Package 'MultiAssayExperiment'

October 18, 2017

Title Software for the integration of multi-omics experiments in Bioconductor

Version 1.2.1

Author MultiAssay SIG

Description Develop an integrative environment where multiple assays are managed and preprocessed for genomic data analysis.

Depends R (>= 3.4.0)

Imports methods, GenomicRanges (>= 1.25.93), BiocGenerics, SummarizedExperiment (>= 1.3.81), S4Vectors, IRanges, Biobase, reshape2, shinydashboard, shiny, stats, tidyr, UpSetR, utils

Suggests BiocStyle, testthat, knitr, GenomicFiles, HDF5Array, RaggedExperiment, rmarkdown

Maintainer Marcel Ramos <marcel.ramosperez@roswellpark.org>

biocViews Infrastructure, DataRepresentation

License Artistic-2.0

LazyData true

LazyLoad yes

VignetteBuilder knitr

URL https://github.com/waldronlab/MultiAssayExperiment/wiki/MultiAssayExperiment-API

BugReports https://github.com/waldronlab/MultiAssayExperiment/issues

Collate 'API.R' 'ExperimentList-class.R' 'MultiAssayExperiment-class.R' 'RangedRaggedAssay-class.R' 'MultiAssayExperiment-methods.R' 'MultiAssayExperiment-helpers.R' 'MultiAssayExperiment-pkg.R' 'MultiAssayExperiment.R' 'assay-methods.R' 'clusterOn.R' 'hasAssay.R' 'listToMap.R' 'mapToList.R' 'prepMultiAssay.R' 'upsetSamples.R'

RoxygenNote 6.0.1

NeedsCompilation no

R topics documented:

`MultiAssayExperiment-package`

MultiAssayExperiment: Build an integrative multi-assay container

---

Description

MultiAssayExperiment allows the manipulation of related multiassay datasets with partially overlapping samples, associated metadata at the level of an entire study, and at the level of the "biological unit". The biological unit may be a patient, plant, yeast strain, etc.

Details

The package hierarchy of information:

- study
- experiments
- samples

See Also

Useful links:

- https://github.com/waldronlab/MultiAssayExperiment/wiki/MultiAssayExperiment-API
- Report bugs at https://github.com/waldronlab/MultiAssayExperiment/issues

API *Refer to the API documentation*

Description

API opens a browser to the API documentation

Usage

```
API(website = TRUE, shiny = FALSE)
```

Arguments

website	(logical default TRUE) launch the API website
shiny	(logical default FALSE) whether to launch the shiny version of the API (experimental)

Value

Documentation via the GitHub wiki

Author(s)

Vincent J Carey

Examples

```
## Runnable example does nothing

API(website = FALSE)
```

assay,RangedRaggedAssay,missing-method *Create a Matrix of score values using a GRanges or own ranges*

Description

This function can take a GRanges argument and use each range to check for overlaps with any of the current ranges in the first argument and return a score value from the corresponding metadata. This function will only operate on fully disjoint ranges (see isDisjoint for details). It can only work if metadata is present and there is a "score" column in the metadata. Please see example on how to add metadata to a RangedRaggedAssay or GRangesList class. This function uses the overlapsAny function from the GenomicRanges package.

Usage

```
## S4 method for signature 'RangedRaggedAssay,missing'
assay(x, i, mcolname = "score",
  background = NA, make.names = FALSE, ranges = NULL, type = "any", ...)
```

Arguments

x	A RangedRaggedAssay or GRangesList class
i	Argument set to missing (not used)
mcolname	A single string indicating the metadata column to use for the values in the resulting assay matrix
background	A default background value for the resulting assay matrix (default NA). This works for non-matching sample and range pairs in the data and will be imputed in the matrix (e.g., 2 for diploid genomes)
make.names	logical (default FALSE) whether to create character format ranges for the rows of the matrix (either from the ranges argument or from the RangedRaggedAssay itself). Example character format: "chr1:2-3:+"
ranges	An optional GRanges object for comparing accross all sample ranges and for superseding the rows for the resulting matrix (default NULL)
type	The type argument from overlapsAny
...	Unused argument

Value

A matrix of values from the score column of the metadata.

See Also

overlapsAny

Examples

```
example("RangedRaggedAssay")

## Add some phony metadata to the RangedRaggedAssay
metadata(myRRA) <- list(snparrray1 = DataFrame(score = 1),
snparray2 = DataFrame(score = 1),
snparray3 = DataFrame(score = 3))

assay(myRRA, background = 2)
```

---

clusterOn *Check expression of a given feature against clinical variable*

Description

Function that outputs a DataFrame with participant ID, sample ID, the select colData column, the expression values for select rownames, and the center values for each gene by cluster.

