Package 'CoreGx'

October 14, 2021

Type Package

Title Classes and Functions to Serve as the Basis for Other 'Gx' Packages

Version 1.4.2

Date 2021-09-28

Description A collection of functions and classes which serve as the foundation for our lab's suite of R packages, such as 'PharmacoGx' and 'RadioGx'. This package was created to abstract shared functionality from other lab package releases to increase ease of maintainability and reduce code repetition in current and future 'Gx' suite programs. Major features include a 'CoreSet' class, from which 'RadioSet' and 'PharmacoSet' are derived, along with get and set methods for each respective slot. Additional functions related to fitting and plotting dose response curves, quantifying statistical correlation and calculating area under the curve (AUC) or survival fraction (SF) are included. For more details please see the included documentation, as well as:
Smirnov, P., Safikhani, Z., El-Hachem, N., Wang, D., She, A., Olsen, C., Freeman, M., Selby, H., Gendoo, D., Grossman, P., Beck, A., Aerts, H., Lupien, M., Goldenberg, A. (2015) <doi:10.1093/bioinformatics/btv723>.
Manem, V., Labie, M., Smirnov, P., Kofia, V., Freeman, M., Koritzinksy, M., Abazeed, M., Haibe-Kains, B., Bratman, S. (2018) <doi:10.1101/449793>.

VignetteBuilder knitr

VignetteEngine knitr::rmarkdown

biocViews Software, Pharmacogenomics, Classification, Survival

Encoding UTF-8

LazyData true

Depends R (>= 4.1), BiocGenerics, SummarizedExperiment

Imports Biobase, S4Vectors, MultiAssayExperiment, MatrixGenerics, piano, BiocParallel, methods, stats, utils, graphics, grDevices, lsa, data.table, crayon, glue, rlang

Suggests pander, markdown, BiocStyle, rmarkdown, knitr, formatR, testthat

License GPL-3

Roxygen list(markdown = TRUE, r6=FALSE)

RoxygenNote 7.1.1

Collate 'allGenerics.R' 'LongTable-class.R' 'CoreSet-class.R'
'CoreSet-accessors.R' 'DataMapper-class.R'
'LongTable-accessors.R' 'LongTableDataMapper-class.R'
'adaptiveMatthewCor.R' 'callingWaterfall.R'
'connectivityScore.R' 'cosinePerm.R' 'datasets.R' 'globals.R'
'gwc.R' 'matthewCor.R' 'methods-$.R' 'methods-[.R'
'methods-[[.R' 'methods-assay.R' 'methods-assayNames.R'
'methods-assays.R' 'methods-buildLongTable.R'
'methods-coerce.R' 'methods-colData.R' 'methods-dim.R'
'methods-dimnames.R' 'methods-drugSensitivitySig.R'
'methods-getIntern.R' 'methods-guessMapping.R'
'methods-intersect.R' 'methods-metaConstruct.R'
'methods-metadata.R' 'methods-reindex.R' 'methods-rowData.R'
'methods-subset.R' 'methods-subsetTo.R' 'utilities.R'
'utils-iteration.R' 'utils-messages.R' 'utils-updateS4.R'

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

git_branch RELEASE_3_13

git_last_commit 32dc73a

git_last_commit_date 2021-09-28

Date/Publication 2021-10-14

Author Petr Smirnov [aut],
Ian Smith [aut],
Christopher Eeles [aut],
Benjamin Haibe-Kains [aut, cre]

Maintainer Benjamin Haibe-Kains <benjamin.haibe.kains@utoronto.ca>

R topics documented:

. *Convenience function for converting R code to a call*

Description

This is used to pass through unevaluated R expressions into subset and [, where they will be evaluated in the correct context.

Usage

```
.(...)
```

Arguments

...	pairlist One or more R expressions to convert to calls.

Value

call An R call object containing the quoted expression.

Examples

```
.(cell_line1 == 'A2058')
```

---

amcc *Calculate an Adaptive Matthews Correlation Coefficient*

Description

This function calculates an Adaptive Matthews Correlation Coefficient (AMCC) for two vectors of values of the same length. It assumes the entries in the two vectors are paired. The Adaptive Matthews Correlation Coefficient for two vectors of values is defined as the Maximum Matthews Coefficient over all possible binary splits of the ranks of the two vectors. In this way, it calculates the best possible agreement of a binary classifier on the two vectors of data. If the AMCC is low, then it is impossible to find any binary classification of the two vectors with a high degree of concordance.

Usage

```
amcc(x, y, step.prct = 0, min.cat = 3, nperm = 1000, nthread = 1, ...)
```

Arguments

x, y	Two paired vectors of values. Could be replicates of observations for the same experiments for example.
step.prct	Instead of testing all possible splits of the data, it is possible to test steps of a percentage size of the total number of ranks in x/y. If this variable is 0, function defaults to testing all possible splits.
min.cat	The minimum number of members per category. Classifications with less members fitting into both categories will not be considered.
nperm	The number of perumatation to use for estimating significance. If 0, then no p-value is calculated.
nthread	Number of threads to parallize over. Both the AMCC calculation and the permutation testing is done in parallel.
...	Additional arguments

Value

Returns a list with two elements. $amcc contains the highest 'mcc' value over all the splits, the p value, as well as the rank at which the split was done.

Examples

```
x <- c(1,2,3,4,5,6,7)
y <- c(1,3,5,4,2,7,6)
amcc(x,y, min.cat=2)
```

---

as *LongTable to data.table conversion*

Description

Coerce a LongTable into a data.table.

Currently only supports coercing to data.table or data.frame

Coerce a data.table with the proper configuration attributes back to a LongTable

Arguments

to	character Class name to coerce to, currently only 'data.table' and 'data.frame' are supported
from	A data.table with the 'LongTable.config' attribute, containing three lists named assayCols, rowDataCols and colDataCols. This attribute is automatically created when coercing from a LongTable to a data.table.

Value

A `data.table` with the data from a LongTable.

`data.table` containing the data from the LongTable, with the 'LongTable.config' attribute containg the metadata needed to reverse the coercing operation.

`LongTable` object configured with the LongTable.config

Examples

```
as(merckLongTable, 'data.table')

dataTable <- as(merckLongTable, 'data.table')
print(attr(dataTable, 'LongTable.config')) # Method doesn't work without this
as(dataTable, 'LongTable')
```

---

as.data.frame.LongTable

Coerce a LongTable to a data.frame

Description

S3 version of coerce method fro convenience.

Usage

```
## S3 method for class 'LongTable'
as.data.frame(x, row.names, optional = TRUE, ...)
```

Arguments

`x`	`LongTable` to coerce to `data.frame`.
`row.names`	An optional `character` vector of rownames. We do not recommend using this parameter, it is included for S3 method consistency with `as.data.frame`.
`optional`	`logical` Is it optional for row and column names to be valid R names? If FALSE will use the make.names function to ensure the row and column names are valid R names. Defaults to TRUE.
`...`	Does nothing.

Value

`data.frame` containing the data from the LongTable, with the 'LongTable.config' attribute containg the metadata needed to reverse the coercion operation.

Examples

```
as(merckLongTable, 'data.frame')
```

as.data.table.LongTable *Coerce a LongTable into a* data.table

Description

S3 version of coerce method for convenience.

Usage

```
## S3 method for class 'LongTable'
as.data.table(x)
```

Arguments

x	LongTable to coerce to a data.table

Value

A data.table containing the data from the LongTable, as well as the 'LongTable.config' attribute which contains the data needed to reverse the coercion.

as.long.table *Coerce from data.table to LongTable*

Description

Coerce a data.table with the proper configuration attributes back to a LongTable

Usage

```
as.long.table(x)
```

Arguments

x	A data.frame with the 'LongTable.config' attribute, containing three lists named assayCols, rowDataCols and colDataCols. This attribute is automatically created when coercing from a LongTable to a data.table.

Value

LongTable object configured with the LongTable.config

Examples

```
dataTable <- as(merckLongTable, 'data.table')
print(attr(dataTable, 'LongTable.config')) # Method doesn't work without this
as.long.table(dataTable)
```

`assayCols` *Generic to access the assay columns of a rectangular object.*

Description

Generic to access the assay columns of a rectangular object.

Usage

```
assayCols(object, ...)
```

Arguments

`object`	`S4` An object to get assay ids from.
`...`	Allow new arguments to this generic.

Value

Depends on the implemented method.

Examples

```
print("Generics shouldn't need examples?")
```

`buildLongTable` *Build a LongTable object*

Description

Build a LongTable object

Usage

```
buildLongTable(from, ...)
```

Arguments

`from`	What to build the LongTable from?
`...`	`pairlist` Allow definition of new parameters for implementations of this generic.

