Package 'SummarizedBenchmark'

September 13, 2022

Type Package

Title Classes and methods for performing benchmark comparisons

Version 2.14.0

BugReports https://github.com/areyesq89/SummarizedBenchmark/issues

URL https://github.com/areyesq89/SummarizedBenchmark,
http://bioconductor.org/packages/SummarizedBenchmark/

Description This package defines the BenchDesign and SummarizedBenchmark classes for building, executing, and evaluating benchmark experiments of computational methods. The Summarized-Benchmark
class extends the RangedSummarizedExperiment object, and is designed to provide infrastructure to store and compare the results of applying different methods to a shared data set.
This class provides an integrated interface to store metadata such as method parameters and software
versions as well as ground truths (when these are available) and evaluation metrics.

biocViews Software, Infrastructure

Depends R (>= 3.6), tidyr, SummarizedExperiment, S4Vectors, BiocGenerics, methods, UpSetR, rlang, stringr, utils, BiocParallel, ggplot2, mclust, dplyr, digest, sessioninfo, crayon, tibble

Suggests iCOBRA, BiocStyle, rmarkdown, knitr, magrittr, IHW, qvalue, testthat, DESeq2, edgeR, limma, tximport, readr, scRNAseq, splatter, scater, rnaseqcomp, biomaRt

License GPL (>= 3)

Encoding UTF-8

LazyData true

VignetteBuilder knitr

RoxygenNote 6.1.1

Config/testthat/edition 3

git_url https://git.bioconductor.org/packages/SummarizedBenchmark

git_branch RELEASE_3_15

git_last_commit aac0d7a

git_last_commit_date 2022-04-26

Date/Publication 2022-09-13

Author Alejandro Reyes [aut] (<https://orcid.org/0000-0001-8717-6612>), Patrick Kimes [aut, cre] (<https://orcid.org/0000-0001-6819-9077>)

Maintainer Patrick Kimes <patrick.kimes@gmail.com>

R topics documented:

`addMethod` *Add method to BenchDesign object*

Description

Takes a `BenchDesign` object and the definition of a new method for benchmarking and returns the original `BenchDesign` with the new method included.

At a minimum, a method label (`label =`), and the workhorse function for the method (`func =`) must be specified for the new method.

Parameters for the method must be specified as a `quos` named list of `parameter = value` pairs mapping entries in the benchmarking data to the function parameters. For users familiar with the **ggplot2** package, this can be viewed similar to the `aes =` mapping of data to geometry parameters.

An optional secondary function, `post`, can be specified if the output of the workhorse function, `func`, needs to be further processed. As an example, `post` may be a simple "getter" function for accessing the column of interest from the large object returned by `func`.

Usage

```
addMethod(bd, label, func, params = rlang::quos(), post = NULL,
  meta = NULL)
```

Arguments

`bd`	`BenchDesign` object.
`label`	Character name for the method.
`func`	Primary function to be benchmarked.
`params`	Named quosure list created using `quos` of `parameter = value` pairs to be passed to `func`.
`post`	Optional post-processing function that takes results of `func` as input. Ignored if `NULL`. If multiple assays (metrics) should be generated for each method, this can be accomplished by specifying a named list of post-processing functions, one for each assay. (default = `NULL`)
`meta`	Optional metadata information for method to be included in `colData` of `SummarizedBenchmark` object generated using `link{buildBench}`. See Details for more information. Ignored if `NULL`. (default = `NULL`)

Details

The optional `meta` parameter accepts a named list of metadata tags to be included for the method in the resulting `SummarizedBenchmark` object. This can be useful for two primary cases. First, it can help keep analyses better organized by allowing the specification of additional information that should be stored with methods, e.g. a tag for "method type" or descriptive information on why the method was included in the comparison. Second, and more improtantly, the `meta` parameter can be used to overwrite the package and version information that is automatically extracted from the function specified to `func`. This is particularly useful when the function passed to `func` is a wrapper for a script in (or outside of) R, and the appropriate package and version information can't be directly pulled from `func`. In this case, the user can either manually specify the `"pkg_name"` and `"pkg_vers"` values to `meta` as a list, or specify a separate function that should be used to determine the package name and version. If a separate function should be used, it should be passed to `meta` as a list entry with the name `pkg_func` and first quoted using `quo`, e.g. `list(pkg_func = quo(p.adjust))`.

Value

Modified `BenchDesign` object with new method added.

Author(s)

Patrick Kimes

See Also

`modifyMethod`, `expandMethod`, `dropMethod`

Examples

```
## create example data set of p-values
df <- data.frame(pval = runif(100))

## example calculating qvalue from pvalues

## using standard call
qv <- qvalue::qvalue(p = df$pval)
qv <- qv$qvalue

## adding same method to BenchDesign
bench <- BenchDesign(data = df)
bench <- addMethod(bench,
                   label = "qv",
                    func = qvalue::qvalue,
                    post = function(x) { x$qvalue },
                    params = rlang::quos(p = pval))
```

addPerformanceMetric *Add performance metric to SummarizedBenchmark object*

Description

This is a function to define performance metrics for benchmarking methods. The function is saved into the performanceMetrics slot.

Usage

```
addPerformanceMetric(object, evalMetric, assay, evalFunction = NULL)
```

Arguments

object	A SummarizedBenchmark object.
evalMetric	A string with the name of the evaluation metric.
assay	A string with an assay name. Indicates the assay that should be given as input to this performance metric.
evalFunction	A function that calculates a performance metric. It should contain at least two arguments, query and truth, where query is the output vector of a method and truth is the vector of true values. If additional parameters are specified, they must contain default values. If NULL, the 'evalMetric' string must be the name of a predefined metric available through 'availableMetrics()$function'.

Value

A SummarizedBenchmark object.

Author(s)

Alejandro Reyes

See Also

availableMetrics, performanceMetrics

Examples

```
data( sb )
sb <- addPerformanceMetric(
   object=sb,
   assay="qvalue",
   evalMetric="TPR",
   evalFunction = function( query, truth, alpha=0.1 ){
       goodHits <- sum( (query < alpha) & truth == 1 )
       goodHits / sum(truth == 1)
   }
)
```

allSB *SummarizedBenchmark object of isoform quantification results*

Description

This object is a SummarizedBenchmark object containing isoform quantifications from salmon, sailfish and kallisto from 4 mouse samples (2 hearts and 2 brains) part of the Mouse BodyMap. Its generation is described in one of the vignettes of this package.