Usage

```
clusterOn(MultiAssayExperiment, colDataCols, rownames, experiments,
  seed = NULL)
```

Arguments

MultiAssayExperiment	A MultiAssayExperiment object
colDataCols	Select columns from the MultiAssayExperiment colData DataFrame
rownames	Features to be used for clustering (e.g., a set of gene names)
experiments	A character vector indicating assays of interest in the ExperimentList
seed	A single integer value passed to set.seed (default NULL)

Value

A DataFrame with appended cluster and center values

Examples

```
example(MultiAssayExperiment)
clusterOn(myMultiAssayExperiment, colDataCols = "sex",
    rownames = c("XIST", "RPS4Y1", "KDM5D"),
    experiments = "RNASeqGene", seed = 42L)
```

ExperimentList — *Construct an* ExperimentList *object for the* MultiAssayExperiment *object slot.*

Description

The ExperimentList class can contain several different types of data. The only requirements for an ExperimentList class are that the objects contained have the following set of methods: dim, [, rownames, colnames

Usage

```
ExperimentList(x)
```

Arguments

x	A list class object

Value

A ExperimentList class object of experiment data

Examples

```
## Create an empty ExperimentList instance
ExperimentList()

## Create array matrix and AnnotatedDataFrame to create an ExpressionSet class
arraydat <- matrix(seq(101, 108), ncol = 4,
                   dimnames = list(
                     c("ENST00000294241", "ENST00000355076"),
```

```
                    c("array1", "array2", "array3", "array4")
                  ))

arraypdat <- as(data.frame(
    slope53 = rnorm(4),
    row.names = c("array1", "array2", "array3", "array4")),
    "AnnotatedDataFrame")

## ExpressionSet constructor
exprdat <- Biobase::ExpressionSet(assayData = arraydat, phenoData = arraypdat)

## Create a sample methylation dataset
methyldat <- matrix(1:10, ncol = 5, dimnames = list(
    c("ENST00000355076", "ENST00000383706"),
    c("methyl1", "methyl2", "methyl3",
      "methyl4", "methyl5")))

## Create a sample RNASeqGene dataset
rnadat <- structure(c(46851, 5, 19, 13, 2197, 507, 84318, 126, 17, 21,
                      23979, 614), .Dim = 3:4, .Dimnames =
                        list(c("XIST", "RPS4Y1", "KDM5D"),
                             c("samparray1", "samparray2", "samparray3",
                               "samparray4")))

## Combine to a named list and call the ExperimentList constructor function
ExpList <- list(Affy = exprdat, Methyl450k = methyldat, RNASeqGene = rnadat)

## Use the ExperimentList constructor
myExperimentList <- ExperimentList(ExpList)
```

`ExperimentList-class` *A container for multi-experiment data*

Description

The `ExperimentList` class is a container that builds on the `SimpleList` with additional checks for consistency in experiment names and length. It contains a `SimpleList` of experiments with sample identifiers. One element present per experiment performed.

Usage

```
## S4 method for signature 'ANY'
ExperimentList(x)

## S4 method for signature 'missing'
ExperimentList(x)

## S4 method for signature 'ExperimentList'
show(object)

## S4 method for signature 'ExperimentList'
dimnames(x)
```

```
## S4 method for signature 'ExperimentList'
mergeReplicates(x, replicates = list(),
  simplify = BiocGenerics::mean, ...)

## S4 method for signature 'ANY,missing'
assay(x, i, ...)

## S4 method for signature 'ExperimentList'
assays(x, ..., withDimnames = TRUE)

## S4 method for signature 'ExperimentList,missing'
assay(x, i, ...)

## S4 method for signature 'ExperimentList,numeric'
assay(x, i, ...)

## S4 method for signature 'ExperimentList,character'
assay(x, i, ...)
```

Arguments

x	constructor: A list object. For mergeReplicates or assay: an ExperimentList object
object	An ExperimentList object
replicates	mergeReplicates: A list or LogicalList where each element represents a sample and a vector of repeated measurements for the sample
simplify	A function for merging columns where duplicates are indicated by replicates
...	Additional arguments. See details for more information.
i	A scalar character or integer index
withDimnames	logical (default TRUE) whether to return dimension names

Details

Convert from SimpleList or list to the multi-experiment data container. When using the **mergeReplicates** method, additional arguments are passed to the given simplify function argument (e.g., na.rm = TRUE)

Value

An ExperimentList object

Methods (by generic)

- ExperimentList: Create an ExperimentList object from an "ANY" class object, mainly list
- ExperimentList: Create an empty ExperimentList for signature "missing"
- show: Show method for ExperimentList class
- dimnames: Get the dimension names for an ExperimentList using CharacterList
- mergeReplicates: Apply the mergeReplicates method on the ExperimentList elements
- assay: Get the assay data for the default ANY class

- assays: Get the assay data from each element in the ExperimentList
- assay: Convenience function for the assay of the first element
- assay: Obtain the specified assay from ExperimentList with a numeric index
- assay: Get the specified assay from ExperimentList with a character index

Examples

```
ExperimentList()
```

---

hasAssay *Checking assay method for any class*

Description

The hasAssay function is intended for developers who would like to include new classes into a MultiAssayExperiment instance. It checks the methods tables of the assay function for the specified class of the argument.

