF or PMF plots</i>
-------------------	---

---

**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	<i>Generate dataframe of a single function/dimension pair</i>
-------------------------------	---

---

**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)
```

**Arguments**

dsList	The DataSetList object
start	Optional start value (Runtime or target value)
stop	Optional end value (Runtime or target value)
scale_log	Whether 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.

**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 DataSetList

**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	<i>Get all dimensions present in a DataSetList</i>
---------	--

---

**Description**

Get all dimensions present in a DataSetList

**Usage**

```
get_dim(dsList)
```

**Arguments**

dsList	The DataSetList
--------	-----------------

**Value**

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

**Examples**

```
get_dim(dsl)
```

---

get_dsc_omnibus	<i>Perform omnibus statistical tests on the matrix of rankings from the DSCtool api</i>
-----------------	---

---

**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 to 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))
```

---

get_dsc_posthoc	<i>Perform post-hoc processing on data from DSCtool</i>
-----------------	---

---

**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)), 2, 2)
```

---

get_dsc_rank	<i>Get the matrix of rankings using the DSCtool api for a DataSetList</i>
--------------	---

---

### 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)
```

---

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	<i>Get all function ids present in a DataSetList</i>
------------	--

---

**Description**

Get all function ids present in a DataSetList

**Usage**

```
get_funcId(dsList)
```

**Arguments**

dsList	The DataSetList
--------	-----------------

**Value**

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

**Examples**

```
get_funcId(dsl)
```

---

get_funcName	<i>Get all function names present in a DataSetList</i>
--------------	--

---

**Description**

Get all function names present in a DataSetList

**Usage**

```
get_funcName(dsList)
```

**Arguments**

dsList            The DataSetList

**Value**

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

**Examples**

```
get_funcName(ds1)
```

---

get_funvals	<i>Get all function values present in a DataSetList</i>
-------------	---

---

**Description**

Get all function values present in a DataSetList

**Usage**

```
get_funvals(dsList)
```

**Arguments**

dsList            The DataSetList

**Value**

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

**Examples**

```
get_funvals(ds1)
```

---

get_FV_overview	<i>Get Function Value condensed overview</i>
-----------------	--

---

**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	<i>Get Funtion Value Samples</i>
---------------	----------------------------------

---

**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	<i>Get Function Value Summary</i>
----------------	-----------------------------------

---

**Description**

Get Function Value Summary

**Usage**

```

get_FV_summary(ds, ...)

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

## S3 method for class 'DataSetList'
get_FV_summary(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
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	<i>Get condensed overview of datasets</i>
--------	---

---

**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 algorithm id (from ‘get\_alId’) to ensure backwards compatibility

**Value**

The list of unique identifiers present in dsl

**Examples**

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

---

get_line_style	<i>Get line styles according to the current styleScheme of the IOHalyzer</i>
----------------	--

---

**Description**

Get line styles according to the current styleScheme of the IOHalyzer

**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	<i>Get the marginal contribution of an algorithm to a portfolio</i>
-----------------------	---

---

**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	<i>Get the maximal running time</i>
-----------	-------------------------------------

---

**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_overview	<i>Get condensed overview of datasets</i>
--------------	---

---

**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	<i>Get the parameter names of the algorithm</i>
--------------	---

---

**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 parameter names, if recorded in the data set

**Examples**

```
get_PAR_name(ds1[[1]])
```

---

get_PAR_sample	<i>Get Parameter Value Samples</i>
----------------	------------------------------------

---

**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	<i>Get Parameter Value Summary</i>
-----------------	------------------------------------

---

**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(ds1, 14)
get_PAR_summary(ds1[[1]], 14)
```

---

get_RT_overview	<i>Get Runtime Value condensed overview</i>
-----------------	---

---

**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	<i>Get RunTime Sample</i>
---------------	---------------------------

---

**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	<i>Get RunTime Summary</i>
----------------	----------------------------

---

**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(ds1, 14)
get_RT_summary(ds1[[1]], 14)
```

---

get_runtimes	<i>Get all runtime values present in a DataSetList</i>
--------------	--

---

**Description**

Get all runtime values present in a DataSetList

**Usage**

```
get_runtimes(dsList)
```

**Arguments**

dsList            The DataSetList

**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**

