Package 'MultiDataSet'

April 15, 2017

Type Package

Title Implementation of the BRGE's (Bioinformatic Research Group in Epidemiology from Center for Research in Environmental Epidemiology) MultiDataSet and MethylationSet

Version 1.2.0

Description Implementation of the BRGE's (Bioinformatic Research Group in Epidemiology from Center for Research in Environmental Epidemiology) MultiDataSet and MethylationSet. Multi-DataSet
is designed for integrating multi omics data sets and MethylationSet to contain normal-ized methylation data.
These package contains base classes for MEAL and rexposome packages.

License file LICENSE

LazyData TRUE

biocViews Software, DataRepresentation

Depends R (>= 3.3), Biobase

Imports BiocGenerics, GenomicRanges, IRanges, minfi, S4Vectors, SummarizedExperiment, methods, IlluminaHumanMethylation450kanno.ilmn12.hg19, utils

RoxygenNote 5.0.1

Suggests MEALData, minfiData, knitr, rmarkdown, testthat, methylumi, omicade4, iClusterPlus, GEOquery

VignetteBuilder knitr

NeedsCompilation no

Author Carlos Ruiz-Arenas [aut, cre],
Carles Hernandez-Ferrer [aut],
Juan R. Gonz<c3><a1>lez [aut]

Maintainer Carlos Ruiz-Arenas <carlos.ruiz@isglobal.org>

R topics documented:

`add_eset` *Method to add an* `eSet` *to* `MultiDataSet`.

Description

This method adds or overwrites a slot of a `MultiDataSet` with the content of the given `eSet`.

Usage

```
add_eset(object, set, dataset.type, dataset.name = NULL, warnings = TRUE,
  overwrite = FALSE, GRanges)
```

Arguments

`object`	`MultiDataSet` that will be filled.
`set`	Object derived from `eSet` to be used to fill the slot.
`dataset.type`	Character with the type of data of the omic set (e.g. expression, methylation...)
`dataset.name`	Character with the specific name for this set (NULL by default). It is useful when there are several sets of the same type (e.g. multiple expression assays)
`warnings`	Logical to indicate if warnings will be displayed.
`overwrite`	Logical to indicate if the set stored in the slot will be overwritten.
`GRanges`	`GenomicRanges` to be included in rowRanges slot.

Value

A new `MultiDataSet` with a slot filled.

See Also

`add_methy`, `add_genexp`, `add_rnaseq`, `add_snps`

Examples

```
multi <- createMultiDataSet()
eset <- new("ExpressionSet", exprs = matrix(runif(10), 5))
multi <- add_eset(multi, eset, "exampledata", GRanges = NA)
```

add_genexp — *Method to add an expression microarray dataset to* MultiDataSet.

Description

This method adds or overwrites the slot "expression" of an MultiDataSet with the content of the given ExpressionSet. The fData of the ExpressionSet must contain the columns chromosome, start and end.

Usage

```
add_genexp(object, gexpSet, ...)
```

Arguments

object	MultiDataSet that will be filled.
gexpSet	ExpressionSet to be used to fill the slot.
...	Arguments to be passed to add_eset.

Value

A new MultiDataSet with the slot "expression" filled.

Examples

```
multi <- createMultiDataSet()
eset <- new("ExpressionSet", exprs = matrix(runif(4), 2))
fData(eset) <- data.frame(chromosome = c("chr1", "chr2"), start = c(12414, 1234321),
 end = c(121241, 124124114), stringsAsFactors = FALSE)
multi <- add_genexp(multi, eset)
```

add_methy — *Method to add a slot of methylation to* MultiDataSet.

Description

This method adds or overwrites the slot "methylation" of an MultiDataSet with the content of the given MethylationSet or RatioSet. The fData of the input object must contain the columns chromosome and position.

Usage

```
add_methy(object, methySet, ...)
```

Arguments

object	MultiDataSet that will be filled.
methySet	MethylationSet or RatioSet to be used to fill the slot.
...	Further arguments to be passed to add_eset.

Value

A new `MultiDataSet` with the slot `"methylation"` filled.

Examples

```
if (require(MEALData)){
 multi <- createMultiDataSet()
 betavals <- betavals[1:100, ]  ## To speed up the example, the beta values are reduced
 methy <- prepareMethylationSet(betavals, pheno)
 multi <- add_methy(multi, methy)
}
```

`add_rnaseq` *Method to add an expression RNA seq dataset to* `MultiDataSet`.