Source

Mouse BodyMap (Li et al, 2014). SRA accession numbers SRR5273705, SRR5273689, SRR5273699 and SRR5273683.

Examples

```
data(quantSB)
```

assayNames<-,SummarizedBenchmark,character-method *Set assay names in SummarizedBenchmark object*

Description

Modifies the assay names of a SummarizedBenchmark object.

Usage

```
## S4 replacement method for signature 'SummarizedBenchmark,character'
assayNames(x, ...) <- value
```

Arguments

x	A SummarizedBenchmark object.
...	Futher arguments, perhaps used by methods.
value	A character vector.

Value

Modfied SummarizedBenchmark object.

Author(s)

Alejandro Reyes

See Also

SummarizedBenchmark

Examples

```
data(sb)
assayNames(sb)[2] <- "log2FC"
```

---

availableMetrics *List pre-defined metrics for SummarizedBenchmark objects*

Description

This function returns a data frame summarizing the default performance metrics provided in this package. The data.frame contains three columns, `functions` is the name of the performance metric, `description` is longer description of the performance metric and `requiredTruth` is logical depending on whether the performance metrics require ground truths.

Usage

```
availableMetrics()
```

Value

A data.frame summarizing the default performance metrics provided in this package.

Author(s)

Alejandro Reyes

Examples

```
availableMetrics()
```

BDData *Create a new BDData object*

Description

Initializes a new BDData object of benchmarking data.

Data sets are stored as BDData objects within BenchDesign objects as well as SummarizedBenchmark objects. However, because data is directly specified to the BenchDesign constructor, there is usually no need to call the BDData constructor to create completely new data objects.

The BDData constructor is most useful for extracting the data sets contained in BenchDesign objects as well as SummarizedBenchmark objects. By default, the BDData object stored in SummarizedBenchmark objects will be MD5 hashes rather than the complete original data set. `compareBDData` can be used to compare various forms of BDData, as shown in the examples below.

Usage

```
BDData(data)

## S4 method for signature 'ANY'
BDData(data)

## S4 method for signature 'BenchDesign'
BDData(data)

## S4 method for signature 'SummarizedBenchmark'
BDData(data)

## S4 method for signature 'BDData'
BDData(data)
```

Arguments

`data`	a list object of data or MD5 hash string

Value

BDData object

Author(s)

Patrick Kimes

See Also

`BDData-class`, `BenchDesign`

Examples

```
## construct from data.frame
datadf <- data.frame(x = 1:5, y = runif(5))
bdd_df <- BDData(datadf)
bdd_df

## construct from MD5 hash of data.frame
bdd_md5 <- BDData(digest::digest(datadf))
bdd_md5

## compare two BDData objects
compareBDData(bdd_df, bdd_md5)

## note that the data is the same, i.e. the MD5 hashes match, but the
## data types ("data" vs. "md5has") are different
```

`BDData-class` *BDData class*

Description

Container for data in a BenchDesign object.

Slots

`data` a list or MD5 hash of the data.

`type` a character string indicating whether the data slot contains the 'data' or a 'md5hash' of the data.

Author(s)

Patrick Kimes

See Also

`BDData`, `BenchDesign-class`, `BDMethod-class`, `BDMethodList-class`

BDData<- *Set data in BenchDesign object*

Description

Adds, removes or replaces BDData in BenchDesign object. Data can be removed by setting the value to NULL.

Usage

```
BDData(x) <- value

## S4 replacement method for signature 'BenchDesign,BDDataOrNULL'
BDData(x) <- value
```

Arguments

x	BenchDesign object.
value	BDData or NULL.

Value

modified BenchDesign object

Author(s)

Patrick Kimes

See Also

BDData

Examples

```
bd <- BenchDesign()
BDData(bd) <- BDData(data.frame(x1 = runif(5)))
bd
```

BDMethod *Create a new BDMethod object*

Description

Initializes a new BenchDesign method object for benchmarking.

New BDMethod objects are typically not directly constructed as they have limited use outside of BenchDesign objects. Instead, methods in a BenchDesign object are more commonly created, modified or removed using function calls on the BenchDesign, e.g. using addMethod to add a new method object.

The constructor can also be used to access BDMethod objects stored in BDMethodList and BenchDesign objects.

Usage

```
BDMethod(x, params = rlang::quos(), post = NULL, meta = NULL, ...)

## S4 method for signature 'quosure'
BDMethod(x, params = rlang::quos(), post = NULL,
  meta = NULL, ...)

## S4 method for signature '`function`'
BDMethod(x, params = rlang::quos(), post = NULL,
  meta = NULL, ...)

## S4 method for signature 'BDMethodList'
BDMethod(x, i = 1)

## S4 method for signature 'BenchDesign'
BDMethod(x, i = 1)
```

Arguments

x	main method function or function quosure. Alternative, may be a BDMethodList or BenchDesign object from which the BDMethod should be extracted.
params	list of quosures specifying function parameters. (default = rlang::quos())
post	list of functions to be applied to the output of x. (default = NULL)
meta	list of metadata. (default = NULL)
...	other parameters.
i	integer index or character name of BDMethod in BDMethodList or BenchDesign object.

Value

BDMethod object

Author(s)

Patrick Kimes

See Also

BDMethod-class, BenchDesign, BDMethodList

Examples

```
## create a simple BDMethod
bdm1 <- BDMethod(x = base::mean)

## create a more complex BDMethod
bdm2 <- BDMethod(x = function(x) { x^2 }, post = base::sqrt,
                 meta = list(note = "simple example"))

## construct a BenchDesign with the BDMethod objects
bd <- BenchDesign(method1 = bdm1, method2 = bdm2)

## access a BDMethod in the BenchDesign
BDMethod(bd, "method1")
```

BDMethod-class *BDMethod class*

Description

Container for individual methods to be compared as part of a benchmark experiment defined in a BenchDesign object. In the SummarizedBenchmark framework, methods are defined by a unique combination of functions, parameters, and any relevant meta data.