Usage

```
hasAssay(object)
```

Arguments

object	A MultiAssayExperiment or named list object instance

Value

A logical value indicating method availability

Examples

```
lst <- structure(list(), .Names=character())
hasAssay(lst)
```

---

listToMap *Convert map from data.frame or DataFrame to list and vice versa*

Description

The mapToList function provides a convenient way of reordering a data.frame to a list. The listToMap function does the opposite by taking a list and converting it to DataFrame.

Usage

```
listToMap(listmap, type = "colnames")

mapToList(dfmap, assayCol = "assay")
```

Arguments

listmap	A named list object containing either experiments (assays), samples (colnames) or features (rownames)
type	Any of the valid types of maps including colnames, rownames, and assays.
dfmap	A data.frame or DataFrame object with identifiers in the first column
assayCol	A character vector of length one indicating the assay names column

Value

A DataFrame class object of names

A list object of DataFrames for each assay

Functions

- listToMap: Inverse of the listToMap function

Examples

```
example("MultiAssayExperiment")

## Create a sampleMap from a list using the listToMap function
mySampleMap <- listToMap(mylist)

## The inverse operation is also available
mylist <- mapToList(mySampleMap)
```

MultiAssayExperiment *Construct a* MultiAssayExperiment *object*

Description

The constructor function for the MultiAssayExperiment-class combines multiple data elements from the different hierarchies of data (study, experiments, and samples). It can create instances where neither a sampleMap or a colData set is provided. Please see the MultiAssayExperiment API documentation for more information by running the API function.

Usage

```
MultiAssayExperiment(experiments = ExperimentList(),
  colData = S4Vectors::DataFrame(), sampleMap = S4Vectors::DataFrame(),
  metadata = NULL, drops = list())
```

Arguments

experiments	A list or ExperimentList of all combined experiments
colData	A DataFrame or data.frame of characteristics for all biological units
sampleMap	A DataFrame or data.frame of assay names, sample identifiers, and colname samples
metadata	An optional argument of "ANY" class (usually list) for content describing the experiments
drops	A list of unmatched information (included after subsetting)

Value

A `MultiAssayExperiment` object that can store experiment and phenotype data

See Also

MultiAssayExperiment-class

Examples

```
## Run the example ExperimentList
example("ExperimentList")

## Load example GRangesList object
example("RangedRaggedAssay")

## Add the RangedRaggedAssay to the list
ExpList <- c(ExpList, CNVgistic = myRRA)

## Create sample maps for each experiment
exprmap <- data.frame(
    primary = c("Jack", "Jill", "Barbara", "Bob"),
    colname = c("array1", "array2", "array3", "array4"),
    stringsAsFactors = FALSE)

methylmap <- data.frame(
    primary = c("Jack", "Jack", "Jill", "Barbara", "Bob"),
    colname = c("methyl1", "methyl2", "methyl3", "methyl4", "methyl5"),
    stringsAsFactors = FALSE)

rangemap <- data.frame(
    primary = c("Jack", "Jill", "Jill"),
    colname = c("snparray1", "snparray2", "snparray3"),
    stringsAsFactors = FALSE)

rnamap <- data.frame(
    primary = c("Jack", "Jill", "Bob", "Barbara"),
    colname = c("samparray1", "samparray2", "samparray3",
                                "samparray4"),
    stringsAsFactors = FALSE)

## Combine as a named list and convert to a DataFrame
mylist <- list(Affy = exprmap, Methyl450k = methylmap,
               CNVgistic = rangemap, RNASeqGene = rnamap)

## Create a sampleMap
mySampleMap <- listToMap(mylist)
## Create an example phenotype data
colDat <- data.frame(sex = c("M", "F", "M", "F"),
                     age = 38:41,
                     row.names = c("Jack", "Jill", "Bob", "Barbara"))

## Create a MultiAssayExperiment instance
myMultiAssayExperiment <- MultiAssayExperiment(experiments = ExpList,
                                               colData = colDat,
                                               sampleMap = mySampleMap)
```

MultiAssayExperiment-class

An integrative multi-assay class for experiment data

Description

The MultiAssayExperiment class can be used to manage results of diverse assays on a collection of specimen. Currently, the class can handle assays that are organized instances of SummarizedExperiment, ExpressionSet, matrix, RangedRaggedAssay (inherits from GRangesList), and RangedVcfStack. Create new MultiAssayExperiment instances with the homonymous constructor, minimally with the argument ExperimentList, potentially also with the arguments colData (see section below) and sampleMap.

