Package ‘MultiAssayExperiment’

March 23, 2017

Title Create Classes and Functions for Managing Multiple Assays on Sets of Samples

Version 1.0.1

Author MultiAssay SIG

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

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

Depends R (>= 3.3.0)

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

Maintainer Marcel Ramos <mramos09@gmail.com>

biocViews Infrastructure, DataRepresentation

License Artistic-2.0

LazyData true

LazyLoad yes

VignetteBuilder knitr

Collate 'API.R' 'ExperimentList-class.R'
  'MultiAssayExperiment-class.R' 'RangedRaggedAssay-class.R'
  'MultiAssayExperiment-methods.R' 'MultiAssayExperiment-pkg.R'
  'MultiAssayExperiment.R' 'PrepMultiAssay.R' 'assay-methods.R'
  'hasAssay.R' 'listToMap.R' 'mapToList.R' 'zzz.R'

RoxygenNote 6.0.1

NeedsCompilation no

R topics documented:

  API ........................................................................................................ 2
  assay,RangedRaggedAssay,ANY-method ............................................ 3
  ExperimentList .................................................................................. 4
  ExperimentList-class ...................................................................... 5
  experiments ..................................................................................... 6
  experiments<- ................................................................................. 6
  getHits ............................................................................................. 7
API

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)
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

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

Arguments

- `x`: A RangedRaggedAssay or GRangesList class
- `mcolname`: A single character string indicating the inner metadata column name to use for creating a matrix (must indicate a numeric variable)
- `ranges`: A GRanges class identifying the ranges of interest
- `background`: A single value for the non-matching background values in the matrix (e.g., 2 for diploid genomes)
- `make.names`: logical (default FALSE) whether to automatically create names from either the ranges argument (if available) or the RangedRaggedAssay (e.g., "chr1:2-3:+")

Value

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

Examples

```r
example("RangedRaggedAssay")

## Add some phony metadata to the RangedRaggedAssay
metadata(myRRA) <- list(snparray1 = DataFrame(score = 1),
                         snparray2 = DataFrame(score = 1),
                         snparray3 = DataFrame(score = 3))

assay(myRRA, background = 2)
```
**ExperimentList**

ExperimentList  

*experiment Acessor function for the ExperimentList slot of a MultiAssayExperiment object*

---

**Description**

*experiment Acessor function for the ExperimentList slot of a MultiAssayExperiment object*

**Usage**

`ExperimentList(x)`

**Arguments**

- **x**  
  A `codeMultiAssayExperiment` class object

**Value**

A `ExperimentList` class object of experiment data

**Examples**

```r
## 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")
exprdat <- Biobase::ExpressionSet(assayData=arraydat, phenoData=arraypdat)

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

## Combine to a named list and call the ExperimentList constructor function
ExpList <- list(exprdat, methyl.dat)
names(ExpList) <- c("Affy", "Methyl450k")
myExperimentList <- ExperimentList(ExpList)
```
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

```r
## 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,missing'
assay(x, i)

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

Arguments

- `x` A list object
- `object` An `ExperimentList` class object
- `i` missing argument

Details

Convert from `SimpleList` or `list` to the multi-experiment data container

Value

An `ExperimentList` class 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 a `MultiAssayExperiment` using `CharacterList`
experiments<-  

  • assay: Get the assay data for the default ANY class  
  • assay: Get the assay data from each element in the ExperimentList

Examples

ExperimentList()

experiments   Accessor function for the ExperimentList slot of a MultiAssayExperiment object

Description

Accessor function for the ExperimentList slot of a MultiAssayExperiment object

Usage

experiments(x)

Arguments

  x   A MultiAssayExperiment class object

Value

A ExperimentList object of assay data

Examples

example("MultiAssayExperiment")
experiments(myMultiAssayExperiment)

experiments<-   Replace an ExperimentList slot value with a given ExperimentList class object

Description

Replace an ExperimentList slot value with a given ExperimentList class object

Usage

experiments(object) <- value

Arguments

  object   A MultiAssayExperiment class object
  value    An ExperimentList object to replace the existing ExperimentList slot
getHits

Value

A ExperimentList class object

Examples

```r
## Load a MultiAssayExperiment
eexample("MultiAssayExperiment")

