Package ‘MultiAssayExperiment’

May 13, 2024

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

Version 1.30.1

Description Harmonize data management of multiple experimental assays performed on an overlapping set of specimens. It provides a familiar Bioconductor user experience by extending concepts from SummarizedExperiment, supporting an open-ended mix of standard data classes for individual assays, and allowing subsetting by genomic ranges or rownames. Facilities are provided for reshaping data into wide and long formats for adaptability to graphing and downstream analysis.

License Artistic-2.0

URL http://waldronlab.io/MultiAssayExperiment/

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

Depends SummarizedExperiment (>= 1.3.81), R (>= 3.5.0)

Imports Biobase, BiocBaseUtils, BiocGenerics, DelayedArray, GenomicRanges (>= 1.25.93), IRanges, methods, S4Vectors (>= 0.23.19), tidyr, utils

Suggests BiocStyle, HDF5Array (>= 1.19.17), knitr, maftools (>= 2.7.10), R.rsp, RaggedExperiment, reshape2, rmarkdown, survival, survminer, testthat, UpSetR

VignetteBuilder knitr, R.rsp

biocViews Infrastructure, DataRepresentation

Encoding UTF-8

RoxygenNote 7.3.1

Roxygen list(markdown = TRUE)


Collate 'ExperimentList-class.R' 'MultiAssayExperiment-class.R'
'MatchedAssayExperiment-class.R' 'subsetBy-methods.R'
'MultiAssayExperiment-subset.R'
'MultiAssayExperiment-methods.R'
'MultiAssayExperiment-helpers.R' 'MultiAssayExperiment-pkg.R'
'MultiAssayExperimentToMAF.R' 'assay-methods.R' 'data.R'
'hasAssay.R' 'listToMap.R' 'mapToList.R' 'prepMultiAssay.R'
'reexports.R' 'saveHDF5MultiAssayExperiment.R' 'upsetSamples.R'
'utilities.R'

Date  2024-04-29

git_url  https://git.bioconductor.org/packages/MultiAssayExperiment

git_branch  RELEASE_3_19

git_last_commit  254c7e9

git_last_commit_date  2024-05-01

Repository  Bioconductor 3.19

Date/Publication  2024-05-12

Author  Marcel Ramos [aut, cre] (<https://orcid.org/0000-0002-3242-0582>),
        Martin Morgan [aut, ctb],
        Lori Shepherd [ctb],
        Hervé Pagès [ctb],
        Vincent J Carey [aut, ctb],
        Levi Waldron [aut],
        MultiAssay SIG [ctb]

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

Contents

MultiAssayExperiment-package ........................................... 3
ExperimentList ................................................................. 4
ExperimentList-class ......................................................... 5
hasAssay ........................................................................... 7
listToMap ........................................................................... 8
MatchedAssayExperiment-class .............................................. 9
miniACC ............................................................................. 10
MultiAssayExperiment .......................................................... 11
MultiAssayExperiment-class .................................................. 13
MultiAssayExperiment-helpers .............................................. 17
MultiAssayExperiment-methods .............................................. 23
MultiAssayExperimentToMAF ............................................... 26
prepMultiAssay ................................................................. 26
reexports ........................................................................... 28
saveHDF5MultiAssayExperiment ............................................ 28
subsetBy ............................................................................. 30
upsetSamples ................................................................... 32

Index ................................................................................. 35
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

Author(s)

Maintainer: Marcel Ramos <marcel.ramos@roswellpark.org> (ORCID)
Authors:

- Martin Morgan [contributor]
- Vincent J Carey [contributor]
- Levi Waldron <lwaldron.research@gmail.com>

Other contributors:

- Lori Shepherd [contributor]
- Hervé Pagès [contributor]
- MultiAssay SIG <biocmultiassay@googlegroups.com> [contributor]

See Also

Useful links:

- [http://waldronlab.io/MultiAssayExperiment/](http://waldronlab.io/MultiAssayExperiment/)
- Report bugs at [https://github.com/waldronlab/MultiAssayExperiment/issues](https://github.com/waldronlab/MultiAssayExperiment/issues)
ExperimentList

Represent multiple experiments as a List-derivative ExperimentList

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, [ , dimnames

Usage

ExperimentList(...)

Arguments

... A named list class object

Value

A ExperimentList class object of experiment data

Examples

## Create an empty ExperimentList instance
ExperimentList()

## Create array matrix and AnnotatedDataFrame to create an ExpressionSet class
arraydat <- matrix(data = seq(101, length.out = 20), ncol = 4,
                   dimnames = list(
                     c("ENST00000294241", "ENST00000355076",
                     "ENST00000383706", "ENST00000234812", "ENST00000383323"),
                     c("array1", "array2", "array3", "array4")
                   ))

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

## SummarizedExperiment constructor
exprdat <- SummarizedExperiment::SummarizedExperiment(arraydat,
colData = colDat)

## Create a sample methylation dataset
methyldat <- matrix(data = seq(1, length.out = 25), ncol = 5,
                      dimnames = list(
                        c("ENST00000355076", "ENST00000383706",
                         "ENST00000383323", "ENST00000234812", "ENST00000294241"),
                        c("methyl1", "methyl2", "methyl3",
                          "methyl4", "methyl5")
                      ))
## Create a sample RNASeqGene dataset
```
rnadat <- matrix(
  data = sample(c(46851, 5, 19, 13, 2197, 507,
                 84318, 126, 17, 21, 23979, 614), size = 20, replace = TRUE),
  ncol = 4,
  dimnames = list(  
    c("XIST", "RPS4Y1", "KDM5D", "ENST00000383323", "ENST00000234812"),
    c("samparray1", "samparray2", "samparray3", "samparray4")  
  ))
```

## Create a mock RangedSummarizedExperiment from a data.frame
```
rangedat <- data.frame(chr="chr2", start = 11:15, end = 12:16,
  strand = c("+", "+", "+", "+", "+"),
  samp0 = c(0,0,1,1,1), samp1 = c(1,0,1,0,1), samp2 = c(0,1,0,1,0),
  row.names = c(paste0("ENST", "00000", 135411:135414), "ENST00000383323"))
rangeSE <- SummarizedExperiment::makeSummarizedExperimentFromDataFrame(rangedat)
```

