Package 'MultiAssayExperiment'

May 2, 2016

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

Version 0.99.199

Author MultiAssay SIG

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

Suggests BiocStyle, testthat, knitr

Depends R (>= 3.3.0)

Imports methods, GenomicRanges, BiocGenerics, SummarizedExperiment, S4Vectors, IRanges, Biobase, shinydashboard, shiny

Maintainer Marcel Ramos <mramos09@gmail.com>

biocViews Infrastructure, DataRepresentation

License Artistic-2.0

LazyData true

LazyLoad yes

VignetteBuilder knitr

Collate 'API.R' 'Elist-class.R' 'MultiAssayExperiment-class.R' 'RangedRaggedAssay-class.R' 'MultiAssayExperiment-methods.R' 'MultiAssayExperiment-pkg.R' 'MultiAssayExperiment.R' 'PrepMultiAssay.R' 'assayMatrix.R' 'hasAssay.R' 'listToMap.R' 'mapToList.R'

RoxygenNote 5.0.1

NeedsCompilation no

R topics documented:

API 2
assay,ExpressionSet,ANY-method 3
assay,matrix,ANY-method 3
assayMatrix 4
Elist 5

Elist-class 6
Elist<- 7
getHits 7
hasAssay 9
listToMap 9
MultiAssayExperiment 10
MultiAssayExperiment-class 12
PrepMultiAssay 14
RangedRaggedAssay 15
RangedRaggedAssay-class 16
sampleMap 18
sampleMap<- 19
subsetByAssay 20
subsetByColumn 21
subsetByRow 22

Index **24**

`API` *Refer to the API documentation*

Description

`API` opens a browser to the API documentation

Usage

```
API(shiny = FALSE)
```

Arguments

`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

```
## This will open in a browser window
## Not run:
API()

## End(Not run)
```

---

assay,ExpressionSet,ANY-method
Harmonize exprs to assay of an ExpressionSet *object*

Description

Harmonize exprs to assay of an ExpressionSet object

Usage

```
## S4 method for signature 'ExpressionSet,ANY'
assay(x)
```

Arguments

x	An ExpressionSet object

Value

A matrix of data

---

assay,matrix,ANY-method
Harmonize show to assay of a matrix *object*

Description

Harmonize show to assay of a matrix object

Usage

```
## S4 method for signature 'matrix,ANY'
assay(x)
```

Arguments

x	A matrix object

Value

A matrix of data

assayMatrix *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

```
assayMatrix(x, ranges = NULL, background = NA, use.names = FALSE)
```

Arguments

`x`	A RangedRaggedAssay or GRangesList class
`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)
`use.names`	logical (default FALSE) whether to use names in the given ranges argument or in the provided 'x' argument

Value

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

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))

assayMatrix(myRRA, background = 2)
```

Elist — *Elist Acessor function for the* `Elist` *slot of a* `MultiAssayExperiment` *object*

Description

Elist Acessor function for the `Elist` slot of a `MultiAssayExperiment` object

Usage

```
Elist(x)
```

Arguments

x	A codeMultiAssayExperiment class object

Value

A `Elist` class object of experiment data

Examples

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

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

## Combine to a named list and call the Elist constructor function
ExpList <- list(exprdat, methyldat)
names(ExpList) <- c("Affy", "Methyl450k")
myElist <- Elist(ExpList)
```

Elist-class *A container for multi-experiment data*

Description

The Elist 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'
Elist(x)

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

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

## S4 method for signature 'Elist'
rownames(x)

## S4 method for signature 'Elist'
colnames(x)
```

Arguments

x	A list object
object	An Elist class object

Details

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

Value

An Elist class object

Methods (by generic)

- Elist: Create an Elist object from an "ANY" class object, mainly list
- Elist: Create an empty Elist for signature "missing"
- show: Show method for Elist class
- rownames: Get all the rownames of an Elist
- colnames: Get sample names from an Elist object

Examples

```
Elist()
```

`Elist<-` *Replace an* `Elist` *slot value with a given* `Elist` *class object*

Description

Replace an `Elist` slot value with a given `Elist` class object

Usage

```
Elist(object) <- value
```

Arguments

`object`	A `MultiAssayExperiment` class object
`value`	An `Elist` object to replace the existing `Elist` slot

Value

A `Elist` class object

Examples

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

## Replace with an empty Elist
Elist(myMultiAssayExperiment) <- Elist()
```

`getHits` *Find hits by class type*

Description

Find hits by class type

Usage

```
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

subject	Any valid element from the Elist class
query	Either a character vector or GRanges object used to search by name or ranges
...	Additional arguments to findOverlaps

Value

Names of matched queries

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

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

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

`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 `list` object instance

Value

A `logical` value indicating method availability

Examples

```
char <- character()
hasAssay(char)
```

`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 = "assayname")
```

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

```
example("sampleMap")

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

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

MultiAssayExperiment *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 function combines multiple data elements from the different hierarchies of data (study, experiments, and samples)

Usage