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

Description

Find hits by class type

Usage

```r
getHits(subject, query, ...) 
```

## S4 method for signature 'MultiAssayExperiment,character'
getHits(subject, query, ...)

## S4 method for signature 'MultiAssayExperiment,GRanges'
getHits(subject, query, ...)

## S4 method for signature 'GRanges,GRanges'
getHits(subject, query, ...)

## S4 method for signature 'ANY,GRanges'
getHits(subject, query, ...)

## S4 method for signature 'RangedSummarizedExperiment,GRanges'
getHits(subject, query, ...)

## S4 method for signature 'ANY,character'
getHits(subject, query, ...)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>subject</td>
<td>Any valid element from the ExperimentList class</td>
</tr>
<tr>
<td>query</td>
<td>Either a character vector or GRanges object used to search by name or ranges</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments to findOverlaps</td>
</tr>
</tbody>
</table>

Value

Names of matched queries
hasAssay

Methods (by class)

- subject = MultiAssayExperiment, query = character: Find all matching rownames by character
- subject = MultiAssayExperiment, query = GRanges: Find all matching rownames by GRanges
- subject = GRanges, query = GRanges: Find and get corresponding names of two GRanges using findOverlaps
- subject = ANY, query = GRanges: Find all matching rownames for range-based objects
- subject = RangedSummarizedExperiment, query = GRanges: Find rownames for RangedSummarizedExperiment hits
- subject = ANY, query = character: Find all matching rownames based on character query

Examples

```r
## Load an example MultiAssayExperiment object
eexample("MultiAssayExperiment")
eexample("GRangesList")

## Find what ranges fit the criteria (see findOverlaps)
getHits(myMultiAssayExperiment, gr1)
```

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

```r
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 list class object containing names of either experiments, assays or features.
- **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

```r
example("sampleMap")
```
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.

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

Usage

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

Arguments

- **experiments**: A list or ExperimentList of all combined experiments
- **pData**: A DataFrame or data.frame of the phenotype data for all participants
- **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 overall experiments.
- **drops**: A list of unmatched information (included after subsetting)

Details

The package hierarchy of information:

- study
- experiments
- samples

Value

A MultiAssayExperiment data object that stores 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, myRRA)
names(ExpList)[3] <- "CNVgistic"

## Run the sample map example
example("sampleMap")

## Create an example phenotype data
pDat <- 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,
                                              pData = pDat,
                                              sampleMap = mySampleMap)

MultiAssayExperiment-class

An integrative MultiAssay 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 eponymous constructor, minimally with the argument ExperimentList, potentially also with the arguments pData (see section below) and sampleMap.

Usage

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

## 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'
metadata(x)

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

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

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

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

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

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

## 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,missing'
assay(x, i)

Arguments

object
A MultiAssayExperiment class object

x
A MultiAssayExperiment object for subsetting

value
A DataFrame or ExperimentList object to replace the existing sampleMap, ExperimentList, or pData slot

... Additional arguments passed down to getHits support function for subsetting by rows
MultiAssayExperiment-class

name  pData column name
verbose (logical default FALSE) whether to output verbose
i Either a character, or GRanges object for subsetting by rows
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

Value
A MultiAssayExperiment object

Methods (by generic)
  • show: Show method for a MultiAssayExperiment
  • sampleMap: Access sampleMap slot from a MultiAssayExperiment
  • experiments: Access ExperimentList class from a MultiAssayExperiment
  • pData: Access pData slot from a MultiAssayExperiment
  • metadata: Access metadata slot from a MultiAssayExperiment
  • length: Get the length of ExperimentList
  • names: Get the names of the ExperimentList
  • sampleMap<-: value: A DataFrame sampleMap representation
  • experiments<-: value: An ExperimentList representation
  • pData<-: value: A DataFrame of specimen data
  • metadata<-: value: Data of type "ANY"
  • $<-, value: DataFrame column
  • updateObject: Update old serialized MultiAssayExperiment objects to new API
  • dimnames: Get the dimension names for a MultiAssayExperiment object
  • $: Access pData column
  • [: 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
  • assay: Get the assay data for a MultiAssayExperiment as a list

Slots
ExperimentList A ExperimentList class object for each assay dataset
pData 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

pData
The pData 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**

*getHits*

**Examples**

`MultiAssayExperiment()`

---

**PrepMultiAssay**

*Prepare a MultiAssayExperiment instance*

**Description**

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

**Usage**

`PrepMultiAssay(ExperimentList, pData, sampleMap)`

**Arguments**

- **ExperimentList**: A list of all combined experiments
- **pData**: 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" element that indicates non-matched names.

**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 assaynames only differ by case and are not '# duplicated, the function will standardize all names to lowercase.
If names cannot be matched between the assay 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 pData. 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

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

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

## Results in a list of components for the MultiAssayExperiment constructor
## function
MultiAssayExperiment(preparedMAE$ExperimentList, preparedMAE$pData, preparedMAE$sampleMap)
```