New BDMethod objects can be created using the BDMethod constructor.

Slots

`f` a function to be benchmarked

`fc` a captured expression of the function `f`

`params` a list of quosures specifying function parameters

`post` a list of functions to be applied to the output of `f`

`meta` a list of meta data

Author(s)

Patrick Kimes

See Also

BDMethod, BenchDesign-class, BDMethodList-class, BDData-class

BDMethod<- *Set method in list or BenchDesign object*

Description

Adds, replaces or removes a named BDMethod method in a BDMethodList or BenchDesign object with a specified BDMethod object.

An existing method can be removed by setting the value to NULL.

Usage

```
BDMethod(x, i) <- value

## S4 replacement method for signature 'BDMethodList,character,BDMethod'
BDMethod(x, i) <- value

## S4 replacement method for signature 'BDMethodList,character,`NULL`'
BDMethod(x, i) <- value

## S4 replacement method for signature 'BenchDesign,character,BDMethod'
BDMethod(x, i) <- value

## S4 replacement method for signature 'BenchDesign,character,`NULL`'
BDMethod(x, i) <- value
```

Arguments

x	BenchDesign or BDMethodList object.
i	character name of method.
value	BDMethod or NULL.

Value

modified BenchDesign object

Author(s)

Patrick Kimes

See Also

BDMethod

Examples

```
bd <- BenchDesign()
BDMethod(bd, "avg") <- BDMethod(x = base::mean)
bd
```

BDMethodList *Create a new BDMethodList object*

Description

Initializes a new SimpleList of BenchDesign method (BDMethod) objects.

Similar to BDMethod objects, BDMethodList typically do not need to be directly constructed. Because the list of methods is only useful as art of a BenchDesign object, it is more common to simply manipulate the list of methods through calls to the corresponding BenchDesign, e.g. addMethod to add a new method to the list.

The constructor can also be used to access the BDMethodList list of methods in a BenchDesign object.

Usage

```
BDMethodList(..., x = NULL)

## S4 method for signature 'ANY'
BDMethodList(..., x = NULL)

## S4 method for signature 'BenchDesign'
BDMethodList(..., x = NULL)

## S4 method for signature 'SummarizedBenchmark'
BDMethodList(..., x = NULL)
```

Arguments

...	a named list of BDMethod objects
x	a BenchDesign or SummarizedBenchmark object to extract the list of methods from. (default = NULL)

Value

BDMethodList object

Author(s)

Patrick Kimes

See Also

BDMethodList-class, BenchDesign, BDMethod

Examples

```
## construct an empty list
bdml <- BDMethodList()

## construct a list with BDMethod objects
bdml <- BDMethodList(m_method = BDMethod(base::mean),
                     s_method = BDMethod(function(x) { x^2 }))
bdml

## construct a BenchDesign with a BDMethodList
bd <- BenchDesign(methods = bdml)

## access the BDMethodList in the BenchDesign
BDMethodList(bd)
```

BDMethodList-class *BDMethodList class*

Description

Extension of the SimpleList class to contain a list of BDMethod objects. The class serves as the primary container for the set of methods in the BenchDesign class.

New BDMethodList objects can be created using the BDMethodList constructor.

Author(s)

Patrick Kimes

See Also

BDMethodList, BenchDesign-class, BDMethod-class, BDData-class

BDMethodList<- *Set method list in BenchDesign object*

Description

Replaces the BDMethodList list of methods in a BenchDesign object with a specified BDMethodList object.

Usage

```
BDMethodList(x) <- value

## S4 replacement method for signature 'BenchDesign,BDMethodList'
BDMethodList(x) <- value
```

Arguments

x	BenchDesign object.
value	BDMethodList list of methods.

Value

modified BenchDesign object

Author(s)

Patrick Kimes

See Also

BDMethod, BDMethodList

Examples

```
bd <- BenchDesign()
BDMethodList(bd) <- BDMethodList(avg = BDMethod(x = base::mean))
bd
```

BenchDesign *Create a new BenchDesign object*

Description

Initializes a new BenchDesign object of benchmarking methods and data.

The BenchDesign class serves as the core container for methods and data used for benchmarking in the SummarizedBenchmark package. The object can be initialized with a list of methods ot be benchmarked, a default benchmarking data set, both or neither. Methods must be passed to the constructor as BDMethod or BDMethodList objects.

The constructor can also be used to access the BenchDesign stored in a SummarizedBenchmark object.

Usage

```
BenchDesign(..., methods = NULL, data = NULL)

## S4 method for signature 'ANY'
BenchDesign(..., methods = NULL, data = NULL)

## S4 method for signature 'SummarizedBenchmark'
BenchDesign(methods, data)
```

Arguments

...	named set of BDMethod objects and/or unnamed BenchDesign objects. Only the methods of any BenchDesign object will be used, and the data slot of the objects will be ignored.
methods	named set of BDMethod objects and/or unnamed BenchDesign objects as a list. (default = NULL)
data	optional data.frame or other list object to be used in the benchmark. (default = NULL)

Value

BenchDesign object.

Author(s)

Patrick Kimes

See Also

BenchDesign-class, BDMethod, BDMethodList

Examples

```
## with no input
bd <- BenchDesign()

## with data - data must be a named argument
datadf <- data.frame(pval = runif(20), x1 = rnorm(20))
bd <- BenchDesign(data = datadf)

## with two methods and data
method_bh <- BDMethod(stats::p.adjust, params = rlang::quos(p = pval, method = "BH"))
method_bf <- BDMethod(stats::p.adjust, params = rlang::quos(p = pval, method = "bonferroni"))
bd <- BenchDesign(bh = method_bh, bonf = method_bf,
                  data = datadf)

## with BDMethodList and data
bdml <- BDMethodList(bh = method_bh, bonf = method_bf)
bd <- BenchDesign(methods = bdml, data = datadf)
```

BenchDesign-class *BenchDesign class*

Description

Along with the SummarizedBenchmark class, one of the two main classses of the SummarizedBenchmark package. The BenchDesign class serves as a container for both the set of methods to be benchmarked and optionally the data to be used for benchmarking.

