Package 'CoreGx'

February 28, 2021

Type Package

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

Version 1.3.0

Date 2020-10-23

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 (>= 4.0)

Imports Biobase, S4Vectors, SummarizedExperiment, piano, BiocParallel, BiocGenerics, methods, stats, utils, graphics, grDevices, lsa, data.table, crayon

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

License GPL-3

RoxygenNote 7.1.1

Collate 'adaptiveMatthewCor.R' 'allGenerics.R' 'callingWaterfall.R' 'class-LongTable.R' 'class-CoreSet.R' 'connectivityScore.R'

'cosinePerm.R' 'datasets.R' 'globals.R' 'gwc.R' 'matthewCor.R'
'methods-$.R' 'methods-[.R' 'methods-[[.R'
'methods-annotation.R' 'methods-assay.R' 'methods-assayNames.R'
'methods-assays.R' 'methods-buildLongTable.R'
'methods-cellInfo.R' 'methods-cellNames.R' 'methods-coerce.R'
'methods-colData.R' 'methods-curation.R'
'methods-datasetType.R' 'methods-dateCreated.R' 'methods-dim.R'
'methods-dimnames.R' 'methods-drugSensitivitySig.R'
'methods-fNames-methods.R' 'methods-featureInfo.R'
'methods-getIntern.R' 'methods-intersect.R'
'methods-mDataNames.R' 'methods-metadata.R'
'methods-molecularProfiles.R' 'methods-molecularProfilesSlot.R'
'methods-name.R' 'methods-pertNumber.R' 'methods-phenoInfo.R'
'methods-reindex.R' 'methods-rowData.R' 'methods-sensNumber.R'
'methods-sensitivityInfo.R' 'methods-sensitivityMeasures.R'
'methods-sensitivityProfiles.R'
'methods-sensitivityRaw-methods.R'
'methods-sensitivitySlot-methods.R' 'methods-subset.R'
'methods-subsetTo.R' 'utilities.R'

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

git_branch master

git_last_commit efa887e

git_last_commit_date 2020-10-27

Date/Publication 2021-02-27

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

...	['parilist'] 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)
```

---

annotation<-,CoreSet,list-method
annotation<- Slot Setter

Description

annotation<- Slot Setter

Usage

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

Arguments

object	A RadioSet
value	A list of annotations to add to the annotatiosn slot of an rSet

Value

A copy of the CoreSet with the updated annotation slot

Functions

- annotation<-,CoreSet,list-method: Update the annotation slot of a tSet

Examples

```
annotation(clevelandSmall_cSet) <- annotation(clevelandSmall_cSet)
```

---

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.table‘s. 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))
colDataCols <- list(colIDs(merckLongTable), colMeta(merckLongTable))
assayCols <- assayCols(merckLongTable)
longTable <- buildLongTable(from=assayList, rowDataCols, colDataCols, assayCols)
```

`cellInfo` *cellInfo Getter*

Description

Get cell line information from a PharmacoSet object

Usage

```
cellInfo(object, ...)
```

Arguments

`object`	The `CoreSet` to retrieve cell info from
`...`	`list` Fall through arguments to allow generic to be defined with different parameters

Value

a `data.frame` with the cell annotations

Examples

```
cellInf <- cellInfo(clevelandSmall_cSet)
```

cellInfo<- *cellInfo<- Generic*

Description

Generic for cellInfo replace method

Usage

```
cellInfo(object) <- value
```

Arguments

object	The CoreSet to replace cell info in
value	A data.frame with the new cell annotations

Value

Updated CoreSet

Examples

```
cellInfo(clevelandSmall_cSet) <- cellInfo(clevelandSmall_cSet)
```

cellNames *cellNames Generic*

Description

A generic for the cellNames method

Usage

```
cellNames(object, ...)
```

Arguments

object	The CoreSet to return cell names from
...	Fallthrough arguements for defining new methods

Value

A vector of the cell names used in the CoreSet

Examples

```
cellNames(clevelandSmall_cSet)
```

cellNames<- *cellNames<- Generic*

Description

A generic for the cellNames replacement method

Usage

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

Arguments

object	The CoreSet to update
...	Fallthrough arguments for defining new methods
value	A character vector of the new cell names

Value

Updated CoreSet

Examples

```
cellNames(clevelandSmall_cSet) <- cellNames(clevelandSmall_cSet)
```

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-class *A Superclass to Contain Data for Genetic Profiling and Viability Screens of Cancer Cell Lines*

Description

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.