---

Create a RangedRaggedAssay

### Description

Create a RangedRaggedAssay

### Usage

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

### Arguments

- **x**  
  A list, GRanges or GRangesList object

### Value

A RangedRaggedAssay class object

### Examples

```r
## 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"),
```
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'
ncol(x)

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

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

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

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

## S4 method for signature 'RangedRaggedAssay,character'
getHits(subject, query, ...)

Arguments

x A RangedRaggedAssay class

i Either a character or GRanges class object to subset by rows

ranges = IRanges(c(1, 4), c(3, 9)),
strand = c("-", "-"), score = c(6L, 2L), GC = c(0.4, 0.1))
gr1 <- GRangesList("gr1" = gr1, "gr2" = gr2, "gr3" = gr3)
names(gr1) <- c("snparray1", "snparray2", "snparray3")

## Create a RangedRaggedAssay object class
myRRA <- RangedRaggedAssay(gr1)
j Either a character, numeric, or logical type for selecting columns (GRangesList method)
...
Any additional arguments passed on to subsetByOverlaps
drop logical (default TRUE) whether to drop empty columns
value A list object of row and column names
object A RangedRaggedAssay class object
subject A RangedRaggedAssay class object
query A character class for searching hits

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
• ncol: Get the column length of a RangedRaggedAssay class object
• nrow: Get the row length of a RangedRaggedAssay class object
• dimnames: Get dimension names for a RangedRaggedAssay
• dimnames< -: value: A modified RangedRaggedAssay object
• show: show method for the RangedRaggedAssay class
• getHits: Find matching features by character in a RangedRaggedAssay

See Also
findOverlaps-methods

---

**sampleMap**

**Accessor function for the sampleMap slot of a MultiAssayExperiment object**

Description

Accessor function for the sampleMap slot of a MultiAssayExperiment object

Usage

sampleMap(x)

Arguments

x A MultiAssayExperiment object

Value

A DataFrame object of sample relationships across experiments
## Create sample maps for each experiment

```r
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)
```

## Combine as a named list and convert to a DataFrame

```r
mylist <- list(exprmap, methylmap, rangemap)
names(mylist) <- c("Affy", "Methyl450k", "CNVgistic")
```

## Create a sampleMap

```r
mySampleMap <- listToMap(mylist)
```

---

**Description**

Replace a slot value with a given DataFrame

**Usage**

```r
sampleMap(object) <- value
```

**Arguments**

- `object` A `MultiAssayExperiment` object
- `value` A `DataFrame` object to replace the existing `sampleMap`

**Value**

A `sampleMap` with replacement values

**Examples**

```r
## Load example
df <- sampleData("MultiAssayExperiment")

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

**Description**

Select which assay(s) to obtain from available datasets

**Usage**

```r
subsetByAssay(x, y)
```

### S4 method for signature 'MultiAssayExperiment'

```r
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**

```r
## Load a MultiAssayExperiment example
e.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))
```
subsetByColumn

Subset MultiAssayExperiment object

Description
subsetByColumn returns a subsetted MultiAssayExperiment object

Usage

subsetByColumn(x, y)

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

## S4 method for signature 'MultiAssayExperiment,character'
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 pData to select for subsetting

Value

A MultiAssayExperiment object

Methods (by class)

• x = MultiAssayExperiment, y = ANY: Either a numeric or logical vector to apply a
column subset of a MultiAssayExperiment object

• x = MultiAssayExperiment, y = character: Use a character vector for subsetting col-
   umn names

• 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
eexample("MultiAssayExperiment")

## Subset by character vector (Jack)
ssubsetByColumn(myMultiAssayExperiment, "Jack")
## Subset by numeric index of pData rows (Jack and Bob)
subsetByColumn(myMultiAssayExperiment, c(1, 3))