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

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

### 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.

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

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

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

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

## S4 method for signature 'ExperimentList'
```
colnames(x, do.NULL = TRUE, prefix = "col")

## S4 method for signature 'ExperimentList'
rownames(x, do.NULL = TRUE, prefix = "row")

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

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

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

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

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

## S4 method for signature 'ExperimentList,character'
assay(x, i, withDimnames = TRUE, ...)

### Arguments

- **object, x** An `ExperimentList` object
- **do.NULL, prefix** See ?base::rownames for a description of these arguments.
- **replicates** mergeReplicates: A list or `LogicalList` where each element represents a sample and a vector of repeated measurements for the sample
- **simplify** A function for merging columns where duplicates are indicated by replicates
- **...** Additional arguments. See details for more information.
- **i** A scalar character or integer index
- **withDimnames** logical (default TRUE) whether to return dimension names

### Value

An `ExperimentList` class object

### Methods (by generic)

- `show(ExperimentList)`: Show method for `ExperimentList` class
- `isEmpty(ExperimentList)`: check for zero length across all experiments
- `dimnames(ExperimentList)`: Get the dimension names for an `ExperimentList` using `CharacterList`
- `colnames(ExperimentList)`: Get the column names for an `ExperimentList` as a `CharacterList` slightly more efficiently
- `rownames(ExperimentList)`: Get the row names for an `ExperimentList` as a `CharacterList` slightly more efficiently
hasAssay

- mergeReplicates(ExperimentList): Apply the mergeReplicates method on the ExperimentList elements
- assay(x = ANY, i = missing): Obtain the specified assay with a numeric or character reference
- assays(ExperimentList): Get the assay data from each element in the ExperimentList

coercion

Convert a list or S4 List to an ExperimentList using the as() function.

In the following example, x is either a list or list:

```r
as(x, "ExperimentList")
```

Examples

```r
ExperimentList()
```

### hasAssay

Checking assay method for any class

**Description**

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

**Usage**

```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 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, fill = TRUE)

mapToList(dfmap, assayCol = "assay")

Arguments

listmap A named list object containing DataFrames with "primary" and "colname" columns
fill logical(1) Whether to fill the map with an empty DataFrame when empty elements are present in the input list
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(): The inverse of the listToMap operation

Examples

example("MultiAssayExperiment")

## Create a sampleMap from a list using the listToMap function
sampMap <- listToMap(maplist)

## The inverse operation is also available
maplist <- mapToList(sampMap)
MatchedAssayExperiment-class

Description

This class supports the use of matched samples where an equal number of observations per biological unit are present in all assays.

Usage

MatchedAssayExperiment(...)

Arguments

... Either a single MultiAssayExperiment or the components to create a valid MultiAssayExperiment

Value

A MatchedAssayExperiment object

Functions

• MatchedAssayExperiment(): Construct a MatchedAssayExperiment class from MultiAssayExperiment

See Also

MultiAssayExperiment

Examples

data("miniACC")
acc <- as(miniACC, "MatchedAssayExperiment")
acc
Description

A MultiAssayExperiment object providing a reduced version of the TCGA ACC dataset for all 92 patients. RNA-seq, copy number, and somatic mutations are included only for genes whose proteins are included in the reverse-phase protein array. The MicroRNA-seq dataset is also included, with infrequently expressed microRNA removed. Clinical, pathological, and subtype information are provided by colData(miniACC), and some additional details are provided by metadata(miniACC).

Usage

data("miniACC")

Format

A MultiAssayExperiment with 5 experiments, providing:

**RNASeq2GeneNorm** RNA-seq count data: an ExpressionSet with 198 rows and 79 columns

**gistict** Recurrent copy number lesions identified by GISTIC2: a SummarizedExperiment with 198 rows and 90 columns

**RPPAArray** Reverse Phase Protein Array: an ExpressionSet with 33 rows and 46 columns.

Rows are indexed by genes, but protein annotations are available from featureData(miniACC["RPPAArray"]). The source of these annotations is noted in abstract(miniACC["RPPAArray"]).

**Mutations** Somatic mutations: a matrix with 223 rows and 90 columns. 1 for any kind of non-silent mutation, zero for silent (synonymous) or no mutation.

**miRNASeqGene** microRNA sequencing: an ExpressionSet with 471 rows and 80 columns.

Rows not having at least 5 counts in at least 5 samples were removed.

Author(s)

Levi Waldron <lwaldron.research@gmail.com>

Source

https://github.com/waldronlab/multiassayexperiment-tcga

References

MultiAssayExperiment

Examples

data("miniACC")
metadata(miniACC)
colnames(colData(miniACC))
table(miniACC$vital_status)
longFormat(
  miniACC["MAPK3", , ],
  colDataCols = c("vital_status", "days_to_death")
)

wideFormat(
  miniACC["MAPK3", , ],
  colDataCols = c("vital_status", "days_to_death")
)

---

MultiAssayExperiment  Construct an integrative representation of multi-omic data with
MultiAssayExperiment

Description

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

Usage

MultiAssayExperiment(
  experiments = ExperimentList(),
  colData = S4Vectors::DataFrame(),
  sampleMap = S4Vectors::DataFrame(assay = factor(), primary = character(), colname = character()),
  metadata = list(),
  drops = list()
)

Arguments

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

A MultiAssayExperiment object that can store experiment and phenotype data

colData

The colData input can be either DataFrame or data.frame with subsequent coercion to DataFrame. The rownames in the colData must match the colnames in the experiments if no sampleMap is provided.

experiments

The experiments input can be of class SimpleList or list. This input becomes the ExperimentList. Each element of the input list or list must be named, rectangular with two dimensions, and have dimnames.

sampleMap

The sampleMap can either be input as DataFrame or data.frame with eventual coercion to DataFrame. The sampleMap relates biological units and biological measurements within each assay. Each row in the sampleMap is a single such link. The standard column names of the sampleMap are "assay", "primary", and "colname". Note that the "assay" column is a factor corresponding to the names of each experiment in the ExperimentList. In the case where these names do not match between the sampleMap and the experiments, the documented experiments in the sampleMap take precedence and experiments are dropped by the harmonization procedure. The constructor function will generate a sampleMap in the case where it is not provided and this method may be a 'safer' alternative for creating the MultiAssayExperiment (so long as the rownames are identical in the colData, if provided). An empty sampleMap may produce empty experiments if the levels of the "assay" factor in the sampleMap do not match the names in the ExperimentList.