Usage

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

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

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

## S4 method for signature 'MultiAssayExperiment'
updateObject(object, ..., verbose = FALSE)

## S4 method for signature 'MultiAssayExperiment'
dimnames(x)

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

## S4 method for signature 'MultiAssayExperiment'
isEmpty(x)

## S4 method for signature 'MultiAssayExperiment'
complete.cases(...)

## S4 method for signature 'MultiAssayExperiment'
c(x, ..., sampleMap = NULL, mapFrom = NULL)

## S4 method for signature 'MultiAssayExperiment'
assays(x, ..., withDimnames = TRUE)

## S4 method for signature 'MultiAssayExperiment,missing'
assay(x, i, ...)

## S4 method for signature 'MultiAssayExperiment,numeric'
assay(x, i, ...)
```

```
## S4 method for signature 'MultiAssayExperiment,character'
assay(x, i, ...)
```

Arguments

object	A MultiAssayExperiment class object
x	A MultiAssayExperiment object for subsetting
...	Additional arguments. See details for more information.
verbose	logical (default FALSE) whether to print extra messages
i	subsetting: Either a character, or GRanges object for subsetting by rows, assay: unused argument (missing)
j	Either a character, logical, or numeric vector for subsetting by columns
k	Either a character, logical, or numeric vector for subsetting by assays
drop	logical (default TRUE) whether to drop empty assay elements in the ExperimentList
sampleMap	c method: a sampleMap list or DataFrame to guide merge
mapFrom	Either a logical, character, or integer vector indicating the experiment(s) that have an identical colname order as the experiment input(s)
withDimnames	logical (default TRUE) whether to return dimension names included in the object

Details

The dots (...) argument allows the user to specify additional arguments in several instances. When subsetting ([) a MultiAssayExperiment, the dots allow for additional arguments to be sent to findOverlaps. When using the mergeReplicates method, the dots are used to specify arguments for the supplied simplify argument and function. When using the **assay** method, additional arguments may be passed to the RangedRaggedAssay method. See the link for more information: assay,RangedRaggedAssay,missing-method. When using **c** method to add experiments to a MultiAssayExperiment, the dots allow extra data classes compatible with the MultiAssayExperiment API. See: API

Value

A MultiAssayExperiment object

Methods (by generic)

- show: Show method for a MultiAssayExperiment
- length: Get the length of ExperimentList
- names: Get the names of the ExperimentList
- updateObject: Update old serialized MultiAssayExperiment objects to new API
- dimnames: Get the dimension names for a MultiAssayExperiment object
- [: Subset a MultiAssayExperiment object
- isEmpty: A logical value indicating an empty MultiAssayExperiment
- complete.cases: Return a logical vector of biological units with data across all experiments
- c: Add an element to the ExperimentList data slot
- assays: Obtain a SimpleList of assay data for all available experiments in the object
- assay: Convenience function for extracting the assay of the first element in the ExperimentList
- assay: Obtain the specified assay from the MultiAssayExperiment with a numeric index
- assay: Get the specified assay from the MultiAssayExperiment with a character index

Slots

`ExperimentList` A `ExperimentList` class object for each assay dataset

`colData` A `DataFrame` of all clinical/specimen data available across experiments

`sampleMap` A `DataFrame` of translatable identifiers of samples and participants

`metadata` Additional data describing the `MultiAssayExperiment` object

`drops` A metadata `list` of dropped information

colData

The `colData` slot is a collection of primary specimen data valid across all experiments. This slot is strictly of class `DataFrame` but arguments for the constructor function allow arguments to be of class `data.frame` and subsequently coerced.

ExperimentList

The `ExperimentList` slot is designed to contain results from each experiment/assay. It contains a SimpleList.

sampleMap

The `sampleMap` contains a `DataFrame` of translatable identifiers of samples and participants or biological units. Standard column names of the sampleMap are "assay", "primary", and "colname".

See Also

MultiAssayExperiment-methods for slot modifying methods

Examples

```
example("MultiAssayExperiment")

## Subsetting
# Rows (i) Rows/Features in each experiment
myMultiAssayExperiment[1, , ]
myMultiAssayExperiment[c(TRUE, FALSE), , ]

# Columns (j) Rows in colData
myMultiAssayExperiment[, rownames(colData(myMultiAssayExperiment))[3:2],  ]

# Assays (k)
myMultiAssayExperiment[, , "Affy"]

## Complete cases (returns logical vector)
completes <- complete.cases(myMultiAssayExperiment)
compMAE <- myMultiAssayExperiment[, completes, ]
compMAE
colData(compMAE)

example("MultiAssayExperiment")

## Add an experiment
test1 <- myMultiAssayExperiment[[1L]]
colnames(test1) <- rownames(colData(myMultiAssayExperiment))
```

