Package ‘MultiAssayExperiment’

March 23, 2017

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

Version  1.1.29

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 (>= 3.4.0)

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

Maintainer  Marcel Ramos <mramos09@gmail.com>

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-pkg.R'
  'MultiAssayExperiment.R' 'PrepMultiAssay.R' 'assay-methods.R'
  'clusterOn.R' 'hasAssay.R' 'listToMap.R' 'mapToList.R'
  'upsetSamples.R'

RoxygenNote  6.0.1

NeedsCompilation  no
R topics documented:

MultiAssayExperiment-package .............................................. 2
API ................................................................. 3
assay,RangedRaggedAssay,missing-method .......................... 3
clusterOn .......................................................... 4
ExperimentList ......................................................... 5
ExperimentList-class .................................................... 6
experiments ........................................................... 8
experiments<- ....................................................... 8
hasAssay ............................................................. 9
listToMap ............................................................. 9
MultiAssayExperiment .................................................. 10
MultiAssayExperiment-class ............................................. 11
PrepMultiAssay ....................................................... 16
RangedRaggedAssay ................................................... 17
RangedRaggedAssay-class ............................................. 18
rearrange .............................................................. 20
sampleMap ........................................................... 21
sampleMap<- ......................................................... 22
subsetByAssay ......................................................... 23
subsetByColumn ....................................................... 24
subsetBypData ........................................................ 25
subsetByRow .......................................................... 26
upsetSamples .......................................................... 27

Index 28

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](https://github.com/waldronlab/MultiAssayExperiment/wiki/MultiAssayExperiment-API)
- Report bugs at [https://github.com/waldronlab/MultiAssayExperiment/issues](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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>website</td>
<td>(logical default TRUE) launch the API website</td>
</tr>
<tr>
<td>shiny</td>
<td>(logical default FALSE) whether to launch the shiny version of the API (experimental)</td>
</tr>
</tbody>
</table>

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", ...)

clusterOn

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 across 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

```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)
```

Description

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

Usage

```r
clusterOn(MultiAssayExperiment, pDataCols, rownames, experiments, seed = NULL)
```
**ExperimentList**

**Arguments**

- **MultiAssayExperiment**
  A MultiAssayExperiment object
- **pDataCols**
  Select columns from the MultiAssayExperiment pData 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**

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

---

**ExperimentList**

Constructor function for the ExperimentList slot of a MultiAssayExperiment object.

**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**

```r
ExperimentList(x)
```

**Arguments**

- **x**
  A list 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
dat <- matrix(seq(101, 108), ncol = 4,
  dimnames = list(c("ENST00000294241", "ENST00000355076"),
```

```r
```
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'
reduce(x, drop.empty.ranges = FALSE,
repeats = NULL, combine = rowMeans, vectorized = TRUE, ...)
## S4 method for signature 'ANY,missing'
assay(x, i, ...)

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

### Arguments
- **x**: constructor: A list object, reduce or assay: an `ExperimentList` object
- **object**: An `ExperimentList` object
- **drop.empty.ranges**: unused argument
- **replicates**: reduce: A list or `LogicalList` where each element represents a sample and a vector of repeated experiments for the sample (default NULL)
- **combine**: reduce: A function for consolidating columns in the matrix representation of the data
- **vectorized**: reduce: (default TRUE) whether the combine function is vectorized, optimized for working down the vector pairs
- **...**: Additional arguments. See details for more information.
- **i**: assay: unused argument

### Details
Convert from `SimpleList` or list to the multi-experiment data container. When using the `reduce` method, additional arguments are passed to the given `combine` 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`
- `reduce`: Apply the reduce method on the `ExperimentList` elements
- `assay`: Get the assay data for the default ANY class
- `assay`: Get the assay data from each element in the `ExperimentList`

### Examples

```r
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

Value

A ExperimentList class object
### hasAssay

**Examples**

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

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

**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**

```r
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 `listToMap` 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**