See Also

MultiAssayExperiment

Examples

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

## 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)

rnamap <- data.frame(
```

MultiAssayExperiment-class

## 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`, `RaggedExperiment` (inherits from `GRangesList`), and `RangedVcfStack`. Create new `MultiAssayExperiment` instances with the homonymous constructor, minimally with the argument `ExperimentList`, potentially also with the arguments `colData` (see section below) and `sampleMap`.

## Usage

### S4 method for signature 'MultiAssayExperiment'

- `show(object)`
- `length(x)`
- `names(x)`

```r
MultiAssayExperiment-class

MultiAssayExperiment - An integrative multi-assay class for experiment data

primary = c("Jack", "Jill", "Bob", "Barbara"),
colname = c("samparray1", "samparray2", "samparray3", "samparray4"),
stringsAsFactors = FALSE)

gistmap <- data.frame(
    primary = c("Jack", "Bob", "Jill"),
colname = c("samp0", "samp1", "samp2"),
stringsAsFactors = FALSE)

## Combine as a named list and convert to a DataFrame
maplist <- list(Affy = exprmap, Methyl450k = methylmap,
    RNASeqGene = rnamap, GISTIC = gistmap)

## Create a sampleMap
sampMap <- listToMap(maplist)

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

## Create a MultiAssayExperiment instance
mae <- MultiAssayExperiment(experiments = ExpList, colData = colDat,
    sampleMap = sampMap)
```
updateObject(object, ..., verbose = FALSE)

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

dimnames(x)

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

c(x, ..., sampleMap = NULL, mapFrom = NULL)

## S4 method for signature 'MultiAssayExperiment'
exportClass(
  object,
  dir = tempdir(),
  fmt,
  ext,
  match = FALSE,
  verbose = TRUE,
  ...
)

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

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

## S4 method for signature 'MultiAssayExperiment,numeric'
assay(x, i, withDimnames = TRUE, ...)

## S4 method for signature 'MultiAssayExperiment,character'
assay(x, i, withDimnames = TRUE, ...)

Arguments

- **object, x** A MultiAssayExperiment object
- **...** Additional arguments for supporting functions. See details.
- **verbose** logical(1) Whether to print additional information (default TRUE)
- **sampleMap** c method: a sampleMap list or DataFrame to guide merge
- **mapFrom** Either a logical, character, or integer vector indicating the experiment(s) that have an identical colname order as the experiment input(s). If using a character input, the name must match exactly.
- **dir** character(1) A directory for saving exported data (default: tempdir())
- **fmt** character(1) or function() Either a format character atomic as supported by write.table either (‘csv’, or ‘tsv’) or a function whose first two arguments are ‘object to save’ and ‘file location’
- **ext** character(1) A file extension supported by the format argument
- **match** logical(1) Whether to coerce the current object to a 'MatchedAssayExperiment' object (default: FALSE)
withDimnames logical (default TRUE) whether to return dimension names included in the object

An integer or character scalar indicating the assay to return

Details

The dots (...) argument allows the user to specify additional arguments in several instances.

- subsetting []: additional arguments sent to `findOverlaps`.
- mergeReplicates: used to specify arguments for the `simplify` functional argument
- assay: may contain withDimnames, which is forwarded to assays
- combining c: compatible MultiAssayExperiment classes passed on to the `ExperimentList` constructor, can be a list, `List`, or a series of named arguments. See the examples below.

Value

A MultiAssayExperiment object

Methods (by generic)

- `show(MultiAssayExperiment)`: Show method for a MultiAssayExperiment
- `length(MultiAssayExperiment)`: Get the length of ExperimentList
- `names(MultiAssayExperiment)`: Get the names of the ExperimentList
- `updateObject(MultiAssayExperiment)`: Update old serialized MultiAssayExperiment objects to new API
- `dimnames(MultiAssayExperiment)`: Get the dimension names for a MultiAssayExperiment object
- `c(MultiAssayExperiment)`: Add a supported data class to the ExperimentList
- `exportClass(MultiAssayExperiment)`: Export data from class to a series of text files
- `assays(MultiAssayExperiment)`: Obtain a `SimpleList` of assay data for all available experiments in the object
- `assay(x = MultiAssayExperiment, i = missing)`: Convenience function for extracting the assay of the first element (default) in the ExperimentList. A numeric or character index can also be provided

Slots

- `ExperimentList` A `ExperimentList` class object for each assay dataset
- `colData` A `DataFrame` of all clinical/specimen data available across experiments
- `sampleMap` A `DataFrame` of translatable identifiers of samples and participants
- `metadata` Additional data describing the MultiAssayExperiment object
- `drops` A metadata list of dropped information
colData

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

ExperimentList

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

sampleMap

The sampleMap contains a DataFrame of translatable identifiers of samples and participants or biological units. The standard column names of the sampleMap are "assay", "primary", and "col-name". Note that the "assay" column is a factor corresponding to the names of each experiment in the ExperimentList. In the case where these names do not match between the sampleMap and the experiments, the documented experiments in the sampleMap take precedence and experiments are dropped by the harmonization procedure. The constructor function will generate a sampleMap in the case where it is not provided and this method may be a safer alternative for creating the MultiAssayExperiment (so long as the rownames are identical in the colData, if provided). An empty sampleMap may produce empty experiments if the levels of the "assay" factor in the sampleMap do not match the names in the ExperimentList.

corercion

Convert a list or S4 List to a MultiAssayExperiment object using the methods::as function. In the following example, x is either a list or List:

as(x, "MultiAssayExperiment")

Convert a MultiAssayExperiment to MAF class object using the methods::as function. In the following example, x is a MultiAssayExperiment:

MultiAssayExperimentToMAF(x)

See Also

MultiAssayExperiment-methods for slot modifying methods, MultiAssayExperiment API

Examples

eexample("MultiAssayExperiment")

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

# Columns (j) Rows in colData
mae[, rownames(colData(mae))[3:2], ]
# Assays (k)
mae[, , "Affy"]

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

example("MultiAssayExperiment")

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

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

test2 <- mae[[3L]]
c(mae, newExp = test2, mapFrom = 3L)

## Add experiment using experiment name in mapFrom
c(mae, RNASeqGeneV2 = test2, mapFrom = "RNASeqGene")

---

MultiAssayExperiment-helpers

A group of helper functions for manipulating and cleaning a MultiAssayExperiment

Description

A set of helper functions were created to help clean and manipulate a MultiAssayExperiment object. intersectRows also works for ExperimentList objects.

- complete.cases: Returns a logical vector corresponding to `colData` rows that have data across all experiments
- isEmpty: Returns a logical TRUE value for zero length MultiAssayExperiment objects
- intersectRows: Takes all common rows across experiments, excludes experiments with empty rownames
- intersectColumns: A wrapper for complete.cases to return a MultiAssayExperiment with only those biological units that have measurements across all experiments
- replicated: Identifies, via logical vectors, colnames that originate from a single biological unit within each assay
- replicates: Provides the replicate colnames found with the replicated function by their name, empty list if none
• anyReplicated: Whether the assay has replicate measurements
• showReplicated: Displays the actual columns that are replicated per assay and biological unit, i.e., primary value (colData rowname) in the sampleMap
• mergeReplicates: A function that combines replicated / repeated measurements across all experiments and is guided by the replicated return value
• longFormat: A MultiAssayExperiment method that returns a small and skinny DataFrame. The colDataCols arguments allows the user to append colData columns to the data.
• wideFormat: A function to reshape the data in a MultiAssayExperiment to a "wide" format DataFrame. Each row in the DataFrame represents an observation (corresponding to an entry in the colData). If replicates are present, their data will be appended at the end of the corresponding row and will generate additional NA data. It is recommended to remove or consolidate technical replicates with mergeReplicates. Optional colDataCols can be added when the original object is a MultiAssayExperiment.
• hasRowRanges: A function that identifies ExperimentList elements that have a rowRanges method
• getWithColData: A convenience function for extracting an assay and associated colData
• renamePrimary: A convenience function to rename the primary biological units as represented in the rownames(colData)
• renameColname: A convenience function to rename the colnames of a particular assay

Usage

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

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

intersectRows(x)

intersectColumns(x)

replicated(x)

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

anyReplicated(x)

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

showReplicated(x)

## S4 method for signature 'MultiAssayExperiment'
showReplicated(x)
```
replicates(x, ...)