```
MultiAssayExperiment(Elist = list(), pData = S4Vectors::DataFrame(),
  sampleMap = S4Vectors::DataFrame(), drops = list())
```

Arguments

Elist	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
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 Elist
example("Elist")

## 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
primary <- data.frame(sex = c("M", "F", "M", "F"),
                      age = 38:41,
                      row.names = c("Jack", "Jill", "Bob", "Barbara"))

## Create a MultiAssayExperiment instance
myMultiAssayExperiment <- MultiAssayExperiment(Elist = ExpList,
                                               pData = primary,
                                               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 Elist, 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'
Elist(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,Elist'
Elist(object) <- value

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

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

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

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

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

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

Arguments

object	A MultiAssayExperiment class object
x	A MultiAssayExperiment object for subsetting
value	A DataFrame or Elist object to replace the existing sampleMap, Elist, or pData slot
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
...	Additional arguments passed down to getHits support function for subsetting by rows
drop	logical (default TRUE) whether to drop empty assay elements in the Elist

Value

A MultiAssayExperiment object

Methods (by generic)

- show: Show method for a MultiAssayExperiment
- sampleMap: Access sampleMap slot from a MultiAssayExperiment
- Elist: Access Elist class from a MultiAssayExperiment
- pData: Access pData slot from a MultiAssayExperiment
- metadata: Access metadata slot from a MultiAssayExperiment
- length: Get the length of Elist
- names: Get the names of the Elist
- sampleMap<-: value: A DataFrame sampleMap representation
- Elist<-: value: An Elist representation
- pData<-: value: A DataFrame of specimen data
- rownames: Get all the rownames for a MultiAssayExperiment using a CharacterList
- colnames: Get all the colnames for a MultiAssayExperiment
- assay: Get the raw data from a MultiAssayExperiment as a list
- [: Subset a MultiAssayExperiment object
- isEmpty: A logical value indicating an empty MultiAssayExperiment

Slots

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

`pData` A `DataFrame` of all clinical 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.

Elist

The `Elist` 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 "primary", "assay", and "assayname".

See Also

`getHits`

Examples

```
MultiAssayExperiment()
```

`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 `Elist`, the `pData`, or `sampleMap`.

Usage

```
PrepMultiAssay(Elist, pData, sampleMap)
```

Arguments

Elist	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 Elist and in the unique assaynames of the sampleMap.

If Elist 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 Elist, 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 Elist or colnames will be made when necessary.

Examples

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

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

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

RangedRaggedAssay *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

```
## 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'
x[i, j, ..., drop = TRUE]

## S4 method for signature 'RangedRaggedAssay,GRanges'
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'
rownames(x)

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

## S4 method for signature 'RangedRaggedAssay,ANY'
assay(x)

## S4 replacement method for signature 'RangedRaggedAssay,character'
colnames(x) <- value

## S4 method for signature 'RangedRaggedAssay,ANY'
assay(x)

## 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
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 character vector representing column or sample names
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
- rownames: Get feature names from a RangedRaggedAssay
- colnames: Get sample names from a RangedRaggedAssay
- assay: Create a matrix of scores for RangedRaggedAssay with only disjoint ranges.
- colnames<-: value: A modified RangedRaggedAssay object
- assay: Get experiment metadata from a RangedRaggedAssay
- 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

Examples

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

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

rangemap <- data.frame(primary = c("Jack", "Jill", "Jill"),
```

```
    assay = c("snparray1", "snparray2", "snparray3"),
    stringsAsFactors = FALSE)

## Combine as a named list and convert to a DataFrame
mylist <- list(exprmap, methylmap, rangemap)
names(mylist) <- c("Affy", "Methyl450k", "CNVgistic")

## Create a sampleMap
mySampleMap <- listToMap(mylist)
```

---

sampleMap<- *Replace a slot value with a given* DataFrame

Description

Replace a slot value with a given DataFrame

Usage

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

Arguments

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

Value

A sampleMap with replacement values

Examples

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

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

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))
```

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 column names
- x = MultiAssayExperiment,y = list: Use a list to subset by samples 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 Elist in the MultiAssayExperiment.

Examples

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

## Subset by character vector (Jack)
subsetByColumn(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

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

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

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

## S4 method for signature 'MultiAssayExperiment,logical'
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 = GRangesORcharacter: Use either a GRanges or character to select the rows for which to subset for
- x = MultiAssayExperiment,y = GRanges: Subset a MultiAssayExperiment with GRanges object
- x = MultiAssayExperiment,y = logical: Use a logical vector to select rows of a MultiAssayExperiment
- 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 Elist to subset. The order of the subsetting elements in this list must match that of the Elist 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 Elist in the MultiAssayExperiment.