Value

Depends on the implemented method

Examples

```
print("Generics shouldn't need examples?")
```

`buildLongTable,list-method`
LongTable build method from list

Description

LongTable build method from list

Usage

```
## S4 method for signature 'list'
buildLongTable(from, rowDataCols, colDataCols, assayCols)
```

Arguments

`from`	`list` A list containing any combination of character file paths, data.tables and data.frames which will be used to construct the LongTable.
`rowDataCols`	`list` List with two `character` vectors, the first specifying one or more columns to be used as cell identifiers (e.g., cell-line name columns) and the second containing any additional metadata columns related to the cell identifiers.
`colDataCols`	`list` List with two `character` vectors, the first specifying one or more columns to be used as column identifiers (e.g., drug name columns) and the second containing any additional metadata columns related to the column identifiers.
`assayCols`	`list` A named list of character vectors specifying how to parse assay columns into a list of `data.tables`. Each list data.table will be named for the name of corresponding list item and contain the columns specified in the character vector of column names in each list item.

Value

A `LongTable` object constructed with the data in `from`.

Functions

- `buildLongTable,list-method`: a LongTable object from a list containing file paths, data.frames and data.tables.

Examples

```
assayList <- assays(merckLongTable, withDimnames=TRUE)
rowDataCols <- list(rowIDs(merckLongTable), rowMeta(merckLongTable))
colDataCols <- list(colIDs(merckLongTable), colMeta(merckLongTable))
assayCols <- assayCols(merckLongTable)
longTable <- buildLongTable(from=assayList, rowDataCols, colDataCols, assayCols)
```

`checkColumnCardinality`

Search a data.frame for 1:`cardinality` *relationships between a group of columns (your identifiers) and all other columns.*

Description

Search a data.frame for 1:`cardinality` relationships between a group of columns (your identifiers) and all other columns.

Usage

```
checkColumnCardinality(df, group, cardinality = 1, ...)
```

Arguments

`df`	A `data.frame` to search for 1:`cardinality` mappings with the columns in `group`.
`group`	A `character` vector of one or more column names to check the cardinality of other columns against.
`cardinality`	The cardinality of to search for (i.e., 1:`cardinality`) relationships with the combination of columns in group. Defaults to 1 (i.e., 1:1 mappings).
`...`	Fall through arguments to data.table::`[`. For developer use. One use case is setting verbose=TRUE to diagnose slow data.table operations.

Value

A `character` vector with the names of the columns with cardinality of 1:`cardinality` with the columns listed in `group`.

Examples

```
df <- rawdata(exampleDataMapper)
checkColumnCardinality(df, group='drug_id')
```

checkCsetStructure *A function to verify the structure of a CoreSet*

Description

This function checks the structure of a PharamcoSet, ensuring that the correct annotations are in place and all the required slots are filled so that matching of cells and drugs can be properly done across different types of data and with other studies.

Usage

```
checkCsetStructure(cSet, plotDist = FALSE, result.dir = ".")
```

Arguments

cSet	A CoreSet to be verified
plotDist	Should the function also plot the distribution of molecular data?
result.dir	The path to the directory for saving the plots as a string

Value

Prints out messages whenever describing the errors found in the structure of the cSet object passed in.

Examples

```
checkCsetStructure(clevelandSmall_cSet)
```

clevelandSmall_cSet *Cleaveland_mut RadioSet subsetted and cast as CoreSet*

Description

This dataset is just a dummy object derived from the Cleveland_mut RadioSet in the RadioGx R package. It's contents should not be interpreted and it is only present to test the functions in this package and provide examples

Usage

```
data(clevelandSmall_cSet)
```

Format

CoreSet object

References

Lamb et al. The Connectivity Map: using gene-expression signatures to connect small molecules, genes, and disease. Science, 2006.

`colIDs` *Generic to access the row identifiers for an object.*

Description

Generic to access the row identifiers for an object.

Usage

```
colIDs(object, ...)
```

Arguments

`object`	`S4` An object to get column id columns from.
`...`	ALlow new arguments to this generic

Value

Depends on the implemented method.

Examples

```
print("Generics shouldn't need examples?")
```

`colMeta` *Generic to access the column identifiers for a rectangular object.*

Description

Generic to access the column identifiers for a rectangular object.

Usage

```
colMeta(object, ...)
```

Arguments

`object`	`S4` An object to get column metadata columns from.
`...`	ALlow new arguments to this generic

Value

Depends on impemented method.

Examples

```
print("Generics shouldn't need examples?")
```

`connectivityScore` *Function computing connectivity scores between two signatures*

Description

A function for finding the connectivity between two signatures, using either the GSEA method based on the KS statistic, or the gwc method based on a weighted spearman statistic. The GSEA analysis is implemented in the piano package.

Usage

```
connectivityScore(
  x,
  y,
  method = c("fgsea", "gwc"),
  nperm = 10000,
  nthread = 1,
  gwc.method = c("spearman", "pearson"),
  ...
)
```

Arguments

`x`	A `matrix` with the first gene signature. In the case of GSEA the vector of values per gene for GSEA in which we are looking for an enrichment. In the case of gwc, this should be a matrix, with the per gene responses in the first column, and the significance values in the second.
`y`	A `matrix` with the second signature. In the case of GSEA, this is the vector of up and down regulated genes we are looking for in our signature, with the direction being determined from the sign. In the case of gwc, this should be a matrix of identical size to x, once again with the per gene responses in the first column, and their significance in the second.
`method`	`character` string identifying which method to use, out of 'fgsea' and 'gwc'
`nperm`	`numeric`, how many permutations should be done to determine significance through permutation testing? The minimum is 100, default is 1e4.
`nthread`	`numeric`, how many cores to run parallel processing on.
`gwc.method`	`character`, should gwc use a weighted spearman or pearson statistic?
`...`	Additional arguments passed down to gsea and gwc functions

Value

numeric a numeric vector with the score and the p-value associated with it

References

F. Pozzi, T. Di Matteo, T. Aste, 'Exponential smoothing weighted correlations', The European Physical Journal B, Vol. 85, No 6, 2012. DOI: 10.1140/epjb/e2012-20697-x

Varemo, L., Nielsen, J. and Nookaew, I. (2013) Enriching the gene set analysis of genome-wide data by incorporating directionality of gene expression and combining statistical hypotheses and methods. Nucleic Acids Research. 41 (8), 4378-4391. doi: 10.1093/nar/gkt111

Examples

```
xValue <- c(1,5,23,4,8,9,2,19,11,12,13)
xSig <- c(0.01, 0.001, .97, 0.01,0.01,0.28,0.7,0.01,0.01,0.01,0.01)
yValue <- c(1,5,10,4,8,19,22,19,11,12,13)
ySig <- c(0.01, 0.001, .97,0.01, 0.01,0.78,0.9,0.01,0.01,0.01,0.01)
xx <- cbind(xValue, xSig)
yy <- cbind(yValue, ySig)
rownames(xx) <- rownames(yy) <- c('1','2','3','4','5','6','7','8','9','10','11')
data.cor <- connectivityScore(xx,yy,method='gwc', gwc.method='spearman', nperm=300)
```

CoreSet *CoreSet constructor*

Description

A constructor that simplifies the process of creating CoreSets, as well as creates empty objects for data not provided to the constructor. Only objects returned by this constructor are expected to work with the CoreSet methods.

Usage

```
CoreSet(
  name,
  molecularProfiles = list(),
  cell = data.frame(),
  sensitivityInfo = data.frame(),
  sensitivityRaw = array(dim = c(0, 0, 0)),
  sensitivityProfiles = matrix(),
  sensitivityN = matrix(nrow = 0, ncol = 0),
  perturbationN = array(NA, dim = c(0, 0, 0)),
  curationCell = data.frame(),
  curationTissue = data.frame(),
  datasetType = c("sensitivity", "perturbation", "both"),
  verify = TRUE
)
```

Arguments

name	A character string detailing the name of the dataset
molecularProfiles	A list of SummarizedExperiment objects containing molecular profiles for each molecular data type.
cell	A data.frame containing the annotations for all the cell lines profiled in the data set, across all data types
sensitivityInfo	A data.frame containing the information for the sensitivity experiments
sensitivityRaw	A 3 Dimensional array contaning the raw drug dose response data for the sensitivity experiments
sensitivityProfiles	data.frame containing drug sensitivity profile statistics such as IC50 and AUC
sensitivityN, perturbationN	A data.frame summarizing the available sensitivity/perturbation data
curationCell, curationTissue	A data.frame mapping the names for cells and tissues used in the data set to universal identifiers used between different CoreSet objects
datasetType	A character string of 'sensitivity', 'preturbation', or both detailing what type of data can be found in the CoreSet, for proper processing of the data
verify	boolean Should the function verify the CoreSet and print out any errors it finds after construction?