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

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

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

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

longFormat(object, colDataCols = NULL, i = 1L)

wideFormat(
  object,
  colDataCols = NULL,
  check.names = TRUE,
  collapse = "_",
  i = 1L
)

hasRowRanges(x)

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

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

getWithColData(x, i, mode = c("append", "replace"), verbose = FALSE)

renamePrimary(x, value)

renameColname(x, i, value)

splitAssays(x, hitList)

## S4 method for signature 'MultiAssayExperiment'
splitAssays(x, hitList)
makeHitList(x, patternList)

Arguments

... Additional arguments. See details for more information.

x A MultiAssayExperiment or ExperimentList

replicates A list of LogicalLists indicating multiple / duplicate entries for each biological unit per assay, see replicated (default replicated(x)).

simplify A function for merging repeat measurements in experiments as indicated by the replicated method for MultiAssayExperiment

object Any supported class object

colDataCols A character, logical, or numeric index for colData columns to be included

i longFormat: The i-th assay in SummarizedExperiment-like objects. A vector input is supported in the case that the SummarizedExperiment object(s) has more than one assay (default 1L), renameColname: Either a numeric or character index indicating the assay whose colnames are to be renamed

check.names (logical default TRUE) Column names of the output DataFrame will be checked for syntactic validity and made unique, if necessary

collapse (character default ")") A single string delimiter for output column names. In wideFormat, experiments and rownames (and when replicate samples are present, colnames) are seperated by this delimiter

mode String indicating how MultiAssayExperiment column-level metadata should be added to the SummarizedExperiment colData.

verbose logical(1) Whether to suppressMessages on subsetting operations in getWithColData (default FALSE)

value renamePrimary: A character vector of the same length as the existing rownames(colData) to use for replacement, renameColname: A CharacterList or list with matching lengths to replace colnames(x)

hitList a named list or List of logical vectors that indicate groupings in the assays

patternList a named list or List of atomic character vectors that are the input to grepl for identifying groupings in the assays