```
get_shapley_values(dsl, get_ECDF_targets(dsl))
```

---

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 function

**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	<i>Generate datatables of runtime or function value targets for a DataSetList</i>
---------------	---

---

**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	<i>Glicko2 ranking of algorithms</i>
-----------------	--------------------------------------

---

**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**

<code>dsl</code>	The DataSetList, can contain multiple functions and dimensions, but should have the same algorithms for all of them
<code>nr_rounds</code>	The number of rounds to run. More rounds leads to a more accurate ranking.
<code>which</code>	Whether to use fixed-target ('by_FV') or fixed-budget ('by_RT') perspective
<code>target_dt</code>	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 aligning data from different heuristics, such as IOHexperimenter. This is done using the [DataSet](#) and [DataSetList](#) functions
- Processing and summarizing this data
- Creating various plots

**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**

<code>title</code>	Title for the plot
<code>x.title</code>	X-axis label
<code>y.title</code>	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**

<code>df</code>	The data to subsample
<code>n</code>	The amount of samples

**Value**

A smaller data.frame

---

max_ERTs	<i>Get the ERT-values for all DataSets in a DataSetList at certain targets</i>
----------	--

---

**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 DataSetList
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	<i>Get the expected function-values for all DataSets in a DataSetList at certain runtimes</i>
----------	---

---

**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 DataSetList
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 function-values

**Examples**

```
mean_FVs(dsl)
```

---

pairwise.test	<i>Performs a pairwise Kolmogorov-Smirnov test on the bootstrapped running times among a data set</i>
---------------	---

---

**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**

<code>x</code>	either a list that contains running time sample for each algorithm as sub-lists, or a DataSetList object
<code>...</code>	all other options
<code>max_eval</code>	list that contains the maximal running time for each algorithm as sub-lists
<code>bootstrap.size</code>	integer, the size of the bootstrapped sample. Set to 0 to disable bootstrapping
<code>ftarget</code>	float, the target value used to determine the running / hitting
<code>which</code>	whether 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.FV.Agregated	<i>Plot expected function value-based comparison over multiple functions or dimensions</i>
-------------------	--

---

**Description**

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

**Usage**

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

```
## S3 method for class 'DataSetList'
Plot.FV.Agregated(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 logarithmically
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.Agregated(dsl)
```

---

Plot.FV.ECDF_AUC	<i>Radarplot of the area under the aggregated ECDF-curve of a DataSetList.</i>
------------------	--

---

**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(ds1, funcId == 1))
```

---

Plot.FV.ECDF_Per_Target	<i>Plot the empirical cumulative distribution as a function of the target values of a DataSetList at certain target runtimes</i>
-------------------------	--

---

**Description**

Plot the empirical cumulative distribution 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 logarithmically
scale.reverse	Whether or not to reverse the x-axis (when using minimization)

**Value**

A plot of the empirical cumulative distribution as a function of the function 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 distribution as a function of the function values of a DataSetList.*

---

**Description**

Plot the aggregated empirical cumulative distribution 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**

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

**Value**

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

**Examples**

```
Plot.FV.ECDF_Single_Func(subset(ds1, funcId == 1))
```

---

<code>Plot.FV.Histogram</code>	<i>Plot histograms of the function values of a DataSetList at a certain target runtime</i>
--------------------------------	--

---

**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**

<code>dsList</code>	A DataSetList (should consist of only one function and dimension).
<code>runtime</code>	The target runtime
<code>plot_mode</code>	How to plot the different histograms for each algorithm. Can be either 'overlay' to show all algorithms on one plot, or 'subplot' to have one plot per algorithm.
<code>use.equal.bins</code>	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 logarithmically
scale.ylog	Whether or not to scale the y-axis logarithmically
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	<i>Plot the parameter values recorded in a DataSetList (aligned by budget)</i>
--------------------	--

---

### 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 logarithmically
scale.ylog	Whether or not to scale the y-axis logarithmically
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(ds1, funcId == 1))
```

---

Plot.FV.PDF	<i>Plot probability density function of the function values of a DataSetList at a certain target runtime</i>
-------------	--

---

**Description**

Plot probability 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 logarithmically

**Value**

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

**Examples**

```
Plot.FV.PDF(subset(ds1, funcId == 1), 100)
```

---

Plot.FV.Single_Func	<i>Plot lineplot of the expected function values of a DataSetList</i>
---------------------	---

---

**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 logarithmically
scale.ylog	Whether or not to scale the y-axis logarithmically
scale.reverse	Whether 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(ds1))

## End(Not run)
```

---

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

---

**Description**

Plot ERT-based comparison over multiple functions or dimensions

**Usage**

```
Plot.RT.Agregated(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.Agregated(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 logarithmically
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.Agregated(ds1)
```

---

Plot.RT.ECDF_AUC	<i>Radarplot of the area under the aggregated ECDF-curve of a DataSetList.</i>
------------------	--