Value

An object of class CoreSet

CoreSet-accessors *Accessing and modifying information in a* CoreSet

---

Description

Documentation for the various setters and getters which allow manipulation of data in the slots of a CoreSet object.

Usage

```
## S4 method for signature 'CoreSet'
annotation(object)

## S4 replacement method for signature 'CoreSet,list'
annotation(object) <- value

## S4 method for signature 'CoreSet'
```

```
dateCreated(object)

## S4 replacement method for signature 'CoreSet,character'
dateCreated(object) <- value

## S4 method for signature 'CoreSet'
name(object)

## S4 replacement method for signature 'CoreSet'
name(object) <- value

## S4 method for signature 'CoreSet'
cellInfo(object)

## S4 replacement method for signature 'CoreSet,data.frame'
cellInfo(object) <- value

## S4 method for signature 'CoreSet'
cellNames(object)

## S4 replacement method for signature 'CoreSet,character'
cellNames(object) <- value

## S4 method for signature 'CoreSet'
curation(object)

## S4 replacement method for signature 'CoreSet,list'
curation(object) <- value

## S4 method for signature 'CoreSet'
datasetType(object)

## S4 replacement method for signature 'CoreSet,character'
datasetType(object) <- value

## S4 method for signature 'CoreSet'
molecularProfiles(object, mDataType, assay)

## S4 replacement method for signature 'CoreSet,character,character,matrix'
molecularProfiles(object, mDataType, assay) <- value

## S4 method for signature 'CoreSet'
featureInfo(object, mDataType)

## S4 replacement method for signature 'CoreSet,character,data.frame'
featureInfo(object, mDataType) <- value

## S4 method for signature 'CoreSet,character'
```

```
phenoInfo(object, mDataType)

## S4 replacement method for signature 'CoreSet,character,data.frame'
phenoInfo(object, mDataType) <- value

## S4 method for signature 'CoreSet,character'
fNames(object, mDataType)

## S4 replacement method for signature 'CoreSet,character,character'
fNames(object, mDataType) <- value

## S4 method for signature 'CoreSet'
mDataNames(object)

## S4 replacement method for signature 'CoreSet'
mDataNames(object) <- value

## S4 method for signature 'CoreSet'
molecularProfilesSlot(object)

## S4 replacement method for signature 'CoreSet,list_or_MAE'
molecularProfilesSlot(object) <- value

## S4 method for signature 'CoreSet'
sensitivityInfo(object, dimension, ...)

## S4 replacement method for signature 'CoreSet,data.frame'
sensitivityInfo(object, dimension, ...) <- value

## S4 method for signature 'CoreSet'
sensitivityMeasures(object)

## S4 replacement method for signature 'CoreSet,character'
sensitivityMeasures(object) <- value

## S4 method for signature 'CoreSet'
sensitivityProfiles(object)

## S4 replacement method for signature 'CoreSet,data.frame'
sensitivityProfiles(object) <- value

## S4 method for signature 'CoreSet'
sensitivityRaw(object)

## S4 replacement method for signature 'CoreSet,array'
sensitivityRaw(object) <- value

## S4 method for signature 'CoreSet'
```

```
sensitivitySlot(object)

## S4 replacement method for signature 'CoreSet,list_or_LongTable'
sensitivitySlot(object) <- value

## S4 method for signature 'CoreSet'
sensNumber(object)

## S4 replacement method for signature 'CoreSet,matrix'
sensNumber(object) <- value

## S4 method for signature 'CoreSet'
pertNumber(object)

## S4 replacement method for signature 'CoreSet,array'
pertNumber(object) <- value
```

Arguments

object	A CoreSet object.
value	See details.
mDataType	character(1) The name of a molecular datatype to access from the molecularProfiles of a CoreSet object.
assay	character(1) A valid assay name in the SummarizedExperiment of @molecularProfiles of a CoreSet object for data type mDataType.
dimension	See details.
...	See details.

Details

@annotation:

annotation: A list of CoreSet annotations with items: 'name', the name of the object; 'dateCreated', date the object was created; 'sessionInfo', the sessionInfo() when the object was created; 'call', the R constructor call; and 'version', the object version.

annotation<-: Setter method for the annotation slot. Arguments:

- value: a list of annotations to update the CoreSet with.

@dateCreated:

dateCreated: character(1) The date the CoreSet object was created, as returned by the date() function.

dateCreated<-: Update the 'dateCreated' item in the annotation slot of a CoreSet object. Arguments:

- value: A character(1) vector, as returned by the date() function.

name: character(1) The name of the CoreSet, retreived from the @annotation slot.

name<-: Update the @annotation$name value in a CoreSet object.

- value: `character(1)` The name of the `CoreSet` object.

cellInfo: `data.frame` Metadata for all cell-lines in a `CoreSet` object.

cellInfo<-: assign updated cell-line annotations to the `CoreSet` object. Arguments:

- value: a `data.frame` object.

cellNames: `character` Retrieve the rownames of the `data.frame` in the `cell` slot from a CoreSet object.

cellNames<-: assign new rownames to the cellInfo slot `data.frame` for a CoreSet object. Arguments:

- value: `character` vector of rownames for the `cellInfo(object)` `data.frame`.

@curation:

curation: A `list` of curated mappings between identifiers in the CoreSet object and the original data publication. Contains two `data.frames`, 'cell' with cell-line ids and 'tissue' with tissue ids.

curation<-: Update the `curation` slot of a CoreSet object. Arugments:

- value: A `list` of `data.frames`, one for each type of curated identifier. For a `CoreSet` object the slot should contain tissue and cell-line id `data.frames`.

datasetType slot:

datasetType: `character(1)` The type treatment response in the `sensitivity` slot. Valid values are 'sensitivity', 'perturbation' or 'both'.

datasetType<-: Update the datasetType slot of a CoreSet object. Arguments:

- value: A `character(1)` vector with one of 'sensitivity', 'perturbation' or 'both'

@molecularProfiles:

molecularProfiles: `matrix()` Retrieve an assay in a `SummarizedExperiment` from the `molecularProfiles` slot of a `CoreSet` object with the specified `mDataType`. Valid `mDataType` arguments can be found with `mDataNames(object)`. Arguments:

- assay: Optional `character(1)` vector specifying an assay in the `SummarizedExperiment` of the `molecularProfiles` slot of the `CoreSet` object for the specified `mDataType`. If excluded, defaults to modifying the first assay in the `SummarizedExperiment` for the given `mDataType`.

molecularProfiles<-: Update an assay in a `SummarizedExperiment` from the `molecularProfiles` slot of a CoreSet object with the specified `mDataType`. Valid `mDataType` arguments can be found with `mDataNames(object)`.

- assay: Optional `character(1)` vector specifying an assay in the `SummarizedExperiment` of the `molecularProfiles` slot of the `CoreSet` object for the specified `mDataType`. If excluded, defaults to modifying the first assay in the `SummarizedExperiment` for the given `mDataType`.
- value: A `matrix` of values to assign to the `assay` slot of the `SummarizedExperiment` for the selected `mDataType`. The rownames and column names must match the associated `SummarizedExperiment`.

featureInfo: Retrieve a DataFrame of feature metadata for the specified mDataType from the molecularProfiles slot of a CoreSet object. More specifically, retrieve the @rowData slot from the SummarizedExperiment from the @molecularProfiles of a CoreSet object with the name mDataType.

featureInfo<-: Update the featureInfo(object,mDataType) DataFrame with new feature metadata. Arguments:

- value: A data.frame or DataFrame with updated feature metadata for the specified molecular profile in the molecularProfiles slot of a CoreSet object.

phenoInfo: Return the @colData slot from the SummarizedExperiment of mDataType, containing sample-level metadata, from a CoreSet object.

phenoInfo<-: Update the @colData slot of the SummarizedExperiment of mDataType in the @molecularProfiles slot of a CoreSet object. This updates the sample-level metadata in-place.

- value: A data.frame or DataFrame object where rows are samples and columns are sample metadata.

fNames: character() The features names from the rowData slot of a SummarizedExperiment of mDataType within a CoreSet object.

fNames: Updates the rownames of the feature metadata (i.e., rowData) for a SummarizedExperiment of mDataType within a CoreSet object.