```r
listToMap(listmap, type = "colnames")
mapToList(dfmap, assayCol = "assay")
```
MultiAssayExperiment

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  Construct a MultiAssayExperiment object

Description

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)
MultiAssayExperiment-class

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, myRRA)
names(ExpList)[4] <- "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 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 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,ANY,ANY'
x[[i, j, ...]]

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

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

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

## S4 method for signature 'MultiAssayExperiment'
reduce(x, drop.empty.ranges = FALSE, 
replicates = NULL, combine = rowMeans, vectorized = TRUE, ...)

## S4 method for signature 'ANY'
reduce(x, drop.empty.ranges = FALSE, replicates = NULL, 
combine = rowMeans, vectorized = TRUE, ...)

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

## S4 method for signature 'MultiAssayExperiment,missing'
assay(x, 1, ...)

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. See details for more information.
name pData column name
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
incomparables duplicated: unused argument
drop.empty.ranges unused generic argument
replicates reduce: A list of LogicalList indicating duplicate entries for each biological unit, 
see the duplicated method for MultiAssayExperiment
combine reduce: function for combining replicate columns/samples (default rowMeans)
vectorized reduce: logical (default TRUE) whether the combine function is vectorized, op-
timized for working down the vector pairs
sampleMap c method: a sampleMap list or DataFrame to guide merge

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 reduce method, the dots are used to specify arguments
for the supplied combine 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

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
- `is Empty`: A logical value indicating an empty `MultiAssayExperiment`
- `[]<-`: Extract the ExperimentList element
- `[[<-`: Replace the ExperimentList element with a supported class (should have documented dimensions in sampleMap)
- `complete.cases`: Return a logical vector of biological units with data across all experiments
- `duplicated`: Find duplicate columns in the data by matching pData rownames
- `reduce`: Housekeeping method for a `MultiAssayExperiment` where only complete.cases are returned, replicate measurements are averaged, and columns are aligned by the row order in pData.
- `reduce`: Consolidate columns for rectangular data structures, mainly matrix
- `c`: Add an element to the ExperimentList data slot
- `assay`: Get the assay data for a `MultiAssayExperiment` as a list
MultiAssayExperiment-class

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".

**Examples**

e.example("MultiAssayExperiment")

## Subsetting

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

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

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

## Complete cases (returns logical vector)
completes <- complete.cases(myMultiAssayExperiment)
compMAE <- myMultiAssayExperiment[, completes, ]
compMAE
pData(compMAE)
e.example("MultiAssayExperiment")

## Add an experiment

test <- myMultiAssayExperiment[[1L]]
colnames(test) <- rownames(pData(myMultiAssayExperiment))

test <- myMultiAssayExperiment[[2L]]
colnames(test) <- rownames(pData(myMultiAssayExperiment))

test <- myMultiAssayExperiment[[3L]]
colnames(test) <- rownames(pData(myMultiAssayExperiment))

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

## Run example
eexample("MultiAssayExperiment")

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

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

See Also

assay, RangedRaggedAssay, missing-method

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"),
              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

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

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

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

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

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

```r
## S4 method for signature 'RangedRaggedAssay'
disjoin(x, mcolname = NULL, FUN = mean, ...)
```

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

```r
## S4 method for signature 'RangedRaggedAssay'
reduce(x, drop.empty.ranges = FALSE,
replicates = NULL, combine = rowMeans, vectorized = TRUE,
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
- `FUN` A function for summarizing non-disjoint ranges (default mean)
RangedRaggedAssay-class

object A RangedRaggedAssay class object
drop.empty.ranges unused argument
replicates reduce: A logical list where each element represents a sample and a vector of repeated experiments for the sample (default NULL)
combine A function for consolidating columns in the matrix representation of the data (default rowMeans)
vectorized logical (default TRUE) whether the combine function is vectorized, optimized for working down the vector pairs

Details

The ... argument allows the user to specify arguments in the subsetByOverlaps function. When calling the reduce method, the additional arguments correspond to those in either the assay method or the reduce method. The reduce 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
• reduce: Use metadata column to produce a matrix which can then be merged across replicates.

See Also

findOverlaps-methods
assay.RangedRaggedAssay,missing-method

Examples

## Load sample MultiAssayExperiment Copy Number data
dataFile <- system.file("extdata", "hnscSample.rds", package = "MultiAssayExperiment")

hnscSample <- readRDS(dataFile)

## assay method for a RangedRaggedAssay
assay(hnscSample[[1L]], mcolname = "Segment_Mean")[1:5, 1:3]

hnscSample[[2]] <- disjoin(hnscSample[[2L]])
matrices <- assay(hnscSample, mcolname = "Segment_Mean")
length(matrices)
class(matrices)

rarrange

Reshape raw data from an object

Description
The `rarrange` function 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 pData columns with the `pDataCols` argument for a `MultiAssayExperiment` object.