Methods are organized as BDMethod objects and stored in as a list using the BDMethodList class. The BDData class is used to store benchmarking data, or in some cases, just the MD5 hash of the original data set. Any list object, including data.frame objects, can be specified for data. More details on the component classes are provided in the corresponding class documentation.

For details on how to create new BenchDesign objects, see the documentation for the `BenchDesign` constructor.

Slots

`data` a list containing the data to be used in the benchmark.

`methods` a `BDMethodList` list of `BDMethod` objects to be compared in the benchmark.

Author(s)

Patrick Kimes

See Also

`BenchDesign`, `BDMethod-class`, `BDMethodList-class`, `BDData-class`

buildBench *Execute BenchDesign*

Description

Function to evaluate methods defined in a BenchDesign on a supplied data set to generate a SummarizedBenchmark of benchmarking results. In addition to the results of applying each method on the data, the returned SummarizedBenchmark also includes metadata for the methods in the colData of the returned object, metadata for the data in the rowData, and session information in the metadata.

Usage

```
buildBench(bd, data = NULL, truthCols = NULL, ftCols = NULL,
  sortIDs = FALSE, keepData = FALSE, catchErrors = TRUE,
  parallel = FALSE, BPPARAM = bpparam())
```

Arguments

bd	BenchDesign object.
data	Data set to be used for benchmarking, will take priority over data set specified to BenchDesign object. Ignored if NULL. (default = NULL)
truthCols	Character vector of column names in data set corresponding to ground truth values for each assay. If specified, column will be added to the groundTruth DataFrame of the returned SummarizedBenchmark object. If the BenchDesign includes only a single assay, the same name will be used for the assay. If the BenchDesign includes multiple assays, to map data set columns with assays, the vector must have names corresponding to the assay names specified to the post parameter at each addMethod call. (default = NULL)
ftCols	Vector of character names of columns in data set that should be included as feature data (row data) in the returned SummarizedBenchmark object. (default = NULL)
sortIDs	Whether the output of each method should be merged and sorted using IDs. See Details for more information. (default = FALSE)
keepData	Whether to store the data as part of the BenchDesign slot of the returned SummarizedBenchmark object. If FALSE, a MD5 hash of the data will be stored with the BenchDesign slot. (default = FALSE)
catchErrors	logical whether errors produced by methods during evaluation should be caught and printed as a message without stopping the entire build process. (default = TRUE)
parallel	Whether to use parallelization for evaluating each method. Parallel execution is performed using **BiocParallel**. Parameters for parallelization should be specified with register or through the BPPARAM parameter. (default = FALSE)
BPPARAM	Optional BiocParallelParam instance to be used when parallel is TRUE. If not specified, the default instance from the parameter registry is used.

Details

Parallelization is performed across methods. Therefore, there is currently no benefit to specifying more cores than the total number of methods in the BenchDesign object.

By default, errors thrown by individual methods in the BenchDesign are caught during evaluation and handled in a way that allows buildBench to continue running with the other methods. The error is printed as a message, and the corresponding column in the returned SummarizedBenchmark object is set to NA. Since many benchmarking experiments can be time and computationally intensive, having to rerun the entire analysis due to a single failed method can be frustrating. Default error catching was included to alleviate these frustrations. However, if this behavior is not desired, setting `catchErrors = FALSE` will turn off error handling.

If `sortIDs = TRUE`, each method must return a named vector or list. The names will be used to align the output of each method in the returned SummarizedBenchmark. Missing values from each method will be set to NA. This can be useful if the different methods return overlapping, but not identical, results. If `truthCols` is also specified, and sorting by IDs is necessary, rather than specifying `sortIDs = TRUE`, specify the string name of a column in the `data` to use to sort the method output to match the order of `truthCols`.

When a method specified in the BenchDesign does not have a postprocessing function specified to `post =`, the trivial `base::identity` function is used as the default postprocessing function.

Value

SummarizedBenchmark object.

Author(s)

Patrick Kimes

See Also

updateBench

Examples

```
## with toy data.frame
df <- data.frame(pval = rnorm(100))
bench <- BenchDesign(data = df)

## add methods
bench <- addMethod(bench, label = "bonf", func = p.adjust,
                   params = rlang::quos(p = pval, method = "bonferroni"))
bench <- addMethod(bench, label = "BH", func = p.adjust,
                   params = rlang::quos(p = pval, method = "BH"))

## evaluate benchmark experiment
sb <- buildBench(bench)

## evaluate benchmark experiment w/ data sepecified
sb <- buildBench(bench, data = df)
```

compareBDData *Compare BDData objects*

Description

Simple comparison of two BDData objects based on comparing both type and data hash.

Usage

```
compareBDData(x, y)
```

Arguments

x	a BDData or BenchDesign object
y	a BDData or BenchDesign object

Value

list of two values giving agreement of "data" and "type".

Author(s)

Patrick Kimes

See Also

compareBenchDesigns

Examples

```
## compare data with same MD5 hash value
bdd1 <- BDData(data.frame(x = 1:10))
bdd1h <- hashBDData(bdd1)
compareBDData(bdd1, bdd1h)

## compare different data, both same type
bdd2 <- BDData(data.frame(x = 2:11))
bdd2h <- hashBDData(bdd2)
compareBDData(bdd1, bdd2)
compareBDData(bdd1h, bdd2h)

## compare completely different data
compareBDData(bdd1, bdd2h)
```

compareBDMethod *Compare BDMethod objects*

Description

Simple comparison of two BDMethod objects based on meta data.

Usage

```
compareBDMethod(x, y)
```

Arguments

x	a BDMethod object
y	a BDMethod object

Value

logical value indicating whether the two objects produced the same meta data.

Author(s)

Patrick Kimes

See Also

compareBenchDesigns

Examples

```
bdm1 <- BDMethod(stats::rnorm, params = rlang::quos(n = 100))
bdm2 <- BDMethod(stats::rt, params = rlang::quos(n = 100, df = 1))

compareBDMethod(bdm1, bdm2)
```

compareBenchDesigns *Compare BenchDesign objects*

Description

Comparison of BenchDesign objects and BenchDesign method information stored in SummarizedBenchmark objects. Inputs can be either BenchDesign or SummarizedBenchmark objects. If SummarizedBenchmark objects are specified, the method metadata stored in the colData will be used for the comparison. If only a single SummarizedBenchmark object is specified, the colData information will be compared with the BenchDesign object in the BenchDesign slot of the object. To compare the BenchDesign slots of SummarizedBenchmark objects, the BenchDesigns should be extracted with BenchDesign(sb) and passed as inputs (see Examples).