```
## Combine current MultiAssayExperiment with additional experiment
## (no sampleMap)
c(myMultiAssayExperiment, newExperiment = test1)

test2 <- myMultiAssayExperiment[[1L]]
c(myMultiAssayExperiment, newExp = test2, mapFrom = 3L)
```

`MultiAssayExperiment-helpers` *A group of helper functions for manipulating and cleaning a MultiAssayExperiment*

Description

A set of helper functions were created to help clean and manipulate a MultiAssayExperiment object.

- complete.cases: Returns a logical vector corresponding to 'colData' rows that have data across all experiments
- duplicated: Returns a 'list' of 'LogicalList's that indicate what measurements originate from the same biological unit
- intersectRows: Takes all common rows across experiments, excludes experiments with empty rownames
- intersectColumns: A wrapper for complete.cases to return a MultiAssayExperiment with only those biological units that have measurements across all experiments
- mergeReplicates: A function that combines duplicated / repeated measurements across all experiments and is guided by the duplicated return value
- longFormat: A `MultiAssayExperiment` method that returns a small and skinny DataFrame. The `colDataCols` arguments allows the user to append `colData` columns to the data.
- wideFormat: A function to return a wide DataFrame where each row represents an observation. Optional `colDataCols` can be added when using a `MultiAssayExperiment`.

Usage

```
intersectRows(x)

intersectColumns(x)

## S4 method for signature 'MultiAssayExperiment'
duplicated(x, incomparables = FALSE, ...)

mergeReplicates(x, replicates = list(), simplify = BiocGenerics::mean, ...)

## S4 method for signature 'MultiAssayExperiment'
mergeReplicates(x, replicates = list(),
  simplify = BiocGenerics::mean, ...)

## S4 method for signature 'ANY'
mergeReplicates(x, replicates = list(),
  simplify = BiocGenerics::mean, ...)
```

```
longFormat(object, ...)

## S4 method for signature 'ANY'
longFormat(object, ...)

## S4 method for signature 'ExperimentList'
longFormat(object, ...)

## S4 method for signature 'MultiAssayExperiment'
longFormat(object, colDataCols = NULL, ...)

wideFormat(object, ...)

## S4 method for signature 'MultiAssayExperiment'
wideFormat(object, colDataCols = NULL, ...)

## S4 method for signature 'ExperimentList'
wideFormat(object, ...)

## S4 method for signature 'ANY'
wideFormat(object, ...)
```

Arguments

x	A MultiAssayExperiment
incomparables	unused argument
...	Additional arguments. See details.
replicates	A list of LogicalLists indicating multiple / duplicate entries for each biological unit, see the duplicated output
simplify	A function for merging repeat measurements in experiments as indicated by replicates for MultiAssayExperiment
object	Any supported class object
colDataCols	selected colData columns to include in the output

Details

The mergeReplicates function is a house-keeping method for a MultiAssayExperiment where only complete.cases are returned, replicate measurements are averaged (by default), and columns are aligned by the row order in colData. Additional arguments can be passed on to the simplify function.

The mergeReplicates "ANY" method consolidates duplicate measurements for rectangular data structures, returns object of the same class (endomorphic)

The longFormat "ANY" class method, works with classes such as ExpressionSet and SummarizedExperiment as well as matrix to provide a consistent long and skinny DataFrame.

mergeReplicates

The mergeReplicates function makes use of the output from duplicated which will point out the duplicate measurements by biological unit in the MultiAssayExperiment. This function will return a MultiAssayExperiment with merged replicates.

longFormat

The longFormat method takes data from the ExperimentList in a MultiAssayExperiment and returns a uniform DataFrame. The resulting DataFrame has columns indicating primary, rowname, colname and value. This method can optionally include colData columns with the colDataCols argument (MultiAssayExperiment method only).

wideFormat

The wideFormat MultiAssayExperiment method returns standardized wide DataFrame where each row represents an observation or biological unit as represented in colData. Optionally, colData columns can be added to the data output. The wideFormat method for an ExperimentList returns a list of wideFormat DataFrames. The "ANY" method returns a wide format DataFrame.

MultiAssayExperiment-methods

Accessing/modifying slot information

Description

A set of accessor and setter generic functions to extract either the sampleMap, the ExperimentList, colData, or metadata slots of a MultiAssayExperiment object

Usage

```
## S4 method for signature 'MultiAssayExperiment'
sampleMap(x)

## S4 method for signature 'MultiAssayExperiment'
experiments(x)

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

## S4 method for signature 'MultiAssayExperiment'
colData(x, ...)

## S4 method for signature 'MultiAssayExperiment'
metadata(x)

## S4 replacement method for signature 'MultiAssayExperiment,DataFrame'
sampleMap(object) <- value

## S4 replacement method for signature 'MultiAssayExperiment,ExperimentList'
experiments(object) <- value

## S4 replacement method for signature 'MultiAssayExperiment,DataFrame'
pData(object) <- value

## S4 replacement method for signature 'MultiAssayExperiment,DataFrame'
colData(x) <- value
```

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

## S4 replacement method for signature 'MultiAssayExperiment'
x$name <- value

## S4 method for signature 'MultiAssayExperiment'
x$name

## S4 method for signature 'MultiAssayExperiment,ANY,ANY'
x[[i, j, ...]]

## S4 replacement method for signature 'MultiAssayExperiment,ANY,ANY'
x[[i, j, ...]] <- value
```

Arguments

x	A MultiAssayExperiment object
object	A MultiAssayExperiment object
...	Argument not in use
value	See details.
name	A column in colData
i	A numeric or character vector of length 1
j	Argument not in use

Value

Accessors: Either a sampleMap, ExperimentList, or DataFrame object

Setters: A MultiAssayExperiment object

Accessors

Eponymous names for accessing MultiAssayExperiment slots with the exception of the ExperimentList accessor named experiments.