Description

This method adds or overwrites the slot `"rnaseq"` of an `MultiDataSet` with the content of the given `ExpressionSet`. The fData of the `ExpressionSet` must contain the columns chromosome, start and end.

Usage

```
add_rnaseq(object, rnaSet, ...)
```

Arguments

`object`	`MultiDataSet` that will be filled.
`rnaSet`	`ExpressionSet` to be used to fill the slot.
`...`	Arguments to be passed to `add_eset`.

Value

A new `MultiDataSet` with the slot `"rnaseq"` filled.

Examples

```
multi <- createMultiDataSet()
eset <- new("ExpressionSet", exprs = matrix(runif(4), 2))
fData(eset) <- data.frame(chromosome = c("chr1", "chr2"), start = c(12414, 1234321),
 end = c(121241, 12122414), stringsAsFactors = FALSE)
multi <- add_genexp(multi, eset)
```

add_rse *Method to add a* RangedSummarizedExperiment *to* MultiDataSet.

Description

This method adds or overwrites a slot of a MultiDataSet with the content of the given RangedSummarizedExperiment.

Usage

```
add_rse(object, set, dataset.type, dataset.name = NULL, warnings = TRUE,
  overwrite = FALSE)
```

Arguments

object	MultiDataSet that will be filled.
set	Object derived from RangedSummarizedExperiment to be used to fill the slot.
dataset.type	Character with the type of data of the omic set (e.g. expression, methylation...)
dataset.name	Character with the specific name for this set (NULL by default). It is useful when there are several sets of the same type (e.g. multiple expression assays)
warnings	Logical to indicate if warnings will be displayed.
overwrite	Logical to indicate if the set stored in the slot will be overwritten.

Value

A new MultiDataSet with a slot filled.

Examples

```
if (require(GenomicRanges) & require(SummarizedExperiment)){
multi <- createMultiDataSet()
counts <- matrix(runif(200 * 6, 1, 1e4), 200)
rowRanges <- GRanges(rep(c("chr1", "chr2"), c(50, 150)),
                     IRanges(floor(runif(200, 1e5, 1e6)), width=100),
                     strand=sample(c("+", "-"), 200, TRUE),
                     feature_id=sprintf("ID%03d", 1:200))
colData <- DataFrame(Treatment=rep(c("ChIP", "Input"), 3),
                    row.names=LETTERS[1:6], id = LETTERS[1:6])
names(rowRanges) <- 1:200
rse <- SummarizedExperiment(assays=SimpleList(counts=counts),
                            rowRanges=rowRanges, colData=colData)
multi <- add_rse(multi, rse, "rseEx")
}
```

`add_snps` *Method to add a slot of SNPs to* `MultiDataSet`.

Description

This method adds or overwrites the slot `"snps"` of an `MultiDataSet` with the content of the given `SnpSet`. The fData of the `SnpSet` must contain the columns chromosome and position.

Usage

```
add_snps(object, snpSet, ...)
```

Arguments

`object`	`MultiDataSet` that will be filled.
`snpSet`	`SnpSet` to be used to fill the slot.
`...`	Arguments to be passed to `add_eset`.

Value

A new `MultiDataSet` with the slot `"snps"` filled.

Examples

```
multi <- createMultiDataSet()
geno <- matrix(c(3,1,2,1), ncol = 2)
colnames(geno) <- c("VAL0156", "VAL0372")
rownames(geno) <- c("rs3115860", "SNP1-1628854")
map <- AnnotatedDataFrame(data.frame(chromosome = c("chr1", "chr2"), position = c(12414, 1234321),
     stringsAsFactors = FALSE))
rownames(map) <- rownames(geno)
snpSet <- new("SnpSet", call = geno, featureData = map)
pheno <- data.frame(id = c("VAL0156", "VAL0372"))
rownames(pheno) <- c("VAL0156", "VAL0372")
pData(snpSet) <- pheno
multi <- add_snps(multi, snpSet)
```

`checkProbes` *Filter* `MethylationSet` *probes*

Description

This function selects probes present in the annotation matrix. Probes without annotation and annotation values without beta values are discarded.

Usage

```
checkProbes(object)
```

Arguments

`object`	`MethylationSet`

Value

`MethylationSet` containing the common samples.