Details

The replicated function finds replicate measurements in each assay and returns a list of LogicalLists. Each element in a single LogicalList corresponds to a biological or primary unit as in the sampleMap. Below is a small graphic for one particular biological unit in one assay, where the logical vector corresponds to the number of measurements/samples in the assay:

```
> replicated(MultiAssayExperiment)
(list str) '--- $ AssayName
(LogicalList str) '--- [[ "Biological Unit" ]]
Replicated if sum(...) > 1 '--- TRUE TRUE FALSE FALSE
anyReplicated determines if any of the assays have at least one replicate. Note. These methods are not available for the ExperimentList class due to a missing sampleMap structure (by design). showReplicated returns a list of CharacterLists where each element corresponds to the the biological or primary units that are replicated in that assay element. The values in the inner list are the colnames in the assay that are technical replicates.

The replicates function (noun) returns the colnames from the sampleMap that were identified as replicates. It returns a list of CharacterLists for each assay present in the MultiAssayExperiment and an inner entry for each biological unit that has replicate observations in that assay.

The mergeReplicates function is a house-keeping method for a MultiAssayExperiment where only complete cases are returned. This by-assay operation averages replicate measurements (by default) and columns are aligned by the row order in colData. Users can provide their own function for merging replicates with the simplify functional argument. Additional inputs ... are sent to the 'simplify' function.

The mergeReplicates "ANY" method consolidates duplicate measurements for rectangular data structures, returns object of the same class (endomorphic). The ellipsis or ... argument allows the user to provide additional arguments to the simplify functional argument.

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

The hasRowRanges method identifies assays that support a rowRanges method and return a GRanges object.

Value

See the itemized list in the description section for details.

mergeReplicates

The mergeReplicates function makes use of the output from replicated which will point out the duplicate measurements by biological unit in the MultiAssayExperiment. This function will return a MultiAssayExperiment with merged replicates. Additional arguments can be provided to the simplify argument via the ellipsis (...). For example, when replicates "TCGA-B" and "TCGA-A" are found in the assay, the name of the first appearing replicate is taken (i.e., "B"). Note that a typical use case of merging replicates occurs when there are multiple measurements on the same sample (within the same assay) and can therefore be averaged.

longFormat

The 'longFormat' method takes data from the ExperimentList in a MultiAssayExperiment and returns a uniform DataFrame. The resulting DataFrame has columns indicating primary, rowname, colname and value. This method can optionally include columns of the MultiAssayExperiment colData named by colDataCols character vector argument. (MultiAssayExperiment method only). The i argument allows the user to specify the assay value for the SummarizedExperiment assay function's i argument.

wideFormat

The wideFormat function returns standardized wide DataFrame where each row represents a biological unit as in the colData. Depending on the data and setup, biological units can be patients,
tumors, specimens, etc. Metadata columns are generated based on the names produced in the wide format DataFrame. These can be accessed via the `mcols()` function. See the wideFormat section for description of the `colDataCols` and `i` arguments.

hasRowRanges

The `hasRowRanges` method identifies assays with associated ranged row data by directly testing the method on the object. The result from the test must be a `GRanges` class object to satisfy the test.

gETCHColData

The `getWithColData` function allows the user to conveniently extract a particular assay as indicated by the `i` index argument. It will also attempt to provide the `colData` along with the extracted object using the `colData<-.` replacement method when possible. Typically, this method is available for `SummarizedExperiment` and `RaggedExperiment` classes.

The setting of `mode` determines how the `colData` is added. If `mode="append"`, the `MultiAssayExperiment` metadata is appended onto that of the `SummarizedExperiment`. If any fields are duplicated by name, the values in the `SummarizedExperiment` are retained, with a warning emitted if the values are different. For `mode="replace"`, the `MultiAssayExperiment` metadata replaces that of the `SummarizedExperiment`, while for `mode="none"`, no replacement or appending is performed.

rename*

The `renamePrimary` function allows the user to conveniently change the actual names of the primary biological units as seen in `rownames(colData)`. `renameColName` allows the user to change the names of a particular assay based on index `i`. `i` can either be a single numeric or character value. See `colnames<-.` method for renaming multiple colnames in a `MultiAssayExperiment`.

splitAssays

The `splitAssays` method separates columns in each of the assays based on the `hitList` input. The `hitList` can be generated using the `makeHitList` helper function. To use the `makeHitList` helper, the user should input a list of patterns that will match on the column names of each assay. These matches should be mutually exclusive as to avoid repetition of columns across assays. See the examples section.

Examples

```r
example(MultiAssayExperiment)
complete.cases(mae)
isEmpty(MultiAssayExperiment())
```

## renaming biological units (primary)

```r
mae2 <- renamePrimary(mae, paste0("pt", 1:4))
colData(mae2)
sampleMap(mae2)
```
## renaming observational units (colname)

```r
mae2 <- renameColname(mae, i = "Affy", paste0("ARRAY", 1:4))
colnames(mae2)
sampleMap(mae2)
```

```r
patts <- list(
  normals = "TCGA-[A-Z0-9]{2}-[A-Z0-9]{4}-11",
  tumors = "TCGA-[A-Z0-9]{2}-[A-Z0-9]{4}-01"
)
data("miniACC")

```r
hits <- makeHitList(miniACC, patts)
```## only turmors present

```r
splitAssays(miniACC, hits)
```
## S4 replacement method for signature 'MultiAssayExperiment,ANY'
sampleMap(object) <- value

drops(x, ...) <- value

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

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

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

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

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

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

## S4 replacement method for signature 'MultiAssayExperiment'
names(x) <- value

## S4 replacement method for signature 'MultiAssayExperiment,ExperimentList'
colnames(x) <- value

## S4 replacement method for signature 'MultiAssayExperiment,list'
colnames(x) <- value

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

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

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

### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Argument not in use</td>
</tr>
<tr>
<td>object, x</td>
<td>A MultiAssayExperiment object</td>
</tr>
<tr>
<td>value</td>
<td>See details.</td>
</tr>
<tr>
<td>name</td>
<td>A column in colData</td>
</tr>
</tbody>
</table>
**Value**

Accessors: Either a sampleMap, ExperimentList, or DataFrame object

Setters: A MultiAssayExperiment object

**Accessors**

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

- colData: Access the colData slot
- sampleMap: Access the sampleMap slot
- experiments: Access the ExperimentList slot
- [[: Access the ExperimentList slot
- $: Access a column in colData
- drops: Get a vector of dropped ExperimentList names

**Setters**

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

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

**Examples**

```r
## Load example MultiAssayExperiment
eexample(MultiAssayExperiment)

## Access the sampleMap
sampleMap(mae)

## Replacement method for a MultiAssayExperiment sampleMap
sampleMap(mae) <- S4Vectors::DataFrame()

## Access the ExperimentList
experiments(mae)

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

## Access the metadata
metadata(mae)
```
## Replace metadata with a list

```r
metadata(mae) <- list(runDate =
    format(Sys.time(), "%B %d, %Y"))
```

## Access the colData

```r
colData(mae)
```

## Access a column in colData

```r
mae$age
```

## Replace a column in colData

```r
mae$age <- mae$age + 1
```

---

**MultiAssayExperimentToMAF**

*Convert MultiAssayExperiment to MAF class*

### Description

Take a `MultiAssayExperiment` object with specific mutation assays and convert these into a `maftools` representation. The names provided via `synAssay` and `nonSynAssay` must match exactly those assays in the `MultiAssayExperiment`.