- value: character() A character vector of new features names for the rowData of the SummarizedExperiment of mDataType in the @molecularProfiles slot of a CoreSet object. Must be the same length as nrow(featureInfo(object,mDataType)), the number of rows in the feature metadata.

mDataNames: character Retrieve the names of the molecular data types available in the molecularProfiles slot of a CoreSet object. These are the options which can be used in the mDataType parameter of various molecularProfiles slot accessors methods.

mDataNames: Update the molecular data type names of the molecularProfiles slot of a CoreSet object. Arguments:

- value: character vector of molecular datatype names, with length equal to length(molecularProfilesSlot(object

molecularProfilesSlot: Return the contents of the @molecularProfiles slot of a CoreSet object. This will either be a list or MultiAssayExperiment of SummarizedExperiments.

molecularProfilesSlot<-: Update the contents of the @molecularProfiles slot of a CoreSet object. Arguemnts:

- value: A list or MultiAssayExperiment of SummarizedExperiments. The list and assays should be named for the molecular datatype in each SummarizedExperiment.

@sensitivity:

Arguments::

- dimension: Optional character(1) One of 'drug', 'cell' or 'assay' to retrieve rowData, colData or the 'assay_metadata' assay from the CoreSet @sensitvity LongTable object, respectively. Ignored with warning if @sensitivity is not a LongTable object.

- ...: Additional arguments to the rowData or colData. LongTable methods. Only used if the sensitivity slot contains a LongTable object instead of a list and the dimension argument is specified.

Methods::
sensitivityInfo: DataFrame or data.frame of sensitivity drug combo by cell-line metadata for the CoreSet object. When the dimension parameter is used, it allows retrieval of the dimension specific metadata from the LongTable object in @sensitivity of a CoreSet object.

sensitivityInfo<-: Update the @sensitivity slot metadata for a CoreSet object. When used without the dimension argument is behaves similar to the old CoreSet implementation, where the @sensitivity slot contained a list with a $info data.frame item. When the dimension arugment is used, more complicated assignments can occur where 'cell' modifies the @sensitvity LongTable colData, 'drug' the rowData and 'assay' the 'assay_metadata' assay. Arguments:

- value: A data.frame of treatment response experiment metadata, documenting experiment level metadata (mapping to drugs and cells). If the @sensitivity slot doesn't contain a LongTable and dimension is not specified, you can only modify existing columns as returned by sensitivityInfo(object).

sensitivityMeaures: Get the 'sensitivityMeasures' available in a CoreSet object. Each measure reprents some summary of cell-line sensitivity to a given dryuh, such as ic50, ec50, AUC, AAC, etc. The results are returned as a character vector with all available metrics for the PSet object.

sensitivityMeaures: Update the sensitivity meaure in a CoreSet object. Thesee values are the column names of the 'profiles' assay and represent various compued sensitviity metrics such as ic50, ec50, AUC, AAC, etc.

- value: A character vector of new sensitivity measure names, the then length of the character vector must matcht he number of columns of the 'profiles' assay, excluding metadata and key columns.

sensitivityProfiles: Return the sensitivity profile summaries from the sensitivity slot. This data.frame cotanins vaarious sensitivity summary metrics, such as ic50, amax, EC50, aac, HS, etc as columns, with rows as drug by sample experiments.

sensitivityProfiles<-: Update the sensitivity profile summaries the sensitivity slot. Arguments: - value: A data.frame the the same number of rows as as returned by sensitivityProfiles(object), but potentially modified columns, such as the computation of additional summary metrics.

sensitivityRaw: Access the raw sensitiity measurents for a CoreSet object. A 3D array where rows are experiment_ids, columns are doses and the third dimension is metric, either 'Dose' for the doses used or 'Viability' for the cell-line viability at that dose.

sensitvityRaw<-: Update the raw dose and viability data in a CoreSet.

- value: A 3D array object where rows are experiment_ids, columns are replicates and pages are c('Dose', 'Viability'), with the corresponding dose or viability measurement for that experiment_id and replicate.

sensNumber: Return a count of viability observations in a CoreSet object for each drug-combo by cell-line combination.

sensNumber<-: Update the 'n' item, which holds a matrix with a count of drug by cell-line experiment counts, in the list in @sensitivity slot of a CoreSet object. Will error when @sensitvviity contains a LongTable object, since the counts are computed on the fly. Arguments:

- value: A `matrix` where rows are cells and columns are drugs, with a count of the number of experiments for each combination as the values.

pertNumber: `array` Summary of available perturbation experiments from in a `CoreSet` object. Returns a 3D `array` with the number of perturbation experiments per drug and cell line, and data type.

pertNumber<-: Update the @perturbation$n value in a `CoreSet` object, which stores a summary of the available perturbation experiments. Arguments:

- value: A new 3D `array` with the number of perturbation experiments per drug and cell line, and data type

Value

Accessors: See details.

Setters: An updated `CoreSet` object, returned invisibly.

Examples

```
data(clevelandSmall_cSet)

## @annotation

annotation(clevelandSmall_cSet)

annotation(clevelandSmall_cSet) <- annotation(clevelandSmall_cSet)

dateCreated(clevelandSmall_cSet)

## dateCreated
dateCreated(clevelandSmall_cSet) <- date()

name(clevelandSmall_cSet)

name(clevelandSmall_cSet) <- 'new_name'

cellInfo(clevelandSmall_cSet) <- cellInfo(clevelandSmall_cSet)

cellNames(clevelandSmall_cSet)

cellNames(clevelandSmall_cSet) <- cellNames(clevelandSmall_cSet)

## curation
curation(clevelandSmall_cSet)

curation(clevelandSmall_cSet) <- curation(clevelandSmall_cSet)

datasetType(clevelandSmall_cSet)

datasetType(clevelandSmall_cSet) <- 'both'

# No assay specified
```

```
molecularProfiles(clevelandSmall_cSet, 'rna') <- molecularProfiles(clevelandSmall_cSet, 'rna')

# Specific assay
molecularProfiles(clevelandSmall_cSet, 'rna', 'exprs') <-
    molecularProfiles(clevelandSmall_cSet, 'rna', 'exprs')

featureInfo(clevelandSmall_cSet, 'rna')

featureInfo(clevelandSmall_cSet, 'rna') <- featureInfo(clevelandSmall_cSet, 'rna')

phenoInfo(clevelandSmall_cSet, 'rna')

phenoInfo(clevelandSmall_cSet, 'rna') <- phenoInfo(clevelandSmall_cSet, 'rna')

fNames(clevelandSmall_cSet, 'rna')

fNames(clevelandSmall_cSet, 'rna') <- fNames(clevelandSmall_cSet, 'rna')

mDataNames(clevelandSmall_cSet)

mDataNames(clevelandSmall_cSet) <- mDataNames(clevelandSmall_cSet)

molecularProfilesSlot(clevelandSmall_cSet)

molecularProfilesSlot(clevelandSmall_cSet) <- molecularProfilesSlot(clevelandSmall_cSet)

sensitivityInfo(clevelandSmall_cSet)

sensitivityInfo(clevelandSmall_cSet) <- sensitivityInfo(clevelandSmall_cSet)

sensitivityMeasures(clevelandSmall_cSet) <- sensitivityMeasures(clevelandSmall_cSet)

sensitivityMeasures(clevelandSmall_cSet) <- sensitivityMeasures(clevelandSmall_cSet)

sensitivityProfiles(clevelandSmall_cSet)

sensitivityProfiles(clevelandSmall_cSet) <- sensitivityProfiles(clevelandSmall_cSet)

head(sensitivityRaw(clevelandSmall_cSet))

sensitivityRaw(clevelandSmall_cSet) <- sensitivityRaw(clevelandSmall_cSet)

sensitivitySlot(clevelandSmall_cSet)

sensitivitySlot(clevelandSmall_cSet) <- sensitivitySlot(clevelandSmall_cSet)

sensNumber(clevelandSmall_cSet)

sensNumber(clevelandSmall_cSet) <- sensNumber(clevelandSmall_cSet)

pertNumber(clevelandSmall_cSet)

pertNumber(clevelandSmall_cSet) <- pertNumber(clevelandSmall_cSet)
```

`CoreSet-class` *CoreSet - A generic data container for molecular profiles and treatment response data*

Description

CoreSet - A generic data container for molecular profiles and treatment response data

Details

The CoreSet (CSet) class was developed as a superclass for pSets in the PharmacoGx and RadioGx packages to contain the data generated in screens of cancer cell lines for their genetic profile and sensitivities to therapy (Pharmacological or Radiation). This class is meant to be a superclass which is contained within the PharmacoSet (pSet) and RadioSet (RSet) objects exported by PharmacoGx and RadioGx. The format of the data is similar for both pSets and rSets, allowing much of the code to be abstracted into the CoreSet super-class. However, the models involved with quantifying cellular response to Pharmacological and Radiation therapy are widely different, and extension of the cSet class allows the packages to apply the correct model for the given data.

Slots

`annotation` See Slots section.

`molecularProfiles` See Slots section.

`cell` See Slots section.

`curation` See Slots section.

`sensitivity` See Slots section.