Examples

```
if (require(MEALData)){
 betavals <- betavals[1:100, ]  ## To speed up the example, the beta values are reduced
 methy <- prepareMethylationSet(betavals, pheno)
 checkProbes(methy)
}
```

`checkSamples` *Modify a* `MethylationSet` *to only contain common samples*

Description

This function removes samples that have beta values but no phenotypes and vice versa. If snps object is present, only samples present in the three set are retained.

Usage

```
checkSamples(object)
```

Arguments

`object`	`MethylationSet`

Value

`MethylationSet` containing the common samples.

Examples

```
if (require(MEALData)){
 betavals <- betavals[1:100, ]  ## To speed up the example, the beta values are reduced
 methy <- prepareMethylationSet(betavals, pheno)
 checkSamples(methy)
}
```

chrNumToChar *Convert chr numbers to chr strings*

Description

Given a vector of number representing the chromosomes, convert them to string (e.g 1 to chr1). 23 is consider chrX, 24 is chrY, 25 is chrXY (probes shared between chromosomes X and Y) and 26 is chrMT.

Usage

```
chrNumToChar(vector)
```

Arguments

vector	The vector with the chromosome numbers

Value

A vector with the chromosomes in string format.

Examples

```
chromosomes <- c(1, 3, 4, 23, 15)
stringChrs <- chrNumToChar(chromosomes)
stringChrs
```

commonIds *Get the name of the ids common to all datasets*

Description

Get the name of the ids common to all datasets

Usage

```
commonIds(object)
```

Arguments

object	`MultiDataSet` that will be filtered.

Value

Character vector with the common ids.

Examples

```
multi <- createMultiDataSet()
eset <- new("ExpressionSet", exprs = matrix(runif(9), ncol = 3))
fData(eset) <- data.frame(chromosome = c("chr1", "chr1", "chr1"),
                          start = c(1, 5, 10),end = c(4, 6, 14),
                          stringsAsFactors = FALSE)
sampleNames(eset) <- c("S1", "S2", "S3")
pData(eset) <- data.frame(id = c("S1", "S2", "S3"))
rownames(pData(eset)) <- c("S1", "S2", "S3")
multi <- add_genexp(multi, eset, dataset.name = "g1")
eset <- new("ExpressionSet", exprs = matrix(runif(8), ncol = 2))
fData(eset) <- data.frame(chromosome = c("chr1", "chr1", "chr1", "chr1"),
                          start = c(1, 14, 25, 104),end = c(11, 16, 28, 115),
                          stringsAsFactors = FALSE)
sampleNames(eset) <- c("S1", "G2")
pData(eset) <- data.frame(id = c("S1", "G2"))
rownames(pData(eset)) <- c("S1", "G2")

multi <- add_genexp(multi, eset, dataset.name="g2")
commonIds(multi)
```

---

commonSamples — *Method to select samples that are present in all datasets.*

Description

This method subsets the datasets to only contain the samples that are in all datasets.

Usage

```
commonSamples(object)
```

Arguments

`object`	`MultiDataSet` that will be filtered.

Value

A new `MultiDataSet` with only the common samples.

Examples

```
multi <- createMultiDataSet()
eset <- new("ExpressionSet", exprs = matrix(runif(9), ncol = 3))
fData(eset) <- data.frame(chromosome = c("chr1", "chr1", "chr1"),
                          start = c(1, 5, 10),end = c(4, 6, 14),
                          stringsAsFactors = FALSE)
sampleNames(eset) <- c("S1", "S2", "S3")
pData(eset) <- data.frame(id = c("S1", "S2", "S3"))
rownames(pData(eset)) <- c("S1", "S2", "S3")
multi <- add_genexp(multi, eset, dataset.name = "g1")
eset <- new("ExpressionSet", exprs = matrix(runif(8), ncol = 2))
```

```
fData(eset) <- data.frame(chromosome = c("chr1", "chr1", "chr1", "chr1"),
                          start = c(1, 14, 25, 104),end = c(11, 16, 28, 115),
                          stringsAsFactors = FALSE)
sampleNames(eset) <- c("S1", "G2")
pData(eset) <- data.frame(id = c("S1", "G2"))
rownames(pData(eset)) <- c("S1", "G2")

multi <- add_genexp(multi, eset, dataset.name="g2")
commonSamples(multi)
```

---

getMs *Transforms beta values to M-values*

Description

Given a MethylationSet or a AnalysisResults returns the matrix of M values using a logit2 transformation. Betas equal to 0 will be transformed to threshold and betas equal to 1, to 1 - threshold.