- colData: Access the colData slot
- sampleMap: Access the sampleMap slot
- experiments: Access the ExperimentList slot
- '[[': Access the ExperimentList slot
- '$': Access a column in colData
- pData: (deprecated) Access the colData slot

Setters

Setter method values (i.e., 'function(x) <- value'):

- experiments<-: An ExperimentList object containing experiment data of supported classes
- sampleMap<-: A DataFrame object relating samples to biological units and assays
- colData<-: A DataFrame object describing the biological units
- metadata<-: A list object of metadata
- '[[<-': Equivalent to the experiments<- setter method for convenience
- '$<-': A vector to replace the indicated column in colData

Examples

```
## Load example MultiAssayExperiment
example(MultiAssayExperiment)

## Access the sampleMap
sampleMap(myMultiAssayExperiment)

## Replacement method for a MultiAssayExperiment sampleMap
sampleMap(myMultiAssayExperiment) <- DataFrame()

## Access the ExperimentList
experiments(myMultiAssayExperiment)

## Replace with an empty ExperimentList
experiments(myMultiAssayExperiment) <- ExperimentList()

## Access the metadata
metadata(myMultiAssayExperiment)

## Replace metadata with a list
metadata(myMultiAssayExperiment) <- list(runDate =
                                           format(Sys.time(), "%B %d, %Y"))

## Access a column in colData
myMultiAssayExperiment$age

## Replace a column in colData
myMultiAssayExperiment$age <- myMultiAssayExperiment$age + 1
```

---

prepMultiAssay *Prepare a* MultiAssayExperiment *instance*

Description

The purpose of this helper function is to faciltate the creation of a MultiAssayExperiment object by detecting any inconsistencies with all types of names in either the ExperimentList, the colData, or sampleMap.

Usage

```
prepMultiAssay(ExperimentList, colData, sampleMap)
```

Arguments

ExperimentList	A list of all combined experiments
colData	A DataFrame of the phenotype data for all participants
sampleMap	A DataFrame of sample identifiers, assay samples, and assay names

Value

A list containing all the essential components of a MultiAssayExperiment as well as a "drops" metadata element that indicates non-matched names. The names of the resulting list correspond to the arguments of the MultiAssayExperiment constructor function.

Checks

The prepMultiAssay function checks that all columns in the sampleMap are character.

It checks that all names and lengths match in both the ExperimentList and in the unique assay names of the sampleMap.

If ExperimentList names and assay names only differ by case and are not duplicated, the function will standardize all names to lowercase.

If names cannot be matched between the colname column of the sampleMap and the colnames of the ExperimentList, those unmatched will be dropped and found in the "drops" element of the resulting list.

Names in the "primary" column of the sampleMap, will be matched to those in the colData. Unmatched "primary" column rows will be dropped from the sampleMap. Suggestions for name fixes in either the ExperimentList or colnames will be made when necessary.

Examples

```
## Run example
example("MultiAssayExperiment")

## Check if there are any inconsistencies within the different names
preparedMAE <- prepMultiAssay(ExpList, colDat, mySampleMap)

## Results in a list of components for the MultiAssayExperiment constructor
## function
MultiAssayExperiment(preparedMAE$experiments, preparedMAE$colData,
preparedMAE$sampleMap)

## Alternatively, use the do.call function
do.call(MultiAssayExperiment, preparedMAE)
```

RangedRaggedAssay *Create a RangedRaggedAssay*

Description

Construct an object representing ranged-based data, typically from a GRangesList. The assay method will extract a particular column from the metadata and represent it in a matrix. See the show method for an example.

Usage

```
RangedRaggedAssay(x = GRangesList())
```

Arguments

x	A list, GRanges or GRangesList object

Value

A RangedRaggedAssay class object

Deprecated

The RangedRaggedAssay class is **deprecated** and defunct by the next release cycle. Please use the **RaggedExperiment** class to represent copy number, mutation and other genomic range based data. See RaggedExperiment for more detail.