Usage

```
compareBenchDesigns(x, y = NULL, ...)

## S4 method for signature 'SummarizedBenchmark,missing'
compareBenchDesigns(x, y = NULL,
  ...)

## S4 method for signature 'SummarizedBenchmark,SummarizedBenchmark'
compareBenchDesigns(x,
  y = NULL, ...)

## S4 method for signature 'SummarizedBenchmark,BenchDesign'
compareBenchDesigns(x,
  y = NULL, ...)

## S4 method for signature 'BenchDesign,SummarizedBenchmark'
compareBenchDesigns(x,
  y = NULL, ...)

## S4 method for signature 'BenchDesign,BenchDesign'
compareBenchDesigns(x, y = NULL, ...)
```

Arguments

x	a SummarizedBenchmark or BenchDesign object
y	an optional second SummarizedBenchmark or BenchDesign object (default = NULL)
...	other parameters

Value

list of comparison results

Author(s)

Patrick Kimes

See Also

compareBDMethod, compareBDData

Examples

```
bd1 <-
    BenchDesign(norm_sd = BDMethod(stats::rnorm,
                                   params = rlang::quos(n = n),
                                   post = sd),
                t_sd = BDMethod(stats::rt,
                                params = rlang::quos(n = n, df = 1),
```

```
                                post = sd))
bd2 <- addMethod(bd1, "chi_sd",
                 func = stats::rchisq,
                 params = rlang::quos(n = n, df = 1),
                 post = sd)

compareBenchDesigns(bd1, bd2)
```

---

dropMethod *Remove method from BenchDesign object*

Description

Takes a BenchDesign object and the name of an existing method and returns a reduced BenchDesign object with the method removed.

Usage

```
dropMethod(bd, label)
```

Arguments

bd	BenchDesign object.
label	Character name of method.

Value

Modified BenchDesign object with specified method dropped.

Author(s)

Patrick Kimes

See Also

modifyMethod, expandMethod, addMethod

Examples

```
## empty BenchDesign
bench <- BenchDesign()

## add methods
bench <- addMethod(bench, label = "bonf", func = p.adjust,
                   params = rlang::quos(p = pval, method = "bonferroni"))
bench <- addMethod(bench, label = "BH", func = p.adjust,
                   params = rlang::quos(p = pval, method = "BH"))
BDMethodList(bench)
```

```
## remove methods
bench <- dropMethod(bench, label = "bonf")
BDMethodList(bench)
```

`estimateMetricsForAssay` *Estimate performance metrics in SummarizedBenchmark object*

Description

These functions estimate the performance metrics, either passed as arguments or added previously with the `addPerformanceMetric` function. The function will estimate the performance metric for each method.

Usage

```
estimateMetricsForAssay(object, assay, evalMetric = NULL,
  addColData = FALSE, evalFunction = NULL, tidy = FALSE, ...)

estimatePerformanceMetrics(object, addColData = FALSE, tidy = FALSE,
  rerun = TRUE, ...)
```

Arguments

`object`	A `SummarizedBenchmark` object.
`assay`	A string with an assay name. Indicates the assay that should be given as input to this performance metric.
`evalMetric`	A string with the name of the evaluation metric.
`addColData`	Logical (default: FALSE). If TRUE, the results are added to the `colData` slot of the `SummarizedExperiment` object and the object is returned. If FALSE, only a `DataFrame` with the results is returned.
`evalFunction`	A function that calculates a performance metric. It should contain at least two arguments, query and truth, where query is the output vector of a method and truth is the vector of ground true values. If additional parameters are specified, they must contain default values. If this parameter is passed, the metrics in the object are ignored and only this evaluation metric is estimated.
`tidy`	Logical (default: FALSE). If TRUE, a long formated `data.frame` is returned.
`...`	Additional parameters passed to the performance functions.
`rerun`	Logical (default: TRUE). By default, all performance metrics are recalculated everytime that `estimatePerformanceMetrics` is called. If FALSE, performance metrics will only be calculated for newly added methods or modified methods.

Value

Either a SummarizedBenchmark object, a DataFrame or a data.frame.

Functions

- estimateMetricsForAssay: Estimate performance metrics for a given assay
- estimatePerformanceMetrics: Estimate performance metrics for all assays

Author(s)

Alejandro Reyes

See Also

availableMetrics, performanceMetrics

Examples

```
data( sb )
sb <- addPerformanceMetric(
   object=sb,
   assay="qvalue",
   evalMetric="TPR",
   evalFunction = function( query, truth, alpha=0.1 ){
       goodHits <- sum( (query < alpha) & truth == 1 )
       goodHits / sum(truth == 1)
   }
)

qvalueMetrics <- estimateMetricsForAssay( sb, assay="qvalue" )
allMetrics <- estimatePerformanceMetrics( sb )
allMetricsTidy <- estimatePerformanceMetrics( sb, tidy=TRUE )
```

---

expandMethod *Expand method in BenchDesign object*

Description

Takes a BenchDesign object, the name of an existing method, and new parameter specifications, and returns a modified BenchDesign object with new methods added. The named method is "expanded" to mutliple methods according to the specified set of parameters.

Usage

```
expandMethod(bd, label, params, onlyone = NULL, .replace = FALSE,
  .overwrite = FALSE)
```

Arguments

bd	BenchDesign object.
label	Character name of method to be expanded.
params	Named list of quosure lists specifying the label of the new methods to be added to the BenchDesign, and the set of parameters to overwrite in the original method definition for each new method. Alternatively, if onlyone is non-NULL, a single quosure list with name = value pairs specifying the label of the new methods and the values to use for overwriting the parameter specified in onlyone.
onlyone	Character name of a parameter to be modified. Only specify if just a single parameter should be replaced in the original method definition. Ignored if NULL. (default = NULL)
.replace	Logical whether original label method should be removed. (default = FALSE)
.overwrite	Logical whether to overwrite the existing list of parameters (TRUE) or to simply add the new parameters to the existing list (FALSE). (default = FALSE)

Value

Modified BenchDesign object with new methods with specified parameters added.