### Usage

```r
MultiAssayExperimentToMAF(x, synAssay = "maf_syn", nonSynAssay = "maf_nonSyn")
```

### Arguments

- `x` : A `MultiAssayExperiment` object
- `synAssay` : character(1) The name of the `ExperimentList` element in the `MultiAssayExperiment` that identifies synonymous variant classifications.
- `nonSynAssay` : character(1) The name of the `ExperimentList` element in the `MultiAssayExperiment` that identifies non-synonymous variant classifications.

---

**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 `colData`, or `sampleMap`. 

Usage

prepMultiAssay(ExperimentList, colData, sampleMap, ...)

Arguments

- **ExperimentList**: A list of all combined experiments
- **colData**: A DataFrame of the phenotype data for all participants
- **sampleMap**: A DataFrame of sample identifiers, assay samples, and assay names
- **...**: Optional arguments for the MultiAssayExperiment constructor function such as metadata and drops.

Value

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

Checks

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

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

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

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

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

Examples

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

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

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

## Alternatively, use the do.call function
do.call(MultiAssayExperiment, preparedMAE)
```
Objects exported from other packages

Description

These objects are imported from other packages. Click on the function name to see their documentation.

- S4Vectors: DataFrame

Examples

DataFrame()

saveHDF5MultiAssayExperiment

Save a MultiAssayExperiment class object to HDF5 and Rds files

Description

This function takes a MultiAssayExperiment object and uses the assays functionality to obtain data matrices out of the experiments. These are then saved into the .h5 file format. This function relies heavily on the HDF5Array package whose installation is required before use. saveHDF5MultiAssayExperiment preserves the classes contained in the ExperimentList with the exception of matrix which is converted to HDF5Matrix. Internal SummarizedExperiment assays are converted to HDF5-backed assays as in HDF5Array::saveHDF5SummarizedExperiment. SummarizedExperiment objects with multiple i-th assays will have the first assay take precedence and others assays will be dropped with a warning. If the first assay in a SummarizedExperiment contains an array, the array is preserved in the process of saving and loading the HDF5-backed MultiAssayExperiment.

Usage

saveHDF5MultiAssayExperiment(
  x,
  dir = "h5_mae",
  prefix = NULL,
  replace = FALSE,
  chunkdim = NULL,
  level = NULL,
  as.sparse = NA,
  verbose = NA
)

loadHDF5MultiAssayExperiment(dir = "h5_mae", prefix = NULL)
saveHDF5MultiAssayExperiment

Arguments

- **x** A `MultiAssayExperiment` object or derivative
- **dir** The path (as a single string) to the directory where to save the HDF5-based `MultiAssayExperiment` object or to load it from.
  - When saving, the directory will be created if it doesn’t already exist. If the directory already exists and no prefix is specified and replace is set to TRUE, then it’s replaced with an empty directory.
- **prefix** An optional prefix to add to the names of the files created inside `dir`. This allows saving more than one object in the same directory. When the prefix is NULL, the name of the `x` input `MultiAssayExperiment` is used. To avoid the default setting use an empty character string i.e., "". An underscore (_) is appended to the prefix when provided; therefore, typical inputs should be words, e.g., "test".
- **replace** When no prefix is specified, should a pre-existing directory be replaced with a new empty one? The content of the pre-existing directory will be lost!
- **chunkdim, level** The dimensions of the chunks and the compression level to use for writing the assay data to disk.
  - Passed to the internal calls to `writeHDF5Array`. See ?`writeHDF5Array` for more information.
- **as.sparse** Whether the assay data should be flagged as sparse or not. If set to NA (the default), then the specific `as.sparse` value to use for each assay is determined by calling `is_sparse()` on them.
  - Passed to the internal calls to `writeHDF5Array`. See ?`writeHDF5Array` for more information and an IMPORTANT NOTE.
- **verbose** Set to TRUE to make the function display progress.
  - In the case of `saveHDF5MultiAssayExperiment()`, `verbose` is set to NA by default, in which case verbosity is controlled by `DelayedArray.verbose.block.processing` option. Setting `verbose` to TRUE or FALSE overrides the option.

Examples

data("miniACC")

testDir <- file.path(tempdir(), "test_mae")

saveHDF5MultiAssayExperiment(
  miniACC, dir = testDir, verbose = TRUE, replace = TRUE
)

## inspect the files in the dir
list.files(testDir)

loadHDF5MultiAssayExperiment(
  dir = testDir
)

## remove example files
unlink(testDir, recursive = TRUE)
**Description**

A set of functions for extracting and dividing a MultiAssayExperiment

**Usage**

```r
subsetByRow(x, y, ...)
subsetByColData(x, y)
subsetByColumn(x, y)
subsetByAssay(x, y)
```

## S4 method for signature 'ExperimentList,ANY'

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

## S4 method for signature 'ExperimentList,list'

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

## S4 method for signature 'ExperimentList,List'

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

## S4 method for signature 'ExperimentList,logical'

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

## S4 method for signature 'ExperimentList,logical'

```r
subsetByColumn(x, y)
```

## S4 method for signature 'ExperimentList,List'

```r
subsetByColumn(x, y)
```

## S4 method for signature 'ExperimentList,List'

```r
subsetByColumn(x, y)
```

## S4 method for signature 'ExperimentList,List'

```r
subsetByColumn(x, y)
```

## S4 method for signature 'ExperimentList,logical'

```r
subsetByColumn(x, y)
```

## S4 method for signature 'ExperimentList'

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

## S4 method for signature 'MultiAssayExperiment,ANY'