See Also

assay,RangedRaggedAssay,missing-method

Examples

```
## Create an example GRangesList object
library(GenomicRanges)
gr1 <-
  GRanges(seqnames = "chr3", ranges = IRanges(58000000, 59502360),
          strand = "+", score = 5L, GC = 0.45)
gr2 <-
  GRanges(seqnames = c("chr3", "chr3"),
          ranges = IRanges(c(58493000, 3), width = 9000),
          strand = c("+", "-"), score = 3:4, GC = c(0.3, 0.5))
gr3 <-
  GRanges(seqnames = c("chr1", "chr2"),
          ranges = IRanges(c(1, 4), c(3, 9)),
          strand = c("-", "-"), score = c(6L, 2L), GC = c(0.4, 0.1))

grl <- GRangesList("gr1" = gr1, "gr2" = gr2, "gr3" = gr3)
names(grl) <- c("snparray1", "snparray2", "snparray3")

## Create a RangedRaggedAssay object class
myRRA <- RangedRaggedAssay(grl)
```

RangedRaggedAssay-class

An extension of the GRangesList class

Description

An extension of the GRangesList class

Subsetting a RangedRaggedAssay can be done using either rownames and column names

Usage

```
## S4 method for signature 'RangedRaggedAssay,ANY,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'RangedRaggedAssay,GRanges,ANY,ANY'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'RangedRaggedAssay'
dim(x)

## S4 method for signature 'RangedRaggedAssay'
```

```
dimnames(x)

## S4 replacement method for signature 'RangedRaggedAssay,list'
dimnames(x) <- value

## S4 method for signature 'RangedRaggedAssay'
disjoin(x, mcolname = NULL,
  simplify = BiocGenerics::mean, ...)

## S4 method for signature 'RangedRaggedAssay'
show(object)

## S4 method for signature 'RangedRaggedAssay'
longFormat(object, ...)

## S4 method for signature 'RangedRaggedAssay'
mergeReplicates(x, replicates = list(),
  simplify = BiocGenerics::mean, mcolname = NULL, ...)
```

Arguments

x	A RangedRaggedAssay class
i	Either a character or GRanges class object to subset by rows
j	Either a character, numeric, or logical type for selecting columns (GRangesList method)
...	Additional arguments. See details for more information.
drop	logical (default TRUE) whether to drop empty columns
value	A list object of row and column names
mcolname	A single character string indicating metadata column to use for summaries
simplify	A function for combining duplicate measurements (e.g., mean)
object	A RangedRaggedAssay class object
replicates	mergeReplicates: A list or LogicalList where each element represents a sample and a vector of repeated measurements for that biological unit

Details

The ... argument allows the user to specify arguments in the subsetByOverlaps function. When calling the mergeReplicates method, the additional arguments correspond to those in either the assay method or the mergeReplicates method. The mergeReplicates arguments include a function for applying over the rows (combine) and a vectorized argument which indicates whether the given function is vectorized or not.

Value

A RangedRaggedAssay class object

Methods (by generic)

- [: Subset a RangedRaggedAssay with either character, numeric, or logical
- [: Subset a RangedRaggedAssay using a GRanges class object

- `dim`: Obtain dimension lengths of a `RangedRaggedAssay` class object
- `dimnames`: Get dimension names for a `RangedRaggedAssay`
- `dimnames<-`: value: A modified `RangedRaggedAssay` object
- `disjoin`: Separate non-disjoint ranges and merge with function
- `show`: show method for the `RangedRaggedAssay` class
- `longFormat`: RangedRaggedAssay class method to return a DataFrame of selected "mcolname" column, defaults to score
- `mergeReplicates`: (deprecated) Use metadata column to produce a matrix which can then be merged across replicates.

See Also

`findOverlaps-methods`

assay,RangedRaggedAssay,missing-method

`subsetByAssay` *Subset* `MultiAssayExperiment` *object by Assay type*

Description

Select which assay(s) to obtain from available datasets

Usage

```
subsetByAssay(x, y)

## S4 method for signature 'MultiAssayExperiment'
subsetByAssay(x, y)
```

Arguments

`x`	A `MultiAssayExperiment` object
`y`	Either a `numeric`, `character` or `logical` object indicating what assay(s) to select

Value

A `MultiAssayExperiment` object

Methods (by class)

- `MultiAssayExperiment`: Use either a `numeric`, `logical`, or `character` vector to subset assays in a `MultiAssayExperiment`

See Also

'subset,MultiAssayExperiment-method'

Examples

```
## Load a MultiAssayExperiment example
example("MultiAssayExperiment")

## Using experiment names
subsetByAssay(myMultiAssayExperiment, "Affy")

## Using numeric indicators
subsetByAssay(myMultiAssayExperiment, 1:2)

## Using a logical vector
subsetByAssay(myMultiAssayExperiment, c(TRUE, FALSE, TRUE))
```

---

subsetByColData *Subset* MultiAssayExperiment *object by* colData *rows*

Description

Select biological units in a MultiAssayExperiment with subsetByColData

Usage

```
subsetByColData(x, y)

## S4 method for signature 'MultiAssayExperiment,ANY'
subsetByColData(x, y)

## S4 method for signature 'MultiAssayExperiment,character'
subsetByColData(x, y)
```

Arguments

x	A MultiAssayExperiment object
y	Either a numeric, character or logical object indicating what colData rows to select

Value

A MultiAssayExperiment object

Methods (by class)

- x = MultiAssayExperiment,y = ANY: Either a numeric, character, or logical vector to apply a column subset of a MultiAssayExperiment object
- x = MultiAssayExperiment,y = character: Use a character vector for subsetting column names