---

**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(ds1, funcId == 1))
```

---

 Plot.RT.ECDF\_Multi\_Func

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

---

### Description

Plot the aggregated empirical cumulative distribution 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 logarithmically by the function 'get_ECDF_targets'

### Value

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

### Examples

```
Plot.RT.ECDF_Multi_Func(ds1)
```

---

 Plot.RT.ECDF\_Per\_Target

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

---

### Description

Plot the empirical cumulative distribution 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 logarithmically

**Value**

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

**Examples**

```
Plot.RT.ECDF_Per_Target(subset(ds1, funcId == 1), 14)
```

---

```
Plot.RT.ECDF_Single_Func
```

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

---

**Description**

Plot the aggregated empirical cumulative distribution 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**

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

**Value**

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

**Examples**

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

---

<code>Plot.RT.Histogram</code>	<i>Plot histograms of the runtimes of a DataSetList at a certain target function value</i>
--------------------------------	--

---

**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**

<code>dsList</code>	A DataSetList (should consist of only one function and dimension).
<code>ftarget</code>	The target function value.
<code>plot_mode</code>	How to plot the different histograms for each algorithm. Can be either 'overlay' to show all algorithms on one plot, or 'subplot' to have one plot per algorithm.
<code>use.equal.bins</code>	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**

<code>dsList</code>	A DataSetList (should consist of only one function OR dimension).
<code>scale.xlog</code>	Whether or not to scale the x-axis logarithmically
<code>scale.ylog</code>	Whether or not to scale the y-axis logarithmically
<code>scale.reverse</code>	Whether or not to reverse the x-axis (when using minimization)
<code>backend</code>	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	<i>Plot the parameter values recorded in a DataSetList (aligned by function value)</i>
--------------------	--

---

### Description

Plot the parameter values recorded in a DataSetList (aligned by function 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 logarithmically
scale.ylog	Whether or not to scale the y-axis logarithmically
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(ds1, funcId == 1))
```

---

Plot.RT.PMF	<i>Plot probability mass function of the runtimes of a DataSetList at a certain target function value</i>
-------------	---

---

**Description**

Plot probability 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 logarithmically
backend	Which plotting library to use. Can be 'plotly' or 'ggplot2'

**Value**

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

**Examples**

```
Plot.RT.PMF(subset(ds1, funcId == 1), 14)
```

---

Plot.RT.Single_Func	<i>Plot lineplot of the ERTs of a DataSetList</i>
---------------------	---

---

**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**

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

**Value**

A plot of ERT-values of the `DataSetList`

**Examples**

```
Plot.RT.Single_Func(subset(ds1, 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(ds1, 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 amount 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 amount 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(ds1, funcId == 2), 16)
```

---

plot_general_data	<i>General function for plotting within IOHanalyzer</i>
-------------------	---

---

**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, ...)
```

**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	Whether or not to show individual points when making a violinplot
...	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**

<code>x</code>	The DataSetList to print
<code>...</code>	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**

<code>fname</code>	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"))
```

---

`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**

<code>name</code>	Your name
<code>username</code>	A username to be identified with. Will be stored on keyring under 'DSCtool_name'
<code>affiliation</code>	Your affiliation (university / company)
<code>email</code>	Your email adress
<code>password</code>	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	<i>Save plotly figure in multiple format</i>
-------------	--

---

**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	<i>Save DataTable in multiple formats</i>
------------	---

---

**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	<i>Scan *.info files for IOHProfiler or COCO</i>
-----------------	--

---

**Description**

Scan \*.info files for IOHProfiler or COCO

**Usage**

```
scan_index_file(folder)
```

**Arguments**

folder	The folder containing the .info files
--------	---------------------------------------

**Value**

The paths to all found .info-files

**Note**

This automatically filters 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)
```

**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

**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	<p>Three default colorschemes are implemented:</p> <ul style="list-style-type: none"> <li>• Default</li> <li>• Variant 1</li> <li>• Variant 2</li> <li>• Variant 3</li> </ul> <p>And it is also possible to select "Custom", which allows uploading of a custom set of colors</p>
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 username 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 successful runs and the success rate

### Examples

```
SP(ds1[[1]]$RT, max(ds1[[1]]$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))
```

---

```
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 execution 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-separated 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

- \* **datasets**
  - dsl, [12](#)
  - dsl\_large, [13](#)
- ==.DataSet, [4](#)
- [.DataSetList, [76](#)
  
- arrange, [5](#)
- as.character.DataSet, [5](#)
- AUC, [6](#)
  
- bootstrap\_RT, [6](#)
  