See Also

getHits

Examples

```
## Load a MultiAssayExperiment example
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

.Elist *(Elist-class)*, 6
.RangedRaggedAssay *(RangedRaggedAssay-class)*, 16
[,MultiAssayExperiment,ANY-method *(MultiAssayExperiment-class)*, 12
[,RangedRaggedAssay,ANY-method *(RangedRaggedAssay-class)*, 16
[,RangedRaggedAssay,GRanges-method *(RangedRaggedAssay-class)*, 16

API, 2
assay,ExpressionSet,ANY-method, 3
assay,matrix,ANY-method, 3
assay,MultiAssayExperiment,ANY-method *(MultiAssayExperiment-class)*, 12
assay,RangedRaggedAssay,ANY-method *(RangedRaggedAssay-class)*, 16
assayMatrix, 4

CharacterList, *13*
colnames,Elist-method *(Elist-class)*, 6
colnames,MultiAssayExperiment-method *(MultiAssayExperiment-class)*, 12
colnames,RangedRaggedAssay-method *(RangedRaggedAssay-class)*, 16
colnames<-,RangedRaggedAssay,character-method *(RangedRaggedAssay-class)*, 16

DataFrame, *11*, *14, 15*
dim,RangedRaggedAssay-method *(RangedRaggedAssay-class)*, 16

Elist, 5, *6*, *8*, *12*, *14, 15*
Elist,ANY-method *(Elist-class)*, 6
Elist,missing-method *(Elist-class)*, 6
Elist,MultiAssayExperiment-method *(MultiAssayExperiment-class)*, 12
Elist-class, 6
Elist<-, 7
Elist<-,MultiAssayExperiment,Elist-method *(MultiAssayExperiment-class)*, 12
ExpressionSet, *12*

getHits, 7, *23*
getHits,ANY,character-method *(getHits)*, 7
getHits,ANY,GRanges-method *(getHits)*, 7
getHits,GRanges,GRanges-method *(getHits)*, 7
getHits,MultiAssayExperiment,character-method *(getHits)*, 7
getHits,MultiAssayExperiment,GRanges-method *(getHits)*, 7
getHits,RangedRaggedAssay,character-method *(RangedRaggedAssay-class)*, 16
getHits,RangedSummarizedExperiment,GRanges-method *(getHits)*, 7
GRanges, *4*, *8*
GRangesList, *4*, *12*, *17*

hasAssay, 9

isEmpty,MultiAssayExperiment-method *(MultiAssayExperiment-class)*, 12

length,MultiAssayExperiment-method *(MultiAssayExperiment-class)*, 12
listToMap, 9

mapToList *(listToMap)*, 9
metadata,MultiAssayExperiment-method *(MultiAssayExperiment-class)*, 12
MultiAssayExperiment, 10, *14, 15*, *20–23*
MultiAssayExperiment-class, 12

MultiAssayExperiment-package (MultiAssayExperiment), 10

names,MultiAssayExperiment-method (MultiAssayExperiment-class), 12
ncol,RangedRaggedAssay-method (RangedRaggedAssay-class), 16
nrow,RangedRaggedAssay-method (RangedRaggedAssay-class), 16

overlapsAny, *4*

pData,MultiAssayExperiment-method (MultiAssayExperiment-class), 12
pData<-,MultiAssayExperiment,DataFrame-method (MultiAssayExperiment-class), 12
PrepMultiAssay, 14

RangedRaggedAssay, *4*, *12*, 15, *16, 17*
RangedRaggedAssay-class, 16
rownames,Elist-method (Elist-class), 6
rownames,MultiAssayExperiment-method (MultiAssayExperiment-class), 12
rownames,RangedRaggedAssay-method (RangedRaggedAssay-class), 16

sampleMap, *12*, *14, 15*, 18
sampleMap,MultiAssayExperiment-method (MultiAssayExperiment-class), 12
sampleMap<-, 19
sampleMap<-,MultiAssayExperiment,DataFrame-method (MultiAssayExperiment-class), 12
show,Elist-method (Elist-class), 6
show,MultiAssayExperiment-method (MultiAssayExperiment-class), 12
SimpleList, *14*
subsetByAssay, 20
subsetByAssay,MultiAssayExperiment-method (subsetByAssay), 20
subsetByColumn, 21
subsetByColumn,MultiAssayExperiment,ANY-method (subsetByColumn), 21
subsetByColumn,MultiAssayExperiment,character-method (subsetByColumn), 21
subsetByColumn,MultiAssayExperiment,List-method (subsetByColumn), 21
subsetByColumn,MultiAssayExperiment,list-method (subsetByColumn), 21
subsetByRow, 22
subsetByRow,MultiAssayExperiment,ANY-method (subsetByRow), 22
subsetByRow,MultiAssayExperiment,GRanges-method (subsetByRow), 22
subsetByRow,MultiAssayExperiment,GRangesORcharacter-method (subsetByRow), 22
subsetByRow,MultiAssayExperiment,List-method (subsetByRow), 22
subsetByRow,MultiAssayExperiment,list-method (subsetByRow), 22
subsetByRow,MultiAssayExperiment,logical-method (subsetByRow), 22
SummarizedExperiment, *12*