Usage

```r
rarrange(object, shape = "long", ...)
```

## S4 method for signature 'ANY'
```r
rarrange(object, shape = "long", ...)
```

## S4 method for signature 'RangedRaggedAssay'
```r
rarrange(object, shape = "long", ...)
```

## S4 method for signature 'ExperimentList'
```r
rarrange(object, shape = "long", ...)
```

## S4 method for signature 'MultiAssayExperiment'
```r
rarrange(object, shape = "long",
         pDataCols = NULL, ...)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>object</td>
<td>Any supported class object</td>
</tr>
<tr>
<td>shape</td>
<td>A single string indicating the shape of the resulting data, options include ‘long’ and ‘wide’ (defaults to the former)</td>
</tr>
<tr>
<td>...</td>
<td>Additional arguments for the <code>RangedRaggedAssay</code> assay method. See below.</td>
</tr>
<tr>
<td>pDataCols</td>
<td>selected pData columns to include in the resulting output</td>
</tr>
</tbody>
</table>

Value
Either a long or wide `DataFrame`

Methods (by class)

- **ANY**: ANY class method, works with classes such as `ExpressionSet` and `SummarizedExperiment` as well as `matrix`

- **RangedRaggedAssay**: `RangedRaggedAssay` class method to return matrix of selected “mcolname” column, defaults to score
sampleMap

- ExperimentList: Rearrange data from the ExperimentList class returns list of DataFrames
- MultiAssayExperiment: Overarching MultiAssayExperiment class method returns a small and skinny DataFrame. The pDataCols arguments allows the user to append pData columns to the long and skinny DataFrame.

See Also

assay, RangedRaggedAssay, missing-method

Examples

eexample("RangedRaggedAssay")
rearrange(myRRA, background = 0)

---

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"),
  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(exprmap, methylmap, rangemap, rnamap)
names(mylist) <- c("Affy", "Methyl450k", "CNVgistic", "RNASeqGene")

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

---

sampleMap <- data.frame()

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
eample("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  

**Description**

subsetByColumn returns a subsetted MultiAssayExperiment object

**Usage**

```r
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 = 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**

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

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

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

Description

Select biological units in a MultiAssayExperiment with subsetBypData

Usage

subsetBypData(x, y)

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

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

Arguments

x A MultiAssayExperiment object

y Either a numeric, character or logical object indicating what pData 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
e.example("MultiAssayExperiment")