Examples

```
## Load a MultiAssayExperiment example
example("MultiAssayExperiment")

## Subset by character vector (Jack)
subsetByColData(myMultiAssayExperiment, "Jack")

## Subset by numeric index of colData rows (Jack and Bob)
subsetByColData(myMultiAssayExperiment, c(1, 3))

## Subset by logical indicator of colData rows (Jack and Jill)
subsetByColData(myMultiAssayExperiment, c(TRUE, TRUE, FALSE, FALSE))
```

---

subsetByColumn *Subset* MultiAssayExperiment *object*

Description

subsetByColumn returns a subsetted MultiAssayExperiment object

Usage

```
subsetByColumn(x, y)

## S4 method for signature 'MultiAssayExperiment,list'
subsetByColumn(x, y)

## S4 method for signature 'MultiAssayExperiment,List'
subsetByColumn(x, y)
```

Arguments

x	A MultiAssayExperiment object
y	Either a numeric, character or logical object indicating what rownames in the colData to select for subsetting

Value

A MultiAssayExperiment object

Methods (by class)

- x = MultiAssayExperiment,y = list: Use a list to subset by colname in a MultiAssayExperiment
- x = MultiAssayExperiment,y = List: Use an S4 List to subset a MultiAssayExperiment. The order of the subsetting elements in this List must match that of the ExperimentList in the MultiAssayExperiment.

Examples

```
## Load a MultiAssayExperiment example
example("MultiAssayExperiment")

subsetByColumn(myMultiAssayExperiment, list(Affy = 1:2,
                Methyl450k = c(3,5,2), RNASeqGene = 2:4, CNVgistic = 1))

subsetWith <- IRanges::mendoapply(`[`, colnames(myMultiAssayExperiment),
                         MoreArgs = list(1:2))
subsetByColumn(myMultiAssayExperiment, subsetWith)
```

---

subsetByRow *Subset* MultiAssayExperiment *object by Feature*

Description

Subset a MultiAssayExperiment class by provided feature names or a GRanges object

Usage

```
subsetByRow(x, y, ...)

## S4 method for signature 'MultiAssayExperiment,ANY'
subsetByRow(x, y, ...)

## S4 method for signature 'MultiAssayExperiment,list'
subsetByRow(x, y)

## S4 method for signature 'MultiAssayExperiment,List'
subsetByRow(x, y)
```

Arguments

x	A MultiAssayExperiment object
y	A character vector or GRanges class object containing feature names or ranges
...	Additional arguments to pass to low level subsetting function primarily when using a GRanges object for subsetting (via getHits)

Value

A MultiAssayExperiment object

Methods (by class)

- x = MultiAssayExperiment,y = ANY: Subset a MultiAssayExperiment with either a numeric or logical vector
- x = MultiAssayExperiment,y = list: Use a list of equal length as the ExperimentList to subset. The order of the subsetting elements in this list must match that of the ExperimentList in the MultiAssayExperiment.
- x = MultiAssayExperiment,y = List: Use an S4 List to subset a MultiAssayExperiment. The order of the subsetting elements in this List must match that of the ExperimentList in the MultiAssayExperiment.

Examples

```
## Load a MultiAssayExperiment example
example("MultiAssayExperiment")

## Use a GRanges object to subset rows where ranged data present
egr <- GenomicRanges::GRanges(seqnames = "chr1",
    IRanges::IRanges(start = 1, end = 3), strand = "-")
subsetByRow(myMultiAssayExperiment, egr)

## Use a logical vector (recycling used)
subsetByRow(myMultiAssayExperiment, c(TRUE, FALSE))

## Use a character vector
subsetByRow(myMultiAssayExperiment, "ENST00000355076")
```

`upsetSamples` *Create a generalized Venn Diagram analog for sample membership in multiple assays, using the upset algorithm in* `UpSetR`

Description

Create a generalized Venn Diagram analog for sample membership in multiple assays, using the upset algorithm in `UpSetR`

Usage

```
upsetSamples(MultiAssayExperiment, nsets = length(MultiAssayExperiment),
  nintersects = 24, order.by = "freq", idclip = function(x) substr(x, 1,
  12), ...)
```

Arguments

`MultiAssayExperiment`	instance of `MultiAssayExperiment-class`
`nsets`	integer number of sets to analyze
`nintersects`	Number of intersections to plot. If set to NA, all intersections will be plotted.
`order.by`	How the intersections in the matrix should be ordered by. Options include frequency (entered as "freq"), degree, or both in any order.
`idclip`	A function that operates on `colnames(MultiAssayExperiment)`, to remove potentially assay-specific token components; use `force` if no clipping is needed
`...`	parameters passed to `upset`

Value

Produces a visualization of set intersections using the UpSet matrix design

Author(s)

Vincent J Carey

Examples

```
example(MultiAssayExperiment)
upsetSamples(myMultiAssayExperiment, idclip = function(x) {
    gsub("[a-z]", "", x)
    })
```

Index