Author(s)

Patrick Kimes

See Also

modifyMethod, addMethod, dropMethod

Examples

```
## empty BenchDesign
bench <- BenchDesign()

## add basic 'padjust' method
bench <- addMethod(bench, label = "padjust",
                   func = p.adjust,
                   params = rlang::quos(p = pval, method = "none"))

## modify multiple parameters - params is a list of quosure lists
newparams <- list(bonf = rlang::quos(p = round(pval, 5), method = "bonferonni"),
                  bh = rlang::quos(p = round(pval, 3), method = "BH"))
bench_exp <- expandMethod(bench, label = "padjust", params = newparams)
BDMethodList(bench_exp)

## only modify a single parameter - params is a quosure list
newparams <- rlang::quos(bonf = "bonferonni", BH = "BH")
bench_exp <- expandMethod(bench, label = "padjust", onlyone = "method", params = newparams)
BDMethodList(bench_exp)
```

groundTruths *Get ground truths in SummarizedBenchmark object*

Description

Method to get groundTruths in SummarizedBenchmark object.

Usage

```
groundTruths(object, ...)

## S4 method for signature 'SummarizedBenchmark'
groundTruths(object, ...)
```

Arguments

object	a SummarizedBenchmark object.
...	further arguments, perhaps used by methods.

Value

modified BenchDesign object

Author(s)

Alejandro Reyes

groundTruths<- *Set ground truths in SummarizedBenchmark object*

Description

Method to set groundTruths in SummarizedBenchmark object.

Usage

```
groundTruths(object, ...) <- value

## S4 replacement method for signature 'SummarizedBenchmark'
groundTruths(object, ...) <- value
```

Arguments

object	a SummarizedBenchmark object.
...	futher arguments, perhaps used by methods.
value	replacement set of ground truths.

Value

modified BenchDesign object

Author(s)

Alejandro Reyes, Patrick Kimes

`hashBDData` *Hash data in BDData object*

Description

Repliaces data stored in a `BDData` object with the MD5 hash of the data. If the data was already a MD5 hash, the original object is returned unchanged. The method can be called directly on `BenchDesign` objects to hash the underlying data as well.

Usage

```
hashBDData(object)

## S4 method for signature 'BDData'
hashBDData(object)

## S4 method for signature 'BenchDesign'
hashBDData(object)
```

Arguments

`object`	a `BDData` or `BenchDesign` object

Value

an object of the same class as `object` with data converted to a MD5 hash.

Author(s)

Patrick Kimes

mcols<-,SummarizedBenchmark-method

Set meta data columns in SummarizedBenchmark object

Description

Modifies the mcols slot of a SummarizedBenchmark object.

Usage

```
## S4 replacement method for signature 'SummarizedBenchmark'
mcols(x, ...) <- value
```

Arguments

x	A SummarizedBenchmark object.
...	Futher arguments, perhaps used by methods.
value	A DataFrame of meta data.

Value

Modfied SummarizedBenchmark object.

Author(s)

Alejandro Reyes

See Also

SummarizedBenchmark

modifyMethod

Modify method in BenchDesign object

Description

Takes a BenchDesign object, the name of an existing method, and new parameter specifications, and returns a modified BenchDesign object with the specified changes.

Usage

```
modifyMethod(bd, label, params, .overwrite = FALSE)
```

Arguments

bd	BenchDesign object.
label	Character name of method to be modified.
params	Named quosure list created using quos of parameter = value paiars to replace in the method definition. The post, and meta parameters of the method can be modified using the special keywords, bd.post, and bd.meta (the prefix denoting that these values should modify BenchDesign parameters). All other named parameters will be added to the list of parameters to be passed to func.
.overwrite	Logical whether to overwrite the complete existing list of parameters to be passed to func (TRUE), or to simply add the new parameters to the existing list and only replace overlapping parameters (FALSE). (default = FALSE)

Value

Modified BenchDesign object with single method parameters modified.

Author(s)

Patrick Kimes

See Also

addMethod, expandMethod, dropMethod

Examples

```
## empty BenchDesign
bench <- BenchDesign()

## add method
bench <- addMethod(bench, label = "qv",
                   func = qvalue::qvalue,
                   post = function(x) { x$qvalue },
                   meta = list(note = "storey's q-value"),
                   params = rlang::quos(p = pval))

## modify method 'meta' property of 'qv' method
bench <- modifyMethod(bench, label = "qv",
                      params = rlang::quos(bd.meta =
                           list(note = "Storey's q-value")))

## verify that method has been updated
printMethod(bench, "qv")
```

`performanceMetrics` *Get performance metrics in SummarizedBenchmark object*

Description

Given a SummarizedBenchmark object, returns a list of lists of performance metrics that have been defined for each assay. Optionally, if `assay =` is specified, performance metrics for only the specified subset of specified assays are returned.

Usage

```
performanceMetrics(object, ...)

## S4 method for signature 'SummarizedBenchmark'
performanceMetrics(object, assay = NULL)
```

Arguments

`object`	a SummarizedBenchmark object.
`...`	futher arguments, perhaps used by methods.
`assay`	a character string indicating an assay name.

Value

A SimpleList with one element for each assay. Each element of the list contains a list of performance metric functions.

Author(s)

Alejandro Reyes

See Also

addPerformanceMetric, estimatePerformanceMetrics

Examples

```
data(sb)
performanceMetrics(sb)
performanceMetrics(sb, assay = "qvalue")
```

`performanceMetrics<-` *Set performance metrics in SummarizedBenchmark object*

Description

Replaces the list of performance metrics in a SummarizedBenchmark object with a new list of performance metric lists.

Usage

```
performanceMetrics(object, ...) <- value


  ## S4 replacement method for signature 'SummarizedBenchmark,SimpleList'
performanceMetrics(object) <- value
```

Arguments

object	a SummarizedBenchmark object.
...	futher arguments, perhaps used by methods.
value	a SimpleList of the same length as the number of assays.