Usage

```
getMs(object, threshold = 1e-04)
```

Arguments

object	MethylationSet or AnalysisResults
threshold	Numeric with the threshold to avoid 0s and 1s.

Value

Matrix with the M values.

Examples

```
if (require(minfiData)){
set <- prepareMethylationSet(MsetEx[1:100, ], pData(MsetEx))
mvalues <- getMs(set)
head(mvalues)
}
```

MethylationSet *MethylationSet instances*

Description

Container with the data needed to perform methylation analysis. MethylationSet inherits from eSet and contains meth matrix as assay data member.

Usage

```
methylationSet(betas, phenotypes, annotationDataFrame, annoString = "custom")

## S4 method for signature 'MethylationSet'
betas(object)

## S4 method for signature 'MethylationSet'
getMs(object, threshold = 1e-04)

## S4 method for signature 'MethylationSet'
checkProbes(object)

## S4 method for signature 'MethylationSet'
checkSamples(object)
```

Arguments

`betas`	Matrix of beta values
`phenotypes`	Data.frame or AnnotatedDataFrame with the phenotypes
`annotationDataFrame`	Data.frame or AnnotatedDataFrame with the phenotypes with the annotation of the methylation sites. A column with the chromosomes named chr and a column with the positions names pos are required.
`annoString`	Character with the name of the annotation used.
`object`	`MethylationSet`
`threshold`	Numeric with the threshold to avoid 0s and 1s.

Details

FeatureData, which contains annotation data, is required to perform any of the analysis.

Value

`MethylationSet`

Methods (by generic)

- `betas`: Get beta matrix
- `getMs`: Get Ms values
- `checkProbes`: Filter probes with annotation
- `checkSamples`: Modify a `MethylationSet` to only contain common samples

Slots

`assayData` Contains matrices with equal dimensions, and with column number equal to nrow(phenoData). assayData must contain a matrix meth with rows representing features (e.g., methylation probes sets) and columns representing samples.

`phenoData` See eSet

`annotation` See eSet

`featureData` See eSet. fData should contain at least chromosome and positions columns.

Examples

```
showClass("MethylationSet")
```

`MultiDataSet` *MultiDataSet: Implementation of the BRGE's basic classes*

Description

Implementation of the BRGE's (Bioinformatic Research Group in Epidemiology from Center for Research in Environmental Epidemiology) MultiDataSet and MethylationSet. MultiDataSet is designed for integrating multi omics data sets and MethylationSet to contain normalized methylation data. MultiDataSet for integrating multi omics data sets

See Also

MultiDataSet

`MultiDataSet-class` *MultiDataSet instances*

Description

The class `MultiDataSet` is a superior class to store multiple datasets in form of triplets (assayData-phenoData-featureData). The datasets must be `eSet` or `SummarizedExperiment`.

Usage

```
## S4 method for signature 'MultiDataSet,eSet'
add_eset(object, set, dataset.type,
  dataset.name = NULL, warnings = TRUE, overwrite = FALSE, GRanges)

## S4 method for signature 'MultiDataSet,ExpressionSet'
add_genexp(object, gexpSet, ...)

## S4 method for signature 'MultiDataSet,ExpressionSet'
add_rnaseq(object, rnaSet, ...)

## S4 method for signature 'MultiDataSet,MethylationSet'
add_methy(object, methySet, ...)

## S4 method for signature 'MultiDataSet,RatioSet'
add_methy(object, methySet, ...)

## S4 method for signature 'MultiDataSet,RangedSummarizedExperiment'
add_rse(object, set,
  dataset.type, dataset.name = NULL, warnings = TRUE, overwrite = FALSE)

## S4 method for signature 'MultiDataSet,SnpSet'
add_snps(object, snpSet, ...)
```

```
## S4 method for signature 'MultiDataSet'
as.list(x)

## S4 method for signature 'MultiDataSet'
assayData(object)

## S4 method for signature 'MultiDataSet'
commonIds(object)

## S4 method for signature 'MultiDataSet'
commonSamples(object)

createMultiDataSet()

## S4 method for signature 'MultiDataSet'
fData(object)

## S4 method for signature 'MultiDataSet'
w_iclusterplus(object, commonSamples = TRUE, ...)

## S4 method for signature 'MultiDataSet'
length(x)

## S4 method for signature 'MultiDataSet'
w_mcia(object, ...)

## S4 method for signature 'MultiDataSet'
names(x)

## S4 method for signature 'MultiDataSet'
rowRangesElements(object)

## S4 method for signature 'MultiDataSet'
sampleNames(object)

## S4 method for signature 'MultiDataSet'
pData(object)

## S4 method for signature 'MultiDataSet'
rowRanges(x)

## S4 method for signature 'MultiDataSet,ANY,ANY'
x[[i]]

## S4 method for signature 'MultiDataSet,ANY,ANY,ANY'
x[i, j, k, ..., drop = FALSE]

## S4 method for signature 'MultiDataSet'
subset(x, feat, phe, warnings = TRUE, keep = TRUE)
```