`perturbation` See Slots section.

`curation` See Slots section.

`datasetType` See Slots section.

Slots

- annotation: A `list` of annotation data about the `CoreSet`, including the $name and the session information for how the object was created, detailing the exact versions of R and all the packages used.
- molecularProfiles: A `list` or `MultiAssayExperiment` containing `CoreSet` object.
- cell: A `data.frame` containg the annotations for all the cell lines profiled in the data set, across all molecular data types and treatment response experiments.
- sensitivity: A `list` or `LongTable` containing all the data for the sensitivity experiments, including $info, a `data.frame` containing the experimental info, $raw a 3D `array` containing raw data, $profiles, a `data.frame` containing sensitivity profiles statistics, and $n, a `data.frame` detailing the number of experiments for each cell-drug/radiationInfo pair

- perturbation: `list` containing $n, a `data.frame` summarizing the available perturbation data. This slot is currently being deprecated.
- curation: `list` containing mappings for `cell`, `tissue` names used in the data set to universal identifiers used between different `CoreSet` objects
- datasetType: `character` string of 'sensitivity', 'perturbation', or both detailing what type of data can be found in the CoreSet, for proper processing of the data

See Also

CoreSet-accessors

`cosinePerm` *Cosine Permutations*

Description

Computes the cosine similarity and significance using permutation test. This function uses random numbers, to ensure reproducibility please call `set.seed()` before running the function.

Usage

```
cosinePerm(
  x,
  y,
  nperm = 1000,
  alternative = c("two.sided", "less", "greater"),
  include.perm = FALSE,
  nthread = 1,
  ...
)
```

Arguments

`x`	`factor` is the factors for the first variable
`y`	`factor` is the factors for the second variable
`nperm`	`integer` is the number of permutations to compute the null distribution of MCC estimates
`alternative`	`string` indicates the alternative hypothesis and must be one of ''two.sided", ''greater" or ''less". You can specify just the initial letter. ''greater" corresponds to positive association, ''less" to negative association. Options are 'two.sided', 'less', or 'greater'
`include.perm`	`boolean` indicates whether the estimates for the null distribution should be returned. Default set to 'FALSE'
`nthread`	`integer` is the number of threads to be used to perform the permutations in parallel
`...`	A `list` of fallthrough parameters

Value

A list estimate of the cosine similarity, p-value and estimates after random permutations (null distribution) in include.perm is set to 'TRUE'

Examples

```
x <- factor(c(1,2,1,2,1))
y <- factor(c(2,2,1,1,1))
cosinePerm(x, y)
```

DataMapper-accessors *Accessing and modifying data in a* DataMapper *object.*

Description

Documentation for the various setters and getters which allow manipulation of data in the slots of a DataMapper object.

Usage

```
## S4 method for signature 'DataMapper'
rawdata(object)
```

Arguments

object	A DataMapper object to get or set data from.

Details

rawdata: Get the raw data slot from a DataMapper object. Returns a list-like containing one or more raw data inputs to the DataMapper object.

Value

Accessors: See details

Setters: An update DataMapper object, returned invisibly.

See Also

Other DataMapper-accessors: LongTableDataMapper-accessors

DataMapper-class — *An S4 Class For Mapping from Raw Experimental Data to a Specific S4 Object*

Description

This object will be used as a way to abstract away data preprocessing.

Slots

- rawdata: A list-like object containing one or more pieces of raw data that will be processed and mapped to the slots of an `S4` object.
- metadata: A `List` of object level metadata.

exampleDataMapper — *Example LongTableDataMapper*

Description

A dummy LongTableDataMapper object to be used in package examples.

Usage

```
data(exampleDataMapper)
```

Format

LongTableDataMapper object

getIntern — *Retrieve the symbol for the object@.intern slot*

Description

Internal slot for storing metadata relevant to the internal operation of an S4 object.

Usage

```
getIntern(object, x, ...)
```

Arguments

object	S4 An object with an @.itern slot containing an environment.
x	character One or more symbol names to retrieve from the object@.intern environment.
...	Allow new parmeters to be defined for this generic.

Details

Warning: This method is intended for developer use and can be ignored by users.

Value

Depends on the implemented method

Examples

```
print("Generics shouldn't need examples?")
```

---

guessMapping *Generic for Guessing the Mapping Between Some Raw Data and an S4 Object*

Description

Generic for Guessing the Mapping Between Some Raw Data and an S4 Object

Usage

```
guessMapping(object, ...)
```

Arguments

object	An S4 object containing so raw data to guess data to object slot mappings for.
...	Allow new arguments to be defined for this generic.

Value

A list with mapping guesses as items.

guessMapping,LongTableDataMapper-method

Guess which columns in raw experiment data map to which dimensions.

Description

Checks for columns which are uniquely identified by a group of identifiers. This should be used to help identify the columns required to uniquely identify the rows, columns, assays and metadata of a DataMapper class object.

Usage

```
## S4 method for signature 'LongTableDataMapper'
guessMapping(object, groups, subset, data = FALSE)
```

Arguments

object	A LongTableDataMapper object.
groups	A list containing one or more vector of column names to group-by. The function uses these to determine 1:1 mappings between the combination of columns in each vector and unique values in the raw data columns.
subset	A logical vector indicating whether to to subset out mapped columns after each grouping. Must be a single TRUE or FALSE or have the same length as groups, indicating whether to subset out mapped columns after each grouping. This will prevent mapping a column to two different groups.
data	A logical vector indicating whether you would like the data for mapped columns to be returned instead of their column names. Defaults to FALSE for easy use assigning mapped columns to a DataMapper object.

Details

Any unmapped columns will be added to the end of the returned list in an item called unmapped.

The function automatically guesses metadata by checking if any columns have only a single value. This is returned as an additional item in the list.

Value

A list, where each item is named for the associated groups item the guess is for. The character vector in each item are columns which are uniquely identified by the identifiers from that group.

Examples

```
guessMapping(exampleDataMapper, groups=list(rows='drug_id', cols='cell_id'),
subset=FALSE)
```

gwc *GWC Score*

Description

Calculate the gwc score between two vectors, using either a weighted spearman or pearson correlation

Usage

```
gwc(
  x1,
  p1,
  x2,
  p2,
  method.cor = c("pearson", "spearman"),
  nperm = 10000,
  truncate.p = 1e-16,
  ...
)
```

Arguments

x1	numeric vector of effect sizes (e.g., fold change or t statitsics) for the first experiment
p1	numeric vector of p-values for each corresponding effect size for the first experiment
x2	numeric effect size (e.g., fold change or t statitsics) for the second experiment
p2	numeric vector of p-values for each corresponding effect size for the second experiment
method.cor	character string identifying if a pearson or spearman correlation should be used
nperm	numeric how many permutations should be done to determine
truncate.p	numeric Truncation value for extremely low p-values
...	Other passed down to internal functions

Value

numeric a vector of two values, the correlation and associated p-value.

Examples

```
data(clevelandSmall_cSet)
x <- molecularProfiles(clevelandSmall_cSet,'rna')[,1]
y <- molecularProfiles(clevelandSmall_cSet,'rna')[,2]
```

```
x_p <- rep(0.05, times=length(x))
y_p <- rep(0.05, times=length(y))
names(x_p) <- names(x)
names(y_p) <- names(y)
gwc(x,x_p,y,y_p, nperm=100)
```

`idCols` *Generic to access the unique id columns in an S4 object used to*

Description

Generic to access the unique id columns in an S4 object used to

Usage

```
idCols(object, ...)
```

Arguments

`object`	An S4 object to get id columns from.
`...`	Allow new arguments to this generic.

Value

Depends on the implemented method

Examples

```
print("Generics shouldn't need examples?")
```

`idCols,LongTable-method` *Retrieve the unique identifier columns used for primary keys in rowData and colData.*

Description

Retrieve the unique identifier columns used for primary keys in rowData and colData.

Usage

```
## S4 method for signature 'LongTable'
idCols(object)
```

Arguments

object	LongTable

Value

character A character vector containing the unique rowIDs and colIDs in a LongTable object.

Examples

```
idCols(merckLongTable)
```

---

is.items *Get the types of all items in a list*

Description

Get the types of all items in a list

Usage

```
is.items(list, ..., FUN = is)
```

Arguments

list	A list to get the types from
...	pairlist Additional arguments to FUN
FUN	function or character Either a function, or the name of a function which returns a single logical value. The default function uses is, specify the desired type in .... You can also use other type checking functions such as is.character, is.numeric, or is.data.frame.

Value

logical A vector indicating if the list item is the specified type.