Generic for sensitivityInfo method

A generic for the sensitivityInfo replacement method

Usage

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

## 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'
datasetType(object) <- value

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

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

```
dateCreated(object) <- value

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Arguments

`object`	An [‘CoreSet‘] to extract the raw sensitivity data from.
`value`	A 3D [‘array‘] containing the raw dose and viability measurements to update the object with.
`mDataType`	`character` The type of molecular data

assay character Name of the desired assay; if excluded defaults to first assay in the SummarizedExperiment for the given mDataType. Use assayNames(molecularProfiles(object,c to check which assays are available for a given molecular datatype.

Value

An object of the CoreSet class

A list of named annotaiton

A list of unique cell and tissue identifiers to check validity of a cSet

A copy of the RadioSet with the updated curation slot

a data.frame with the experiment info if dimension is excluded, otherwise a 'data.table' with annotations for the cells or drugs dimension of the LongTable.

Updated CoreSet

A ['character'] vector of all the available sensitivity measures.

A update CoreSet object with the new sensitivity measures

a ['data.frame'] with sensitivity profile summaries for CoreSet

Updated CoreSet

A ['array'] containing the raw sensitivity data as experiment by dose level by metric.

A 3D ['array'] containing the raw sensitivity data as experiment by dose level by metric.

Methods (by generic)

- annotation: Retrieve the annotations slot form an rSet
- cellInfo: Returns the annotations for all the cell lines tested on in the CoreSet
- cellInfo<-: Update the cell line annotations
- cellNames: Return the cell names used in the dataset
- cellNames<-: Update the cell names used in the dataset
- curation: Retrieve the curation slot form a cSet
- curation<-: Update the curation slot of a cSet
- datasetType: Update the dataset type of an rSet and return a copy of the updated object
- datasetType<-: Update the dataset type of an rSet and return a copy of the updated object
- dateCreated: Return the date the CoreSet was created
- dateCreated<-: Update the date a cSet was created
- fNames: Return the feature names used in the dataset
- fNames<-: Update the feature names used in a molecular profile
- featureInfo: Return the feature info for the given molecular data
- featureInfo<-: Replace the gene info for the molecular data
- featureInfo<-: Replace the gene info for the molecular data
- mDataNames: Return the molecular data types available in a cSet object
- mDataNames<-: Return the molecular data types available in a cSet object
- molecularProfiles: Return the given type of molecular data from the CoreSet
- molecularProfiles<-: Update the given type of molecular data from the CoreSet
- molecularProfiles<-: Update the given type of molecular data from the CoreSet

- molecularProfilesSlot: Return a list containing all molecularProfiles in the cSet
- molecularProfilesSlot<-: Update the contents of the molecularProfiles slot in a CoreSet and returns an update copy
- name: Return the name of the CoreSet
- name<-: Return the name of the CoreSet
- pertNumber: Return the summary of available perturbation experiments
- pertNumber<-: Update the summary of available perturbation experiments
- phenoInfo: Return the experiment info from the given type of molecular data in CoreSet
- phenoInfo<-: Update the given type of molecular data experiment info in the CoreSet
- phenoInfo<-: Update the given type of molecular data experiment info in the CoreSet
- sensNumber: Return the summary of available sensitivity experiments
- sensNumber<-: Update the summary of available sensitivity experiments
- sensitivityInfo: Return the drug dose sensitivity experiment info
- sensitivityInfo<-: Update the sensitivity experiment info
- sensitivityMeasures: Returns the available sensitivity profile summaries, for example, whether there are IC50 values available
- sensitivityMeasures<-: Updates the sensitivity measures in a 'CoreSet' object and returns the updated object
- sensitivityProfiles: Return the sensitivity profile summaries from the sensitivity slot.
- sensitivityProfiles<-: Update the sensitivity profile summaries the sensitivity slot.
- sensitivityProfiles<-: Update the phenotypic data for the drug dose sensitivity
- sensitivityRaw: Get the raw dose and vaibility data from a CoreSet object.
- sensitivityRaw<-: Set the raw dose and viability data for a cSet and return and updated copty
- sensitivitySlot: Retrieve the contents of the sensitivity slot
- sensitivitySlot<-: Set the raw dose and viability data for a cSet and return and updated copty

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 containing SummarizedExperiments type object for holding data for RNA, DNA, SNP and Copy Number Variation measurements respectively, with associated rowData and colData containing the row and column metadata

cell A data.frame containg the annotations for all the cell lines profiled in the data set, across all data types

sensitivity A list 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 A list containting $n, a data.frame summarizing the available perturbation data,

curation A list containing mappings for cell, tissue names used in the data set to universal identifiers used between different CoreSet objects

datasetType A character string of 'sensitivity', 'perturbation', or both detailing what type of data can be found in the CoreSet, for proper processing of the data

Examples