Arguments

`object`	`MultiDataSet`
`set`	Object derived from `eSet` to be used to fill the slot.
`dataset.type`	Character with the type of data of the omic set (e.g. expression, methylation...)
`dataset.name`	Character with the specific name for this set (NULL by default). It is useful when there
`warnings`	Logical to indicate if warnings will be displayed.
`overwrite`	Logical to indicate if the set stored in the slot will be overwritten.
`GRanges`	`GenomicRanges` to be included in rowRanges slot.
`gexpSet`	`ExpressionSet` to be used to fill the slot.
`...`	Further arguments passed to `add_eset`.
`rnaSet`	`ExpressionSet` to be used to fill the slot.
`methySet`	`MethylationSet` to be used to fill the slot.
`snpSet`	`SnpSet` to be used to fill the slot.
`x`	`MultiDataSet`
`commonSamples`	Logical to indicate if common samples are selected
`i`	Character corresponding to selected sample names. They should match the id column of phenoData.
`j`	Character with the name of the selected tables.
`k`	`GenomicRange` used to filter the features.
`drop`	If TRUE, sets with no samples or features will be discarded
`feat`	Logical expression indicating features to keep
`phe`	Logical expression indicating the phenotype of the samples to keep
`keep`	If FALSE, sets where the expression cannot be evaluated will be discarded.

Details

The names of the three lists (`assayData`, `phenoData` and `featureData`) must be the same.

Value

`MultiDataSet`

Methods (by generic)

- `add_eset`: Method to add an `eSet` to `MultiDataSet`.
- `add_genexp`: Method to add a slot of expression to `MultiDataSet`.
- `add_rnaseq`: Method to add a slot of (RNASeq) expression to `MultiDataSet`.
- `add_methy`: Method to add a slot of methylation to `MultiDataSet`.
- `add_methy`: Method to add a slot of methylation to `MultiDataSet`.
- `add_rse`: Method to add a `RangedSummarizedExperiment` to `MultiDataSet`.
- `add_snps`: Method to add a slot of SNPs to `MultiDataSet`.
- `as.list`: Returns a list with the first matrix of each dataset.
- `assayData`: Retrieve all assay data blocks.

- `commonIds`: Get the name of the ids common to all datasets
- `commonSamples`: Get a MultiDataSet only with the samples present in all the tables
- `fData`: Retrieve information on features.
- `w_iclusterplus`: Apply iClusterPlus clustering method to a MultiDataSet object
- `length`: Returns the number of sets into the object.
- `w_mcia`: Apply mcia integration method to a MultiDataSet object
- `names`: Get the names of the slots.
- `rowRangesElements`: Get the name of the datasets that have rowRanges
- `sampleNames`: Get sample names
- `pData`: Retrieve information on experimental phenotypes.
- `rowRanges`: Retrieve information on feature ranges.
- `[[`: Get a set from a slot
- `[`: Subset a MultiDataSet
- `subset`: Filter a subset using feature ids or phenotypes

Slots

`assayData` List of `assayData` elements.

`phenoData` List of `AnnotatedDataFrame` containing the phenoData of each `assayData`.

`featureData` List of `AnnotatedDataFrame` containing the featureData of each `assayData`.

`rowRanges` List of `GenomicRanges` containing the rowRanges of each `assayData`.

`return_method` List of functions used to create the original object.

See Also

`add_eset`, `add_rse`

Examples

```
createMultiDataSet()
```

`prepareMethylationSet` *Generating a* `MethylationSet`

Description

This function creates a `MethylationSet` using from a matrix of beta values and a data.frame of phenotypes.