## Subset by logical indicator of pData rows (Jack and Jill)
subsetByColumn(myMultiAssayExperiment, c(TRUE, TRUE, FALSE, FALSE))

---

**subsetByRow**

**Subset MultiAssayExperiment object by Feature**

### Description

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

### Usage

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

### S4 method for signature 'MultiAssayExperiment,GRangesORcharacter'

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

### S4 method for signature 'MultiAssayExperiment,GRanges'

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

### S4 method for signature 'MultiAssayExperiment,ANY'

```r
subsetByRow(x, y)  
```

### S4 method for signature 'MultiAssayExperiment,List'

```r
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 = GRangesORcharacter**: Use either a GRanges or character to select the rows for which to subset by
- **x = MultiAssayExperiment, y = GRanges**: Subset a MultiAssayExperiment with GRanges object
subsetByRow

• \( x = \text{MultiAssayExperiment}, y = \text{ANY} \): Subset a MultiAssayExperiment with either a numeric or logical vector

• \( x = \text{MultiAssayExperiment}, y = \text{list} \): 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.

• \( x = \text{MultiAssayExperiment}, y = \text{List} \): 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.

See Also
getHits

Examples

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

## Use a GRanges object to subset rows where ranged data present
egr <- GRanges(seqnames = "chr1", 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")
```
## Index

<table>
<thead>
<tr>
<th>Function</th>
<th>Class/Method</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>.ExperimentList</td>
<td>(ExperimentList-class)</td>
<td>5</td>
</tr>
<tr>
<td>.RangedRaggedAssay</td>
<td>(RangedRaggedAssay-class)</td>
<td>16</td>
</tr>
<tr>
<td>[</td>
<td>MultiAssayExperiment, ANY, ANY, ANY-method</td>
<td>11</td>
</tr>
<tr>
<td>[</td>
<td>MultiAssayExperiment, ANY-method</td>
<td>11</td>
</tr>
<tr>
<td>[</td>
<td>RangedRaggedAssay, ANY, ANY, ANY-method</td>
<td>16</td>
</tr>
<tr>
<td>[</td>
<td>RangedRaggedAssay, ANY-method</td>
<td>16</td>
</tr>
<tr>
<td>[</td>
<td>RangedRaggedAssay, GRanges, ANY, ANY-method</td>
<td>16</td>
</tr>
<tr>
<td>$</td>
<td>MultiAssayExperiment-method</td>
<td>11</td>
</tr>
<tr>
<td>$&lt;-$, MultiAssayExperiment-method</td>
<td>(MultiAssayExperiment-class)</td>
<td>11</td>
</tr>
<tr>
<td>API</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>assay, ANY, missing-method</td>
<td>(ExperimentList-class)</td>
<td>5</td>
</tr>
<tr>
<td>assay, ExperimentList, missing-method</td>
<td>(ExperimentList-class)</td>
<td>5</td>
</tr>
<tr>
<td>assay, MultiAssayExperiment, missing-method</td>
<td>(MultiAssayExperiment-class)</td>
<td>11</td>
</tr>
<tr>
<td>assay, RangedRaggedAssay, ANY-method</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>CharacterList.</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>complete.cases, MultiAssayExperiment-method</td>
<td>(MultiAssayExperiment-class)</td>
<td>11</td>
</tr>
<tr>
<td>DataFrame</td>
<td></td>
<td>10, 13, 14</td>
</tr>
<tr>
<td>dim, RangedRaggedAssay-method</td>
<td>(RangedRaggedAssay-class)</td>
<td>16</td>
</tr>
<tr>
<td>dimnames, MultiAssayExperiment-method</td>
<td>(MultiAssayExperiment-class)</td>
<td>11</td>
</tr>
<tr>
<td>dimnames, RangedRaggedAssay-method</td>
<td>(RangedRaggedAssay-class)</td>
<td>16</td>
</tr>
<tr>
<td>dimnames&lt;-, RangedRaggedAssay.list-method</td>
<td>(RangedRaggedAssay-class)</td>
<td>16</td>
</tr>
<tr>
<td>ExperimentList, 4, 5–7, 10, 11, 13–15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ExperimentList, ANY-method</td>
<td>(ExperimentList-class)</td>
<td>5</td>
</tr>
<tr>
<td>ExperimentList, missing-method</td>
<td>(ExperimentList-class)</td>
<td>5</td>
</tr>
<tr>
<td>ExperimentList-class</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>experiments, 6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>experiments&lt;-, MultiAssayExperiment, ExperimentList-method</td>
<td>(MultiAssayExperiment-class)</td>
<td>11</td>
</tr>
<tr>
<td>ExpressionSet</td>
<td></td>
<td>11</td>
</tr>
<tr>
<td>getHits, 7, 22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>getHits, ANY, character-method</td>
<td>(getHits)</td>
<td>7</td>
</tr>
<tr>
<td>getHits, ANY, GRanges-method</td>
<td>(getHits)</td>
<td>7</td>
</tr>
<tr>
<td>getHits, GRanges, GRanges-method</td>
<td>(getHits)</td>
<td>7</td>
</tr>
<tr>
<td>getHits, MultiAssayExperiment, character-method</td>
<td>(getHits)</td>
<td>7</td>
</tr>
<tr>
<td>getHits, MultiAssayExperiment, GRanges-method</td>
<td>(getHits)</td>
<td>7</td>
</tr>
<tr>
<td>getHits, RangedRaggedAssay, character-method</td>
<td>(RangedRaggedAssay-class)</td>
<td>16</td>
</tr>
<tr>
<td>getHits, RangedSummarizedExperiment, GRanges-method</td>
<td>(getHits)</td>
<td>7</td>
</tr>
<tr>
<td>GRanges, 3, 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GRangesList, 3, 11, 17</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hasAssay, 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>isEmpty, MultiAssayExperiment-method</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