- c.DataSet, [7](#)
- c.DataSetList, [8](#)
- cat.DataSet, [8](#)
- change\_id, [9](#)
- check\_dsc\_configured, [9](#)
- check\_format, [10](#)
- clean\_DataSetList, [10](#)
  
- DataSet, [11](#), [41](#)
- DataSetList, [11](#), [41](#)
- dsl, [12](#)
- dsl\_large, [13](#)
  
- ECDF, [13](#)
  
- fast\_RT\_samples, [14](#)
  
- generate\_data.Aggr, [14](#)
- generate\_data.AUC, [15](#)
- generate\_data.ECDF, [15](#)
- generate\_data.ECDF\_raw, [16](#)
- generate\_data.hist, [17](#)
- generate\_data.Parameters, [17](#)
- generate\_data.PMF, [18](#)
- generate\_data.Single\_Function, [18](#)
- get\_algId, [19](#)
- get\_color\_scheme, [19](#)
- get\_color\_scheme\_dt, [20](#)
- get\_default\_ECDF\_targets, [20](#)
  
- get\_dim, [21](#)
- get\_dsc\_omnibus, [21](#)
- get\_dsc\_posthoc, [22](#)
- get\_dsc\_rank, [23](#)
- get\_ECDF\_targets, [24](#)
- get\_ERT, [24](#)
- get\_funcId, [25](#)
- get\_funcName, [26](#)
- get\_funvals, [26](#)
- get\_FV\_overview, [27](#)
- get\_FV\_sample, [27](#)
- get\_FV\_summary, [28](#)
- get\_id, [29](#)
- get\_line\_style, [30](#)
- get\_marg\_contrib\_ecdf, [30](#)
- get\_maxRT, [31](#)
- get\_overview, [31](#)
- get\_PAR\_name, [33](#)
- get\_PAR\_sample, [33](#)
- get\_PAR\_summary, [34](#)
- get\_parId, [32](#)
- get\_RT\_overview, [35](#)
- get\_RT\_sample, [36](#)
- get\_RT\_summary, [37](#)
- get\_runtimes, [37](#)
- get\_shapley\_values, [38](#)
- get\_static\_attribute\_values, [39](#)
- get\_static\_attributes, [39](#)
- get\_target\_dt, [40](#)
- glicko2\_ranking, [40](#)
  
- IOH\_plot\_ly\_default, [42](#)
- IOHanalyzer, [41](#)
- IOHanalyzer-package (IOHanalyzer), [41](#)
  
- limit.data, [42](#)
  
- max\_ERTs, [43](#)
- mean\_FVs, [43](#)
  
- pairwise.test, [44](#)

Plot.FV.Agregated, [45](#)  
Plot.FV.ECDF\_AUC, [46](#)  
Plot.FV.ECDF\_Per\_Target, [46](#)  
Plot.FV.ECDF\_Single\_Func, [47](#)  
Plot.FV.Histogram, [48](#)  
Plot.FV.Multi\_Func, [49](#)  
Plot.FV.Parameters, [50](#)  
Plot.FV.PDF, [51](#)  
Plot.FV.Single\_Func, [51](#)  
Plot.Performviz, [52](#)  
Plot.RT.Agregated, [53](#)  
Plot.RT.ECDF\_AUC, [54](#)  
Plot.RT.ECDF\_Multi\_Func, [55](#)  
Plot.RT.ECDF\_Per\_Target, [55](#)  
Plot.RT.ECDF\_Single\_Func, [56](#)  
Plot.RT.Histogram, [57](#)  
Plot.RT.Multi\_Func, [58](#)  
Plot.RT.Parameters, [59](#)  
Plot.RT.PMF, [60](#)  
Plot.RT.Single\_Func, [60](#)  
Plot.Stats.Glicko2\_Candlestick, [62](#)  
Plot.Stats.Significance\_Graph, [62](#)  
Plot.Stats.Significance\_Heatmap, [63](#)  
plot\_general\_data, [64](#)  
print.DataSet, [65](#)  
print.DataSetList, [66](#)  
  
read\_index\_file, [66](#)  
register\_DSC, [67](#)  
runServer, [67](#)  
  
save\_plotly, [68](#)  
save\_table, [69](#)  
scan\_index\_file, [69](#)  
seq\_FV, [70](#)  
seq\_RT, [71](#)  
set\_color\_scheme, [71](#)  
set\_DSC\_credentials, [72](#)  
SP, [73](#)  
subset.DataSet, [73](#)  
subset.DataSetList, [74](#)  
summary.DataSet, [75](#)  
summary.DataSetList, [75](#)