Value

Silently, the newly specified SimpleList of performance metric lists.

Author(s)

Alejandro Reyes

See Also

addPerformanceMetric, estimatePerformanceMetrics, performanceMetrics

Examples

```
data(sb)
performanceMetrics(sb)
performanceMetrics(sb) <- SimpleList(qvalue = list(), logFC = list())
```

`plotMethodsOverlap` *Plot UpSetR for SummarizedBenchmark object*

Description

This function looks for an assay, called by default 'qvalue', and given an alpha threshold, it binarizes the assay matrix depending on whether its values are below the alpha threshold. Then it uses the function `upset` to plot the overlaps. The plot is only generated if at least 2 methods have observations that pass the alpha threshold.

Usage

```
plotMethodsOverlap(object, assay = "qvalue", alpha = 0.1, ...)
```

Arguments

`object`	A `SummarizedBenchmark` object.
`assay`	The name of an assay.
`alpha`	An alpha value.
`...`	Further arguments passed to `upset`

Value

An upseR plot.

Author(s)

Alejandro Reyes

See Also

`plotROC`, `estimatePerformanceMetrics`

Examples

```
data(sb)
## Not run:
plotMethodsOverlap(sb)

## End(Not run)
```

plotROC *Plot ROC curve for SummarizedBenchmark object*

Description

This function inputs a SummarizedBenchmark object, looks for an assay called 'qvalue' and plots receiver operating characteristic curves for each of the methods to benchmark.

Usage

```
plotROC(object, assay = "qvalue")
```

Arguments

object	A SummarizedBenchmark object.
assay	An assay name.

Value

A ggplot object.

Author(s)

Alejandro Reyes

See Also

plotMethodsOverlap, estimatePerformanceMetrics

Examples

```
data(sb)
## Not run:
plotROC(sb)

## End(Not run)
```

printMethod *Pretty print methods in a BenchDesign object*

Description

Print out details about a method included in the BenchDesign. The `printMethods` function is just a wrapper to call `printMethod` on all methods in the BenchDesign.

Usage

```
printMethod(bd, n = NULL)

printMethods(bd)
```

Arguments

bd	BenchDesign object.
n	name of a method in the BenchDesign to show.

Value

Brief description is returned to console.

Author(s)

Patrick Kimes

See Also

BDMethod-class, BenchDesign-class

Examples

```
## create empty BenchDesign
bench <- BenchDesign()

## currently no methods
printMethods(bench)

## add method
bench <- addMethod(bench, label = "method_a", p.adjust)
bench <- addMethod(bench, label = "method_b", qvalue::qvalue)

## show a single method
printMethod(bench, "method_a")

## show all methods
printMethods(bench)
```

sb *SummarizedBenchmark example*

Description

This object contains the example data from the iCOBRA package reformatted as a SummarizedBenchmark object. It consists of differential expression results from DESeq2 edgeR and limma-voom.

Source

Example data from the iCOBRA package.

Examples

```
data(sb)
```

show,BDData-method *Show BDData object*

Description

Show BDData object

Usage

```
## S4 method for signature 'BDData'
show(object)
```

Arguments

object	BDData object to show

Value

Print description of BDData object to console

show,BDMethod-method *Show BDMethod object*

Description

Show BDMethod object

Usage

```
## S4 method for signature 'BDMethod'
show(object)
```

Arguments

object	BDMethod object to show

Value

Print description of BDMethod object to console

show,BDMethodList-method *Show BDMethodList object*

Description

Show BDMethodList object

Usage

```
## S4 method for signature 'BDMethodList'
show(object)
```

Arguments

object	BDMethodList object to show

Value

Print description of BDMethodList object to console

show,BenchDesign-method *Show BenchDesign object*

Description

Show BenchDesign object

Usage

```
## S4 method for signature 'BenchDesign'
show(object)
```

Arguments

object	BenchDesign object to show

Value

Print description of BenchDesign object to console

SummarizedBenchmark *Create a new SummarizedBenchmark object*

Description

Function to construct SummarizedBenchmark objects.

Usage

```
SummarizedBenchmark(assays, colData, ftData = NULL, groundTruth = NULL,
  performanceMetrics = NULL, BenchDesign = NULL, ...)
```

Arguments

assays	A list containing outputs of the methods to be benchmark. Each element of the list must contain a matrix or data.frame of n x m, n being the number of features tested (e.g. genes) and m being the number of methods in the benchmark. Each element of the list must contain a single assay (the outputs of the methods). For example, for a benchmark of differential expression methods, one assay could contain the q-values from the different methods and another assay could be the estimated log fold changes.
colData	A DataFrame describing the annotation of the methods. These could include version of the software or the parameters used to run them.

ftData — A DataFrame object describing the rows. This parameter is equivalent to the parameter rowData of a SummarizedExperiment.

groundTruth — If present, a DataFrame containing the ground truths. If provided, the number of columns must be the same as the number of assays (NA's are accepted). The names of the columns should have the same names as the assays.

performanceMetrics — A SimpleList of the same length as the number of assays. Each element of the list must be a list of functions. Each function must contain the parameters 'query' and 'truth'.

BenchDesign — A BenchDesign containing the code used to construct the object. (default = NULL)

... — Additional parameters passed to SummarizedExperiment.

Value

A SummarizedBenchmark object.

Author(s)

Alejandro Reyes

Examples

```
## loading the example data from iCOBRA
library(iCOBRA)
data(cobradata_example)

## a bit of data wrangling and reformatting
assays <- list(
    qvalue=cobradata_example@padj,
    logFC=cobradata_example@score )
assays[["qvalue"]]$DESeq2 <- p.adjust(cobradata_example@pval$DESeq2, method="BH")
groundTruth <- DataFrame( cobradata_example@truth[,c("status", "logFC")] )
colnames(groundTruth) <- names( assays )
colData <- DataFrame( method=colnames(assays[[1]]) )
groundTruth <- groundTruth[rownames(assays[[1]]),]

## constructing a SummarizedBenchmark object
sb <- SummarizedBenchmark(
    assays=assays, colData=colData,
    groundTruth=groundTruth )
colData(sb)$label <- rownames(colData(sb))
```

SummarizedBenchmark-class *SummarizedBenchmark class*

Description

Extension of the RangedSummarizedExperiment to store the output of different methods intended for the same purpose in a given dataset. For example, a differential expression analysis could be done using **limma**-voom, **edgeR** and **DESeq2**. The SummarizedBenchmark class provides a framework that is useful to store, benckmark and compare results.