Examples

```
list <- list(c(1,2,3), c('a','b','c'))
is.items(list, 'character')
```

lapply,MultiAssayExperiment-method *lapply lapply method for* MultiAssayExperiment

Description

lapply lapply method for MultiAssayExperiment

Usage

```
## S4 method for signature 'MultiAssayExperiment'
lapply(X, FUN, ...)
```

Arguments

X	A MultiAssayExperiment object.
FUN	A function to be applied to each SummarizedExperiment in a in X.
...	Fall through parameters to FUN

Value

A MultiAssayExperiment object, modifed such that experiments(X) <-endoapply(experiments(X),FUN,...).s

list_or_LongTable-class *A class union to allow multiple types in a CoreSet slot*

Description

A class union to allow multiple types in a CoreSet slot

LongTable *LongTable constructor method*

Description

Builds a LongTable object from rectangular objects. The rowData argument should contain row level metadata, while the colData argument should contain column level metadata, for the experimental assays in the assays list. The rowIDs and colIDs lists are used to configure the internal keys mapping rows or columns to rows in the assays. Each list should contain at minimum one character vector, specifying which columns in rowData or colData are required to uniquely identify each row. An optional second character vector can be included, specifying any metadata columns for either dimension. These should contain information about each row but NOT be required to uniquely identify a row in the colData or rowData objects. Additional metadata can be attached to a LongTable by passing a list to the metadata argument.

Usage

```
LongTable(
  rowData,
  rowIDs,
  colData,
  colIDs,
  assays,
  metadata = list(),
  keep.rownames = FALSE
)
```

Arguments

rowData	data.table, data.frame, matrix A table like object coercible to a data.table containing the a unique rowID column which is used to key assays, as well as additional row metadata to subset on.
rowIDs	character, integer A vector specifying the names or integer indexes of the row data identifier columns. These columns will be pasted together to make up the row.names of the LongTable object.
colData	data.table, data.frame, matrix A table like object coercible to a data.table containing the a unique colID column which is used to key assays, as well as additional column metadata to subset on.
colIDs	character, integer A vector specifying the names or integer indexes of the col data identifier columns. These columns will be pasted together to make up the col.names of the LongTable object.
assays	A list containing one or more objects coercible to a data.table, and keyed by rowID and colID corresponding to the rowID and colID columns in colData and rowData.
metadata	A list of metadata associated with the LongTable object being constructed
keep.rownames	logical or character Logical: whether rownames should be added as a column if coercing to a data.table, default is FALSE. If TRUE, rownames are added to the column 'rn'. Character: specify a custom column name to store the rownames in.

Value

A LongTable object containing the data for a treatment response experiment and configured according to the rowIDs and colIDs arguments.

LongTableDataMapper	*Constructor for the* LongTableDataMapper *class, which maps from one or more raw experimental data files to the slots of a* LongTable *object.*

Description

Constructor for the LongTableDataMapper class, which maps from one or more raw experimental data files to the slots of a LongTable object.

Usage

```
LongTableDataMapper(
  rawdata,
  rowDataMap = list(),
  colDataMap = list(),
  assayMap = list(),
  metadataMap = list()
)
```

Arguments

rawdata	A data.frame of raw data from a treatment response experiment. This will be coerced to a data.table internally. We recommend using joins to aggregate your raw data if it is not present in a single file.
rowDataMap	A list-like object containing two character vectors. The first is column names in rawdata needed to uniquely identify each row, the second is additional columns which map to rows, but are not required to uniquely identify them. Rows should be drugs.
colDataMap	A list-like object containing two character vectors. The first is column names in rawdata needed to uniquely identify each column, the second is additional columns which map to rows, but are not required to uniquely identify them. Columns should be samples.
assayMap	A list-like where each item is a character vector of rawdata column names to assign to an assay, where the name of that assay is the name of the list item. If names are omitted, assays will be numbered by their index in the list
metadataMap	A list-like where each item is a character vector of rawdata column names to assign to the @metadata of the LongTable, where the name of that assay is the name of the list item. If names are omitted, assays will be numbered by their index in the list.

Details

The guessMapping method can be used to test hypotheses about the cardinality of one or more sets of identifier columns. This is helpful to determine the id columns for rowDataMap and colDataMap, as well as identify columns mapping to assays or metadata.

To attach metadata not associated with rawdata, please use the metadata assignment method on your LongTableDataMapper. This metadata will be merge with any metadata from metadataMap and added to the LongTable which this object ultimately constructs.

Value

A LongTable object, with columns mapped to it's slots according to the various maps in the LongTableDataMapper object.

See Also

guessMapping

LongTableDataMapper-accessors

Accessing and modifying data in a LongTableDataMapper *object.*

Description

Documentation for the various setters and getters which allow manipulation of data in the slots of a LongTableDataMapper object.

Usage

```
## S4 method for signature 'LongTableDataMapper'
rowDataMap(object)

## S4 replacement method for signature 'LongTableDataMapper,list_or_List'
rowDataMap(object) <- value

## S4 method for signature 'LongTableDataMapper'
colDataMap(object)

## S4 replacement method for signature 'LongTableDataMapper,list_or_List'
colDataMap(object) <- value

## S4 method for signature 'LongTableDataMapper'
assayMap(object)

## S4 replacement method for signature 'LongTableDataMapper,list_or_List'
assayMap(object) <- value

## S4 method for signature 'LongTableDataMapper'
metadataMap(object)

## S4 replacement method for signature 'LongTableDataMapper,list_or_List'
metadataMap(object) <- value
```

Arguments

object	A LongTableDataMapper object to get or set data from.
value	See details.

Details

rawdata: Get the raw data slot from a `LongTableDataMapper` object. Returns a list-like containing one or more raw data inputs to the `LongTableDataMapper` object.

rowDataMap: `list` of two `character` vectors, the first are the columns required to uniquely identify each row of a `LongTableDataMapper` and the second any additional row-level metadata. If the character vectors have names, the resulting columns are automatically renamed to the item name of the specified column.

rowDataMap<-: Update the @rowDataMap slot of a `LongTableDataMapper` object, returning an invisible NULL. Arguments:

- value: A `list` or `List` where the first item is the names of the identifier columns – columns needed to uniquely identify each row in rowData – and the second item is the metadata associated with those the identifier columns, but not required to uniquely identify rows in the object rowData.

colDataMap: `list` of two `character` vectors, the first are the columns required to uniquely identify each row of a `LongTableDataMapper` and the second any additional col-level metadata. If the character vectors have names, the resulting columns are automatically renamed to the item name of the specified column.

colDataMap<-: Update the @colDataMap slot of a `LongTableDataMapper` object, returning an invisible NULL. Arguments:

- value: A `list` or `List` where the first item is the names of the identifier columns – columns needed to uniquely identify each row in colData – and the second item is the metadata associated with those the identifier columns, but not required to uniquely identify rows in the object rowData.

assayMap: A `list` of character vectors. The name of each list item will be the assay in a `LongTableDataMapper` object that the columns in the `character` vector will be assigned to. Column renaming occurs automatically when the character vectors have names (from the value to the name).

assayMap<-: Updates the @assaMap slot of a `LongTableDataMapper` object, returning an invisible NULL. Arguments:

- value: A `list` of character vectors, where the name of each list item is the name of an assay and the values of each character vector specify the columns mapping to the assay in the `S4` object the `LongTableDataMapper` constructs.

metadataMap: A `list` of `character` vectors. Each item is an element of the constructed objects @metadata slot.

metadataMap<-: Updates `LongTableDataMapper` object in-place, then returns an `invisible(NULL)`. Arguments:

- value: A `list` of `character` vectors. The name of each list item is the name of the item in the @metadata slot of the `LongTableDataMapper` object created when `metaConstruct` is called on the `DataMapper`, and a character vector specifies the columns of @rawdata to assign to each item.

Value

Accessors: See details

Setters: An update `LongTableDataMapper` object, returned invisibly.

See Also

Other DataMapper-accessors: `DataMapper-accessors`

Examples

```
rowDataMap(exampleDataMapper)

rowDataMap(exampleDataMapper) <- list(c('drug_id'), c())

colDataMap(exampleDataMapper)

colDataMap(exampleDataMapper) <- list(c('cell_id'), c())

assayMap(exampleDataMapper)

assayMap(exampleDataMapper) <- list(sensitivity=c(viability1='viability'))

metadataMap(exampleDataMapper)

metadataMap(exampleDataMapper) <- list(object_metadata=c('metadata'))
```

`LongTableDataMapper-class` *A Class for Mapping Between Raw Data and an* `LongTable` *Object*

Description

A Class for Mapping Between Raw Data and an `LongTable` Object

Usage

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

Arguments

`object`	A `LongTableDataMapper` to display in the console.

Value

`invisible` Prints to console.

Functions

- `show,LongTableDataMapper-method`: Show method for LongTableDataMapper. Determines how the object is displayed in the console.