23
(MultiAssayExperiment-class), 11
length,MultiAssayExperiment-method (MultiAssayExperiment-class), 11
listToMap, 9
mapToList(listToMap), 9
metadata,MultiAssayExperiment-method (MultiAssayExperiment-class), 11
metadata<-,MultiAssayExperiment-method (MultiAssayExperiment-class), 11
MultiAssayExperiment, 10,13,14,19–21
MultiAssayExperiment-class, 10,11
MultiAssayExperiment-package (MultiAssayExperiment), 10
names,MultiAssayExperiment-method (MultiAssayExperiment-class), 11
ncol,RangedRaggedAssay-method (RangedRaggedAssay-class), 16
nrow,RangedRaggedAssay-method (RangedRaggedAssay-class), 16
overlapsAny, 3
pData,MultiAssayExperiment-method (MultiAssayExperiment-class), 11
data<-,MultiAssayExperiment,DataFrame-method (MultiAssayExperiment-class), 11
PrepMultiAssay, 14
RangedRaggedAssay, 3,11,15,16,17
RangedRaggedAssay-class, 16
sampleMap, 11,14,15,17
sampleMap,MultiAssayExperiment-method (MultiAssayExperiment-class), 11
sampleMap<-, 18
sampleMap<-,MultiAssayExperiment,DataFrame-method (MultiAssayExperiment-class), 11
show,ExperimentList-method (ExperimentList-class), 5
show,MultiAssayExperiment-method (MultiAssayExperiment-class), 11
show,RangedRaggedAssay-method (RangedRaggedAssay-class), 16
SimpleList, 14
subsetByAssay, 19
subsetByAssay,MultiAssayExperiment-method (subsetByAssay), 19
subsetByColumn, 20
subsetByColumn,MultiAssayExperiment,ANY-method (subsetByColumn), 20
subsetByColumn,MultiAssayExperiment,character-method (subsetByColumn), 20
subsetByColumn,MultiAssayExperiment,List-method (subsetByColumn), 20
subsetByColumn,MultiAssayExperiment,list-method (subsetByColumn), 20
subsetByRow, 21
subsetByRow,MultiAssayExperiment,ANY-method (subsetByRow), 21
subsetByRow,MultiAssayExperiment,GRanges-method (subsetByRow), 21
subsetByRow,MultiAssayExperiment,GRangesORcharacter-method (subsetByRow), 21
subsetByRow,MultiAssayExperiment,List-method (subsetByRow), 21
subsetByRow,MultiAssayExperiment,list-method (subsetByRow), 21
SummarizedExperiment, 11
updateObject,MultiAssayExperiment-method (MultiAssayExperiment-class), 11