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

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

## Subset by logical indicator of pData rows (Jack and Jill)
subsetBypData(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,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
eval.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)
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
```r
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
```r
example(MultiAssayExperiment)
upsetSamples(myMultiAssayExperiment, idclip = function(x) {
  gsub("[a-z]", ",", x)
})
```
Index

.ExperimentList (ExperimentList-class), 6
.RangedRaggedAssay (RangedRaggedAssay-class), 18
[,MultiAssayExperiment,ANY,ANY,ANY-method (MultiAssayExperiment-class), 11
[,MultiAssayExperiment,ANY-method (MultiAssayExperiment-class), 11
[,RangedRaggedAssay,ANY,ANY,ANY-method (RangedRaggedAssay-class), 18
[,RangedRaggedAssay,ANY-method (RangedRaggedAssay-class), 18
[,RangedRaggedAssay,GRanges,ANY,ANY-method (RangedRaggedAssay-class), 18
[,RangedRaggedAssay,GRanges-method (RangedRaggedAssay-class), 18
][,MultiAssayExperiment,ANY,ANY-method (MultiAssayExperiment-class), 11
][<-,MultiAssayExperiment,ANY,ANY-method (MultiAssayExperiment-class), 11
$,MultiAssayExperiment-method (MultiAssayExperiment-class), 11
$<-,MultiAssayExperiment-method (MultiAssayExperiment-class), 11

API, 3, 14
assay,ANY,missing-method (ExperimentList-class), 6
assay,ExperimentList,missing-method (ExperimentList-class), 6
assay,MultiAssayExperiment,missing-method (MultiAssayExperiment-class), 11
assay,RangedRaggedAssay,missing-method, 3, 14, 19
c,MultiAssayExperiment-method (MultiAssayExperiment-class), 11
CharacterList, 7
clusterOn, 4
complete.cases,MultiAssayExperiment-method (MultiAssayExperiment-class), 11
combine, 7
dataFrame, 4, 10, 15, 16, 20
dim,RangedRaggedAssay-method (RangedRaggedAssay-class), 18
dimnames,ExperimentList-method (ExperimentList-class), 6
dimnames,MultiAssayExperiment-method (MultiAssayExperiment-class), 11
dimnames,RangedRaggedAssay-method (RangedRaggedAssay-class), 18
dimnames<-,RangedRaggedAssay,list-method (RangedRaggedAssay-class), 18
disjoin,RangedRaggedAssay-method (RangedRaggedAssay-class), 18
duplicated,MultiAssayExperiment-method (MultiAssayExperiment-class), 11

ExperimentList, 5, 7, 10, 11, 14–16, 20
ExperimentList,ANY-method (ExperimentList-class), 6
ExperimentList,missing-method (ExperimentList-class), 6
ExperimentList-class, 6
experiments, 8
experiments,MultiAssayExperiment-method (MultiAssayExperiment-class), 11
experiments<-, 8
experiments<-,MultiAssayExperiment,ExperimentList-method (MultiAssayExperiment-class), 11
ExpressionSet, 11, 20
findOverlaps, 13
GRanges, 4

28
INDEX

GRangesList, 3, 4, 11, 17, 18
hasAssay, 9
isEmpty, MultiAssayExperiment-method (MultiAssayExperiment-class), 11
length, MultiAssayExperiment-method (MultiAssayExperiment-class), 11
listToMap, 9
LogicalList, 7, 13
mapToList (listToMap), 9
metadata, MultiAssayExperiment-method (MultiAssayExperiment-class), 11
metadata<-, MultiAssayExperiment-method (MultiAssayExperiment-class), 11
MultiAssayExperiment, 10, 14, 16, 20, 23–26
MultiAssayExperiment-class, 10, 11
names, MultiAssayExperiment-method (MultiAssayExperiment-class), 11
overlapsAny, 3, 4
pData, MultiAssayExperiment-method (MultiAssayExperiment-class), 11
pData<-, MultiAssayExperiment, DataFrame-method (MultiAssayExperiment-class), 11
PrepMultiAssay, 16
ReducedRaggedAssay, 3, 4, 11, 17, 18–20
ReducedRaggedAssay-class, 18
rearrange, 20
rearrange, ANY-method (rearrange), 20
rearrange, ExperimentList-method (rearrange), 20
rearrange, MultiAssayExperiment-method (rearrange), 20
rearrange, ReducedRaggedAssay-method (rearrange), 20
reduce, ANY-method (MultiAssayExperiment-class), 11
reduce, ExperimentList-method (ExperimentList-class), 6
reduce, MultiAssayExperiment-method (MultiAssayExperiment-class), 11
reduce, ReducedRaggedAssay-method (ReducedRaggedAssay-class), 18
sampleMap, 11, 15, 16, 21
sampleMap, MultiAssayExperiment-method (MultiAssayExperiment-class), 11
sampleMap<-, 22
sampleMap<-, MultiAssayExperiment, DataFrame-method (MultiAssayExperiment-class), 11
set.seed, 5
show, ExperimentList-method (ExperimentList-class), 6
show, MultiAssayExperiment-method (MultiAssayExperiment-class), 11
show, ReducedRaggedAssay-method (ReducedRaggedAssay-class), 18
SimpleList, 15
subsetByAssay, 23
subsetByAssay, MultiAssayExperiment-method (subsetByAssay), 23
subsetByColumn, 24
subsetByColumn, MultiAssayExperiment, List-method (subsetByColumn), 24
subsetByColumn, MultiAssayExperiment, List-method (subsetByColumn), 24
subsetByOverlaps, 19
subsetBypData, 25
SummarizedExperiment, 11, 20
updateObject, MultiAssayExperiment-method (MultiAssayExperiment-class), 11
upset, 27
upsetSamples, 27