Slots

`rowDataMap` See Slots section.

`colDataMap` See Slots section.

`assayMap` See Slots section.

`metadataMap` See Slots section.

Slots

- rowDataMap: A list-like object containing two `character` vectors. The first is column names in `rawdata` needed to uniquely identify each row, the second is additional columns which map to rows, but are not required to uniquely identify them. Rows should be drugs.
- colDataMap: A list-like object containing two `character` vectors. The first is column names in `rawdata` needed to uniquely identify each column, the second is additional columns which map to rows, but are not required to uniquely identify them. Columns should be samples.
- assayMap: A list-like where each item is a `character` vector of `rawdata` column names to assign to an assay, where the name of that assay is the name of the list item. If names are omitted, assays will be numbered by their index in the list.
- metadataMap: A list-like where each item is a `character` vector of `rawdata` column names to assign to the @metadata of the `LongTable`, where the name of that assay is the name of the list item. If names are omitted, assays will be numbered by their index in the list.
- rawdata: A list-like object containing one or more pieces of raw data that will be processed and mapped to the slots of an `S4` object.
- metadata: A `List` of object level metadata.

Examples

```
show(exampleDataMapper)
```

---

mcc *Compute a Mathews Correlation Coefficient*

Description

The function computes a Matthews correlation coefficient for two factors provided to the function. It assumes each factor is a factor of class labels, and the enteries are paired in order of the vectors.

Usage

```
mcc(x, y, nperm = 1000, nthread = 1, ...)
```

Arguments

x, y	factor of the same length with the same number of levels
nperm	numeric number of permutations for significance estimation. If 0, no permutation testing is done
nthread	numeric can parallelize permutation texting using BiocParallels bplapply
...	list Additional arguments

Details

Please note: we recommend you call set.seed() before using this function to ensure the reproducibility of your results. Write down the seed number or save it in a script if you intend to use the results in a publication.

Value

A list with the MCC as the $estimate, and p value as $p.value

Examples

```
x <- factor(c(1,2,1,2,3,1))
y <- factor(c(2,1,1,1,2,2))
mcc(x,y)
```

---

merckLongTable *Merck Drug Combination Data LongTable*

Description

This is a LongTable object created from some drug combination data provided to our lab by Merck.

Usage

```
data(merckLongTable)
```

Format

LongTable object

References

TODO:: Include a reference

metaConstruct *Generic for preprocessing complex data before being used in the constructor of an* S4 *container object.*

Description

This method is intended to abstract away complex constructor arguments and data preprocessing steps needed to transform raw data, such as that produced in a treatment-response or next-gen sequencing experiment, and automate building of the appropriate S4 container object. This is is intended to allow mapping between different experimental designs, in the form of an S4 configuration object, and various S4 class containers in the Bioconductor community and beyond.

Usage

```
metaConstruct(mapper, ...)

## S4 method for signature 'LongTableDataMapper'
metaConstruct(mapper)
```

Arguments

mapper	An LongTableDataMapper object abstracting arguments to an the LongTable constructor.
...	Allow new arguments to be defined for this generic.

Value

An S4 object for which the class corresponds to the type of the build configuration object passed to this method.

A LongTable object, as specified in the mapper.

Examples

```
data(exampleDataMapper)
rowDataMap(exampleDataMapper) <- list(c('drug_id'), c())
colDataMap(exampleDataMapper) <- list(c('cell_id'), c())
assayMap(exampleDataMapper) <- list(sensitivity=c('viability'))
metadataMap(exampleDataMapper) <- list(study_metadata=c('metadata'))
longTable <- metaConstruct(exampleDataMapper)
longTable
```

`metadata,LongTable-method`
Getter method for the metadata slot of a `LongTable` *object*

Description

Getter method for the metadata slot of a `LongTable` object

Usage

```
## S4 method for signature 'LongTable'
metadata(x)
```

Arguments

`x`	The `LongTable` object from which to retrieve the metadata list.

Value

`list` The contents of the `metadata` slot of the `LongTable` object.

`metadata<-,LongTable-method`
Setter method for the metadata slot of a `LongTable` *object*

Description

Setter method for the metadata slot of a `LongTable` object

Usage

```
## S4 replacement method for signature 'LongTable'
metadata(x) <- value
```

Arguments

`x`	`LongTable` The LongTable to update
`value`	`list` A list of new metadata associated with a `LongTable` object.

Value

`LongTable` A copy of the `LongTable` object with the `value` in the metadata slot.

reindex *Generic method for resetting indexing in an S4 object*

Description

This method allows integer indexes used to maintain referential integrity internal to an S4 object to be reset. This is useful particularly after subsetting an object, as certain indexes may no longer be present in the object data. Reindexing removes gaps integer indexes and ensures that the smallest contiguous integer values are used in an objects indexes.

Usage

```
reindex(object, ...)
```

Arguments

object	S4 An object to redo indexing for
...	pairlist Allow definition of new parameters to this generic.

Value

Depends on the implemented method

Examples

```
print("Generics shouldn't need examples?")
```

reindex,LongTable-method *Redo indexing for a LongTable object to remove any gaps in integer indexes*

Description

After subsetting a LongTable, it is possible that values of rowKey or colKey could no longer be present in the object. As a result there the indexes will no longer be contiguous integers. This method will calcualte a new set of rowKey and colKey values such that integer indexes are the smallest set of contiguous integers possible for the data.

Usage

```
## S4 method for signature 'LongTable'
reindex(object)
```

Arguments

object	The LongTable object to recalcualte indexes (rowKey and colKey values) for.

Value

A copy of the LongTable with all keys as the smallest set of contiguous integers possible given the current data.

---

rowIDs *Generic to access the row identifiers from*

Description

Generic to access the row identifiers from

Usage

```
rowIDs(object, ...)
```

Arguments

object	S4 An object to get row id columns from.
...	Allow new arguments to this generic.

Value

Depends on the implemented method.

Examples

```
print("Generics shouldn't need examples?")
```

---

rowMeta *Generic to access the row identifiers from*

Description

Generic to access the row identifiers from

Usage

```
rowMeta(object, ...)
```

Arguments

object	S4 An object to get row metadata columns from.
...	Allow new arguments to this generic.

Value

Depends on the implemented method.

Examples

```
print("Generics shouldn't need examples?")
```

---

sensitivityInfo *Generic function to get the annotations for a treatment response experiment from an S4 class*

Description

Generic function to get the annotations for a treatment response experiment from an S4 class

Usage

```
sensitivityInfo(object, ...)
```

Arguments

object	An S4 object to get treatment response experiment annotations from.
...	Allow new arguments to be defined for this generic.

Value

Depends on the implemented method

Examples

```
print("Generics shouldn't need examples?")
```

sensitivityInfo<- *sensitivityInfo<- Generic Method*

Description

Generic function to get the annotations for a treatment response experiment from an S4 class.

Usage

```
sensitivityInfo(object, ...) <- value
```

Arguments

object	An S4 object to set treatment response experiment annotations for.
...	Allow new arguments to be defined for this generic.
value	The new treatment response experiment annotations.

Value

Depends on the implemented method

Examples

```
print("Generics shouldn't need examples?")
```

sensitivityMeasures *sensitivityMeasures Generic*

Description

Get the names of the sensitivity summary metrics available in an S4 object.

Usage

```
sensitivityMeasures(object, ...)
```

Arguments

object	An S4 object to retrieve the names of sensitivty summary measurements for.
...	Fallthrough arguements for defining new methods

Value

Depends on the implemented method

Examples

```
sensitivityMeasures(clevelandSmall_cSet)
```

---

sensitivityMeasures<- *sensitivityMeasures<- Generic*

Description

Set the names of the sensitivity summary metrics available in an S4 object.

Usage

```
sensitivityMeasures(object, ...) <- value
```

Arguments

object	An S4 object to update.
...	Allow new methods to be defined for this generic.
value	A set of names for sensitivity measures to use to update the object with.

Value

Depends on the implemented method

Examples

```
print("Generics shouldn't need examples?")
```

---

sensitivityProfiles *sensitivityProfiles Generic*

Description

A generic for sensitivityProfiles getter method

Usage

```
sensitivityProfiles(object, ...)
```

Arguments

object	The S4 object to retrieve sensitivity profile summaries from.
...	pairlist Allow defining new arguments for this generic.

Value

Depends on the implemented method

Examples

```
print("Generics shouldn't need examples?")
```

---

sensitivityProfiles<- *sensitivityProfiles<- Generic*

Description

A generic for the sensitivityProfiles replacement method

Usage

```
sensitivityProfiles(object, ...) <- value
```

Arguments

object	An S4 object to update the sensitivity profile summaries for.
...	Fallthrough arguments for defining new methods
value	An object with the new sensitivity profiles. If a matrix object is passed in, converted to data.frame before assignment

Value

Updated CoreSet

---

sensitivityRaw *sensitivityRaw Generic Method*

Description

Generic function to get the raw data array for a treatment response experiment from an S4 class.