```r
subsetByColData(x, y)
```

## S4 method for signature 'MultiAssayExperiment,character'

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

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

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

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

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

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

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

### Arguments

- **x**: A `MultiAssayExperiment` or `ExperimentList`
- **y**: Any argument used for subsetting, can be a character, logical, integer, list or List vector
- **...**: Additional arguments passed on to lower level functions.
- **i**: Either a character, integer, logical or GRanges object for subsetting by rows
- **j**: Either a character, logical, or numeric vector for subsetting by colData rows. See details for more information.
- **k**: Either a character, logical, or numeric vector for subsetting by assays
- **drop**: logical (default FALSE) whether to drop all empty assay elements in the ExperimentList
- **value**: An assay compatible with the MultiAssayExperiment API

### Details

Subsetting a MultiAssayExperiment by the `j` index can yield a call to either `subsetByColData` or `subsetByColumn`. For vector inputs, the subset will be applied to the colData rows. For List-type inputs, the List will be applied to each of the elements in the ExperimentList. The order of the subsetting elements in the List must match that of the ExperimentList in the MultiAssayExperiment.

- `subsetByColData`: Select biological units by vector input types
- `subsetByColumn`: Select observations by assay or for each assay
- `subsetByRow`: Select rows by assay or for each assay
- `subsetByAssay`: Select experiments
Value

subsetBy* operations are endomorphic and return either MultiAssayExperiment or ExperimentList depending on the input.

Examples

```r
## Load the example MultiAssayExperiment
eexample("MultiAssayExperiment")

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

## Using numeric indices
subsetByAssay(mae, 1:2)

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

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

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

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

subsetWith <- S4Vectors::mendoapply("\[", colnames(mae),
    MoreArgs = list(1:2))
subsetByColumn(mae, subsetWith)

## Use a GRanges object to subset rows where ranged data present
egr <- GenomicRanges::GRanges(seqnames = "chr2",
    IRanges::IRanges(start = 11, end = 13), strand = "-")
subsetByRow(mae, egr)

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

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

upsetSamples

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

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

Usage

upsetSamples(
  MultiAssayExperiment,
  nsets = NULL,
  sets = names(MultiAssayExperiment),
  nintersects = NA_integer_,
  order.by = "freq",
  check.names = FALSE,
  ...
)

Arguments

MultiAssayExperiment  A MultiAssayExperiment object
nsets                 numeric(1) The number of sets to analyze. If specified, sets will be ignored.
sets                  character() A character vector of names in MultiAssayExperiment to use. If specified, nsets will be ignored.
nintersects          numeric(1) The number of intersections to plot. By default, 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.
check.names          logical(1) Whether to munge names as in the data.frame() constructor (default FALSE).
...                   parameters passed to UpSetR::upset

Value

Produces a visualization of set intersections using the UpSet matrix design

Note

This function is intended to provide convenient visualization of assay availability configurations in MultiAssayExperiment instances. The UpSetR::upset function requires data.frame input and has many parameters to tune appearance of the result. Assay name handling is important for interpretability.

Author(s)

Vincent J Carey
Examples

data(miniACC)
upsetSamples(miniACC)
upsetSamples(miniACC, nsets = 3, nintersects = 3)
Index

* data

miniACC, 10
[,MultiAssayExperiment,ANY,ANY,ANY-method
  (subsetBy), 30
[,MultiAssayExperiment,ANY-method
  (subsetBy), 30
[<-,MultiAssayExperiment,ANY,ANY,ANY-method
  (subsetBy), 30
][[,MultiAssayExperiment,ANY,ANY-method
  (subsetBy), 30
][<-,MultiAssayExperiment,ANY,ANY-method
  (subsetBy), 30
$,MultiAssayExperiment-method
  (MultiAssayExperiment-methods), 23
$<-,MultiAssayExperiment-method
  (MultiAssayExperiment-methods), 23

anyReplicated
  (MultiAssayExperiment-methods), 17
anyReplicated,MultiAssayExperiment-method
  (MultiAssayExperiment-methods), 17
assay,ANY,missing-method
  (ExperimentList-class), 5
assay,ExperimentList,character-method
  (ExperimentList-class), 5
assay,ExperimentList,missing-method
  (ExperimentList-class), 5
assay,ExperimentList,numeric-method
  (ExperimentList-class), 5
assay,MultiAssayExperiment,character-method
  (MultiAssayExperiment-class), 13
assay,MultiAssayExperiment,missing-method
  (MultiAssayExperiment-class), 13
assay,MultiAssayExperiment,numeric-method
  (MultiAssayExperiment-class), 13

coerce,MultiAssayExperiment-method
  (MultiAssayExperiment-class), 13
coerce,MultiAssayExperiment,MatchedAssayExperiment-method
  (MultiAssayExperiment-class), 9
coerce,ExperimentList
  (ExperimentList-class), 5
coerce-MultiAssayExperiment
  (MultiAssayExperiment-class), 13
colData, 22
colData,MultiAssayExperiment-method
  (MultiAssayExperiment-methods), 23
colData<-,MultiAssayExperiment,ANY-method
  (MultiAssayExperiment-methods), 23
colData<-,MultiAssayExperiment,DataFrame-method
  (MultiAssayExperiment-methods), 23

CharacterList, 6, 21
coerce,List,ExperimentList-method
  (ExperimentList-class), 5
coerce,list,ExperimentList-method
  (ExperimentList-class), 5
coerce,List,MultiAssayExperiment-method
  (MultiAssayExperiment-class), 13
colnames, ExperimentList-method
  (ExperimentList-class), 5

colnames<-, MultiAssayExperiment, List-method
  (MultiAssayExperiment-methods), 23

colnames<-, MultiAssayExperiment, list-method
  (MultiAssayExperiment-methods), 23

colnames<-, MultiAssayExperiment-method
  (MultiAssayExperiment-methods), 17

colnames<-, MultiAssayExperiment, List-method
  (MultiAssayExperiment-methods), 23

colnames<-, MultiAssayExperiment, list-method
  (MultiAssayExperiment-methods), 23

colnames<-, MultiAssayExperiment, List-method
  (MultiAssayExperiment-methods), 23