Slots

performanceMetrics A SimpleList of the same length as the number of assays containing performance functions to be compared with the ground truths.

BenchDesign A BenchDesign originally used to generate the results in the object.

Author(s)

Alejandro Reyes

Examples

```
## loading the example data from iCOBRA
library(iCOBRA)
data(cobradata_example)

## a bit of data wrangling and reformatting
assays <- list(
    qvalue=cobradata_example@padj,
    logFC=cobradata_example@score )
assays[["qvalue"]]$DESeq2 <- p.adjust(cobradata_example@pval$DESeq2, method="BH")
groundTruth <- DataFrame( cobradata_example@truth[,c("status", "logFC")] )
colnames(groundTruth) <- names( assays )
colData <- DataFrame( method=colnames(assays[[1]]) )
groundTruth <- groundTruth[rownames(assays[[1]]),]

## constructing a SummarizedBenchmark object
sb <- SummarizedBenchmark(
    assays=assays, colData=colData,
    groundTruth=groundTruth )
```

tdat *Example data.frame containing results for 50 two-sample t-tests.*

Description

Example data.frame containing results for 50 two-sample t-tests.

Format

a data.frame that contains the results of 50 simulated two-sample t-tests, with each row corresponding to an independent test. The data.frame includes the following 5 columns: 1. H = binary 0/1 whether data for the test was simulated under the null (0) or alternative (1) 2. test_statistic = test-statistics of the t-test 3. effect_size = mean difference between the two sample groups 4. pval = p-value of the t-test 5. SE = standard error of the t-test

Examples

```
data(tdat)
```

tidyBDMethod *Tidy BDMethod Data*

Description

A helper function to extract information for a single or multiple BDMethod object or a list of BDMethod objects.

Usage

```
tidyBDMethod(obj, dat = NULL, eval = FALSE, label = FALSE)

## S4 method for signature 'BDMethod'
tidyBDMethod(obj, dat, eval)

## S4 method for signature 'list'
tidyBDMethod(obj, dat = NULL, eval = FALSE,
  label = FALSE)

## S4 method for signature 'SimpleList'
tidyBDMethod(obj, dat = NULL, eval = FALSE,
  label = FALSE)

## S4 method for signature 'BenchDesign'
tidyBDMethod(obj, dat = NULL, eval = FALSE,
  label = FALSE)
```

Arguments

obj	BDMethod object, list/List of BDMethod objects (e.g. a BDMethodList), or a BenchDesign object
dat	optional data object to use when evaluating any unevaluated expressionsin bdm meta data. (default = NULL)
eval	logical whether to evaluate any quosures in the meta slot of the BDMethod objects. (default = FALSE)
label	logical whether to add a "label" column to the resulting table containing the names of the methods if obj was specified as a named list. (default = FALSE)

Details

If any quosures are specified to the "meta" slot of a BDMethod object, the quosure is converted to a text string using rlang::quo_text.

Value

A named vector of meta data if only a single BDMethod object specified, else a tibble of meta data for the specified list of methods.

Author(s)

Patrick Kimes

---

tidyUpMetrics *Tidy up performance metrics in SummarizedBenchmark object*

Description

This function takes as input a SummarizedBenchmark object, extracts the estimated performance metrics and reformats them into a long-formated data frame.

Usage

```
tidyUpMetrics(object)
```

Arguments

object	A SummarizedBenchmark object.

Value

A tidy data.frame

Author(s)

Alejandro Reyes

See Also

estimatePerformanceMetrics

Examples

```
data( "sb", package="SummarizedBenchmark" )
sb <- estimateMetricsForAssay( sb, assay="qvalue", evalMetric="rejections",
    evalFunction=function( query, truth, alpha=0.1 ){
        sum( query < alpha )
    },
    addColData=TRUE )
tidyUpMetrics( sb )
```

---

updateBench *Check/Update SummarizedBenchmark*

Description

Function to update or check status of SummarizedBenchmark results.

If only a SummarizedBenchmark object is specified, the function will check whether 'func', 'param', 'meta', 'post' or the 'pkg_vers' of the methods in the BenchDesign stored with the SummarizedBenchmark do not match values stored in the colData. By default, no methods will be executed to update results. To actually execute updates, set dryrun = FALSE.

If a BenchDesign object is specified in addition to a SummarizedBenchmark object, the function will check which methods in the new BenchDesign need to be executed to update the SummarizedBenchmark results. Again, by default, no methods will be executed unless dryrun = FLASE is specified.

Unless reuseParams = FALSE is specified, the parameters of the last execution session stored in the the colData of the SummarizedBenchmark object will be used.

Usage

```
updateBench(sb, bd = NULL, dryrun = TRUE, version = FALSE,
  keepAll = TRUE, reuseParams = TRUE, ...)
```

Arguments

sb	a SummarizedBenchmark object
bd	a BenchDesign object
dryrun	logical whether to just print description of what would be updated rather than actually running any methods. (default = TRUE)
version	logical whether to re-run methods with only package version differences. (default = FALSE)
keepAll	logical whether to keep methods run in original SummarizedBenchmark but not in new BenchDesign. Only used if bd is not NULL. (default = TRUE)

reuseParams	logical whether to reuse parameters from buildBench call used to create SummarizedBenchmark object (if available). Directly specified buildBench parameters still take precedence. (default = TRUE)
...	optional parameters to pass to buildBench.

Value

SumamrizedBenchmark object.

Author(s)

Patrick Kimes

See Also

buildBench

Examples

```
## load example SummarizedBenchmark object
data(allSB)
sb <- allSB[[1]]

## check if results are out of date
updateBench(sb)

## modify BenchDesign
bd <- BenchDesign(sb)
bd <- dropMethod(bd, "kallisto-default")

## check if results need to be updated with new BenchDesign
updateBench(sb, bd)
```

Index