Usage

```
prepareMethylationSet(matrix, phenotypes,
  annotation = "IlluminaHumanMethylation450kanno.ilmn12.hg19",
  chromosome = "chr", position = "pos", genes = "UCSC_RefGene_Name",
  group = "UCSC_RefGene_Group", filterNA_threshold = 0.05,
  verbose = FALSE)
```

Arguments

`matrix`	Data.frame or a matrix with samples on the columns and cpgs on the rows. A `minfi` object can be used to.
`phenotypes`	Data.frame or vector with the phenotypic features of the samples. Samples will be in the rows and variables in the columns. If matrix is a `minfi` object, phenotypes can be taken from it.
`annotation`	Character with the name of the annotation package or data.frame or AnnotationDataFrame with the annotation.
`chromosome`	Character with the column containing chromosome name in the annotation data.
`position`	chromosome Character with the column containing position coordinate in the annotation data.
`genes`	Character with the column containing gene names related to the methylation site in the annotation data. (Optional)
`group`	Character with the column containing the position of the probe related to the gene named in gene column. (Optional)
`filterNA_threshold`	Numeric with the maximum percentage of NA allowed for each of the probes. If 1, there will be no filtering, if 0 all probes containing at least a NA will be filtered.
`verbose`	Logical value. If TRUE, it writes out some messages indicating progress. If FALSE nothing should be printed.

Details

prepareMethylationSet is a useful wrapper to create `MethylationSet`. Rigth now, prepareMethylationSet supports two entry points: a `minfi` object and a matrix of betas.

Phenotypes are compulsory and can be supplied as data.frame or AnnotatedDataFrame.

By default, annotation is taken from `minfi` package and `IlluminaHumanMethylation450kanno.ilmn12.hg19` package is used, being the default arguments adapted to use this annotation. To use this annotation, `IlluminaHumanMethylation450kanno.ilmn12.hg19` must be installed and methylation sites must be named like in Illumina 450k chip. Use of this annotation ensures correct results in all the analysis.

If custom annotation is desired, there are two compulsory features: chromosomes and positions. Chromosomes should be supplied in the character form (e.g. chr1). Two additional features will be used during the presentation of results but not during the analyses: genes and group. Genes are the gene names of the genes around the cpg site and group defines the groups of the genes. Both columns will appear in the results but they are not used through the workflow. It should be noticed that `BlockFinder` only supports `minfi` annotation, so it is not advised to be used with custom annotation.

Value

`MethylationSet` with phenotypes and annotation.

Examples

```
if (require(minfiData)){
 betas <- getBeta(MsetEx)[1:1000, ]
 pheno <- pData(MsetEx)
 set <- prepareMethylationSet(betas, pheno)
 }
```

`rowRangesElements` *Get the name of the datasets that have rowRanges*

Description

Get the name of the datasets that have rowRanges

Usage

```
rowRangesElements(object)
```

Arguments

`object`	`MultiDataSet`

Value

Character vector with the slots that have rowRanges.

Examples

```
multi <- createMultiDataSet()
eset <- new("ExpressionSet", exprs = matrix(runif(10), 5))
eset2 <- new("ExpressionSet", exprs = matrix(runif(8), ncol = 2))
fData(eset2) <- data.frame(chromosome = c("chr1", "chr1", "chr1", "chr1"),
                           start = c(1, 14, 25, 104),end = c(11, 16, 28, 115),
                           stringsAsFactors = FALSE)
multi <- add_eset(multi, eset, "exampledata", GRanges = NA)
multi <- add_genexp(multi, eset2)
rowRangesElements(multi)
```

`w_iclusterplus` *Apply iClusterPlus clustering method to a MultiDataSet object*

Description

Method iClusterPlus is applied on a MultiDataSet object after getting the common samples along all the contained datasets.

Usage

```
w_iclusterplus(object, commonSamples = TRUE, ...)
```

Arguments

`object`	`MultiDataSet`
`commonSamples`	Logical to indicate if common samples are selected
`...`	Arguments passed to function iClusterPlus

Value

A list of results from iClusterPlus

Note

Argument `type` for iClusterPlus is filled within the method.

`w_mcia` *Apply mcia integration method to a MultiDataSet object*

Description

Method mcia is applied on a MultiDataSet object after getting the common samples along all the contained datasets.

Usage

```
w_mcia(object, ...)
```

Arguments

`object`	`MultiDataSet`
`...`	Arguments passed to function mcia

Value

A list of results from mcia

Index