complete.cases, MultiAssayExperiment-method
  (MultiAssayExperiment-helpers), 17

DataFrame, 8, 11, 16, 18, 21, 25, 27, 28
DataFrame (reexports), 28

dimnames, ExperimentList-method
  (ExperimentList-class), 5

dimnames, MultiAssayExperiment-method
  (MultiAssayExperiment-methods), 13

drops, MultiAssayExperiment-method
  (MultiAssayExperiment-methods), 23

drops<-, MultiAssayExperiment-method
  (MultiAssayExperiment-methods), 23

drops<-, MultiAssayExperiment-method
  (MultiAssayExperiment-methods), 23

ExperimentList, 4, 6, 7, 11–13, 15, 16, 21,
  23, 25–28
ExperimentList-class, 5

experiments, MultiAssayExperiment-method
  (MultiAssayExperiment-methods), 23

isAssay, 7

isEmpty, ExperimentList-method
  (ExperimentList-class), 5

isEmpty, MultiAssayExperiment-method
  (MultiAssayExperiment-methods), 17

exportClass, MultiAssayExperiment-method
  (MultiAssayExperiment-class), 13

expressionSet, 13, 21

getWithColData
  (MultiAssayExperiment-methods), 17

GRanges, 21, 22
GRangesList, 13

hasAssay, 7

hasRowRanges
  (MultiAssayExperiment-methods), 17

hasRowRanges, ExperimentList-method
  (MultiAssayExperiment-methods), 17

hasRowRanges, MultiAssayExperiment-method
  (MultiAssayExperiment-methods), 17

HDF5MultiAssayExperiment
  (saveHDF5MultiAssayExperiment), 28

intersectColumns
  (MultiAssayExperiment-methods), 17

intersectRows
  (MultiAssayExperiment-methods), 17

isAssay, 7

isAssay, MultiAssayExperiment-method
  (MultiAssayExperiment-methods), 17

length, MultiAssayExperiment-method
  (MultiAssayExperiment-class), 13

List, 7, 15, 16
ListToMap, 8

loadHDF5MultiAssayExperiment
  (saveHDF5MultiAssayExperiment), 28

LogicalList, 6, 20
longFormat
(MultiAssayExperiment-helpers), 17
makeHitList
(MultiAssayExperiment-helpers), 17
makeMatchList
(MultiAssayExperiment-helpers), 17
mapToList(listToMap), 8
MatchedAssayExperiment
(MatchedAssayExperiment-class), 9
MatchedAssayExperiment-class, 9
mcols(), 22
mergeReplicates
(MultiAssayExperiment-helpers), 17
mergeReplicates, ANY-method
(MultiAssayExperiment-helpers), 17
mergeReplicates, ExperimentList-method
(ExperimentList-class), 5
mergeReplicates, MultiAssayExperiment-method
(MultiAssayExperiment-helpers), 17
metadata, MultiAssayExperiment-method
(MultiAssayExperiment-methods), 23
metadata<-, MultiAssayExperiment-method
(MultiAssayExperiment-methods), 23
methods::as, 16
miniACC, 10
MultiAssayExperiment, 9–11, 12, 23, 26,
27, 29
MultiAssayExperiment-class, 13
MultiAssayExperiment-helpers, 17
MultiAssayExperiment-methods, 16, 23
MultiAssayExperiment-package, 3
MultiAssayExperimentToMAF, 26
names, MultiAssayExperiment-method
(MultiAssayExperiment-class), 13
names<-, MultiAssayExperiment-method
(MultiAssayExperiment-methods), 23
prepMultiAssay, 26
reexports, 28
renameColname
(MultiAssayExperiment-helpers), 17
renamePrimary
(MultiAssayExperiment-helpers), 17
replicated
(MultiAssayExperiment-helpers), 17
replicated, MultiAssayExperiment-method
(MultiAssayExperiment-helpers), 17
replicates
(MultiAssayExperiment-helpers), 17
replicates, MultiAssayExperiment-method
(MultiAssayExperiment-helpers), 17
rownames, 6
rownames, ExperimentList-method
(ExperimentList-class), 5
rowRanges, 18, 21
sampleMap, 12, 13, 16, 26, 27
sampleMap
(MultiAssayExperiment-methods), 23
sampleMap, MultiAssayExperiment-method
(MultiAssayExperiment-methods), 23
sampleMap<-
(MultiAssayExperiment-methods), 23
sampleMap<-, MultiAssayExperiment, ANY-method
(MultiAssayExperiment-methods), 23
sampleMap<-, MultiAssayExperiment, DataFrame-method
(MultiAssayExperiment-methods), 23
saveHDF5MultiAssayExperiment, 28
show, ExperimentList-method
(ExperimentList-class), 5
show, MultiAssayExperiment-method
(MultiAssayExperiment-class), 13
showReplicated
   (MultiAssayExperiment-helpers), 17
showReplicated, MultiAssayExperiment-method
   (MultiAssayExperiment-helpers), 17
SimpleList, 12, 15, 16
splitAssays
   (MultiAssayExperiment-helpers), 17
splitAssays, MultiAssayExperiment-method
   (MultiAssayExperiment-helpers), 17
subset (subsetBy), 30
subsetBy, 30
subsetByAssay (subsetBy), 30
subsetByAssay, ExperimentList-method
   (subsetBy), 30
subsetByAssay, MultiAssayExperiment-method
   (subsetBy), 30
subsetByColData (subsetBy), 30
subsetByColData, MultiAssayExperiment, ANY-method
   (subsetBy), 30
subsetByColData, MultiAssayExperiment, character-method
   (subsetBy), 30
subsetByColumn (subsetBy), 30
subsetByColumn, ExperimentList, List-method
   (subsetBy), 30
subsetByColumn, ExperimentList, list-method
   (subsetBy), 30
subsetByColumn, ExperimentList, logical-method
   (subsetBy), 30
subsetByColumn, MultiAssayExperiment, ANY-method
   (subsetBy), 30
subsetByRow (subsetBy), 30
subsetByRow, ExperimentList, ANY-method
   (subsetBy), 30
subsetByRow, ExperimentList, list-method
   (subsetBy), 30
subsetByRow, ExperimentList, logical-method
   (subsetBy), 30
subsetByRow, MultiAssayExperiment, ANY-method
   (subsetBy), 30
SummarizedExperiment, 13, 21, 22
updateObject, MultiAssayExperiment-method
   (MultiAssayExperiment-class), 13
UpSetR::upset, 33
upsetSamples, 32
wideFormat
   (MultiAssayExperiment-helpers), 17
writeHDF5Array, 29