Usage

```
sensitivityRaw(object, ...)
```

Arguments

object	An S4 object to extract the raw sensitivity experiment data from.
...	pairlist Allow new parameters to be defined for this generic.

Value

Depends on the implemented method

Examples

```
print("Generics shouldn't need examples?")
```

`sensitivityRaw<-` *sensitivityRaw<- Generic*

Description

Generic function to set the raw data array for a treatment response experiment in an S4 class.

Usage

```
sensitivityRaw(object, ...) <- value
```

Arguments

`object`	An `S4` object to extract the raw sensitivity data from.
`...`	`pairlist` Allow new parameters to be defined for this generic.
`value`	An object containing dose and viability metrics to update the object with.

Value

Depends on the implemented method

`sensitivitySlotToLongTable` *sensitivitySlotToLongTable Generic*

Description

Convert the sensitivity slot in an object inheriting from a CoreSet from a list to a LongTable.

Usage

```
sensitivitySlotToLongTable(object, ...)
```

Arguments

`object`	`CoreSet` Object inheriting from CoreSet.
`...`	Allow new arguments to be defined on this generic.

Value

A LongTable object containing the data in the sensitivity slot.

Examples

```
print("Generics shouldn't need examples?")
```

---

show,CoreSet-method *Show a CoreSet*

Description

Show a CoreSet

Usage

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

Arguments

object	CoreSet

Value

Prints the CoreSet object to the output stream, and returns invisible NULL.

Examples

```
show(clevelandSmall_cSet)
```

---

show,LongTable-method *Show method for the LongTable class*

Description

Show method for the LongTable class

Usage

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

Arguments

object	A LongTable object to print the results for.

Value

invisible Prints to console.

Examples

```
show(merckLongTable)
```

---

showSigAnnot — *Get the annotations for a* Signature *class object, as returned by* drugSensitivitysig *or* radSensitivtySig *functions available in* PharmacoGx *and* RadioGx*, respectively.*

Description

Get the annotations for a Signature class object, as returned by drugSensitivitysig or radSensitivtySig functions available in PharmacoGx and RadioGx, respectively.

Usage

```
showSigAnnot(object, ...)
```

Arguments

object	A Signature class object
...	Allow definition of new arguments to this generic

Value

NULL Prints the signature annotations to console

Examples

```
print("Generics shouldn't need examples?")
```

subset,LongTable-method

Subset method for a LongTable object.

Description

Allows use of the colData and rowData `data.table` objects to query based on rowID and colID, which is then used to subset all value data.tables stored in the dataList slot. This function is endomorphic, it always returns a LongTable object.

Usage

```
## S4 method for signature 'LongTable'
subset(x, i, j, assays, reindex = TRUE)
```

Arguments

`x`	`LongTable` The object to subset.
`i`	`character`, `numeric`, `logical` or `expression` Character: pass in a character vector of drug names, which will subset the object on all row id columns matching the vector. Numeric or Logical: these select based on the rowKey from the `rowData` method for the `LongTable`. Call: Accepts valid query statements to the `data.table` i parameter, this can be used to make complex queries using the `data.table` API for the `rowData` data.table.
`j`	`character`, `numeric`, `logical` or `expression` Character: pass in a character vector of drug names, which will subset the object on all drug id columns matching the vector. Numeric or Logical: these select based on the rowID from the `rowData` method for the `LongTable`. Call: Accepts valid query statements to the `data.table` i parameter, this can be used to make complex queries using the `data.table` API for the `colData` data.table.
`assays`	`character`, `numeric` or `logical` Optional list of assay names to subset. Can be used to subset the assays list further, returning only the selected items in the new LongTable.
`reindex`	`logical` Should the col/rowKeys be remapped after subsetting. defaults to TRUE. For chained subsetting you may be able to get performance gains by setting to FALSE and calling reindex() manually after subsetting is finished.

Value

`LongTable` A new `LongTable` object subset based on the specified parameters.

Examples

```
# Character
subset(merckLongTable, 'ABT-888', 'CAOV3')
# Numeric
```

```
subset(merckLongTable, 1, c(1, 2))
# Logical
subset(merckLongTable, , colData(merckLongTable)$cellid == 'A2058')
# Call
subset(merckLongTable, drug1id == 'Dasatinib' & drug2id != '5-FU',
    cellid == 'A2058')
```

---

summarizeMolecularProfiles

Summarize molecular profile data such that there is a single entry for each cell line/treatment combination

---

Description

Summarize molecular profile data such that there is a single entry for each cell line/treatment combination

Usage

```
summarizeMolecularProfiles(object, ...)
```

Arguments

object	An S4 object to summarize the molecular profiles for.
...	Allow definition of new arguments to this generic

Value

Depends on the implemented method

Examples

```
print("Generics shouldn't need examples?")
```

---

summarizeSensitivityProfiles

Summarize across replicates for a sensitivity dose-response experiment

---

Description

Summarize across replicates for a sensitivity dose-response experiment

Usage

```
summarizeSensitivityProfiles(object, ...)
```

Arguments

object	An S4 object to summarize sensitivity profiles for.
...	Allow definition of new arguments to this generic

Value

Depends on the implemented method

Examples

```
print("Generics shouldn't need examples?")
```

[,LongTable,ANY,ANY,ANY-method
[LongTable Method

Description

Single bracket subsetting for a LongTable object. See subset for more details.

Usage

```
## S4 method for signature 'LongTable,ANY,ANY,ANY'
x[i, j, assays, ..., drop = FALSE]
```

Arguments

x	LongTable The object to subset.
i	character, numeric, logical or call Character: pass in a character vector of drug names, which will subset the object on all row id columns matching the vector. This parameter also supports valid R regex query strings which will match on the colnames of x. For convenience, * is converted to .* automatically. Colon can be to denote a specific part of the colnames string to query. Numeric or Logical: these select based on the rowKey from the rowData method for the LongTable. Call: Accepts valid query statements to the data.table i parameter as a call object. We have provided the function .() to conveniently convert raw R statements into a call for use in this function.
j	character, numeric, logical or call Character: pass in a character vector of drug names, which will subset the object on all drug id columns matching the vector. This parameter also supports regex queries. Colon can be to denote a specific part of the colnames string to query. Numeric or Logical: these select

	based on the rowID from the `rowData` method for the `LongTable`. Call: Accepts valid query statements to the `data.table` i parameter as a call object. We have provided the function .() to conveniently convert raw R statements into a call for use in this function.
`assays`	`character` Names of assays which should be kept in the `LongTable` after subsetting.
`...`	Included to ensure drop can only be set by name.
`drop`	`logical` Included for compatibility with the '[' primitive, it defaults to FALSE and changing it does nothing.

Details

This function is endomorphic, it always returns a LongTable object.

Value

A `LongTable` containing only the data specified in the function parameters.

Examples

```
# Character
merckLongTable['ABT-888', 'CAOV3']
# Numeric
merckLongTable[1, c(1, 2)]
# Logical
merckLongTable[, colData(merckLongTable)$cellid == 'A2058']
# Call
merckLongTable[
     .(drug1id == 'Dasatinib' & drug2id != '5-FU'),
     .(cellid == 'A2058'),
 ]
```

`[[<-,LongTable,ANY,ANY-method`

[[<- *Method for LongTable Class*

Description

Just a wrapper around assay<- for convenience. See ?'assay<-,LongTable,character-method'.

Usage

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

Arguments

x	A LongTable to update.
i	The name of the assay to update, must be in assayNames(object).
value	A data.frame

Value

A LongTable object with the assay i updated using value.

Examples

```
merckLongTable[['sensitivity']] <- merckLongTable[['sensitivity']]
```

$,LongTable-method *Select an assay from a LongTable object*

Description

Select an assay from a LongTable object

Usage

```
## S4 method for signature 'LongTable'
x$name
```

Arguments

x	A LongTable object to retrieve an assay from
name	character The name of the assay to get.

Value

data.frame The assay object.

Examples

```
merckLongTable$sensitivity
```

`$<-,LongTable-method` *Update an assay from a LongTable object*

Description

Update an assay from a LongTable object

Usage

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

Arguments

`x`	A `LongTable` to update an assay for.
`name`	`character(1)` The name of the assay to update
`value`	A `data.frame` or `data.table` to update the assay with.

Value

Updates the assay `name` in `x` with `value`, returning an invsible NULL.

Examples

```
merckLongTable$sensitivity <- merckLongTable$sensitivity
```

Index