```
annotation(clevelandSmall_cSet)

data(clevelandSmall_cSet)
curation(clevelandSmall_cSet)

data(clevelandSmall_cSet)
curation(clevelandSmall_cSet) <- curation(clevelandSmall_cSet)

data(clevelandSmall_cSet)
sensitivityInfo(clevelandSmall_cSet)

sensitivityInfo(clevelandSmall_cSet) <- sensitivityInfo(clevelandSmall_cSet)

sensitivityMeasures(clevelandSmall_cSet)

data(clevelandSmall_cSet)
sensitivityMeasures(clevelandSmall_cSet) <- sensitivityMeasures(clevelandSmall_cSet)

sensitivityProfiles(clevelandSmall_cSet)

sensitivityProfiles(clevelandSmall_cSet) <- sensitivityProfiles(clevelandSmall_cSet)

data(clevelandSmall_cSet)
sensRaw <- sensitivityRaw(clevelandSmall_cSet)
head(sensRaw)

data(clevelandSmall_cSet)
sensitivityRaw(clevelandSmall_cSet) <- sensitivityRaw(clevelandSmall_cSet)
```

`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)
```

---

curation *curation Slot Getter*

Description

curation Slot Getter

Usage

```
curation(object, ...)
```

Arguments

object	A object
...	A list to allow definition of new parameters on this generic

Value

A list of unique cell and tissue identifiers to check validity of an rSet

Examples

```
data(clevelandSmall_cSet)
curation(clevelandSmall_cSet)
```

curation<- *curation<- Slot Setter*

Description

#' @examples data(clevelandSmall_cSet) curation(clevelandSmall_cSet) <- curation(clevelandSmall_cSet)

Usage

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

Arguments

object	A object
...	A list to allow definition of new parameters on this generic
value	A list of curations for the cell and tissues types in the rSet object

Value

A copy of the RadioSet with the updated curation slot

---

datasetType *datasetType Generic*

Description

A generic for retrieving the dataset type of an rSet object

Usage

```
datasetType(object, ...)
```

Arguments

object	A CoreSet from which to retrieve the dataset type
...	A list containing fall through arguments; this allows addition of new parameters to methods for this generic

Value

A character vector containing the dataset tpye

Examples

```
data(clevelandSmall_cSet)
datasetType(clevelandSmall_cSet)
```

datasetType<- *datasetType<- Replacement Generic*

Description

A generic for updating the dataset type of a RadioSet object

Usage

```
datasetType(object) <- value
```

Arguments

object	A RadioSet from which to retrieve the dataset type
value	A character vector containing the dataset type

Value

A character vector containing the dataset tpye

Examples

```
data(clevelandSmall_cSet)
datasetType(clevelandSmall_cSet)
```

dateCreated *dateCreated Generic*

Description

A generic for the dateCreated method

Usage

```
dateCreated(object, ...)
```

Arguments

object	A CoreSet
...	Fallthrough arguements for defining new methods

Value

The date the CoreSet was created

Examples

```
dateCreated(clevelandSmall_cSet)
```

dateCreated<- *dateCreated<- Generic*

Description

A generic for the dateCreated method

Usage

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

Arguments

object	A CoreSet
...	Fallthrough arguements for defining new methods
value	A datetime object to update the cSet with

Value

The date the CoreSet was created

Examples

```
dateCreated(clevelandSmall_cSet) <- date()
```

featureInfo *featureInfo Generic*

Description

Generic for featureInfo method

Usage

```
featureInfo(object, mDataType, ...)
```

Arguments

object	The CoreSet to retrieve feature annotations from
mDataType	the type of molecular data
...	Fallthrough arguements for defining new methods

Value

a data.frame with the feature annotations

Examples

```
featureInfo(clevelandSmall_cSet, "rna")
```

featureInfo<- *featureInfo<- Generic*

Description

Generic for featureInfo replace method

Usage

```
featureInfo(object, mDataType) <- value
```

Arguments

object	The CoreSet to replace gene annotations in
mDataType	The type of molecular data to be updated
value	A data.frame with the new feature annotations

Value

Updated CoreSet

Examples

```
featureInfo(clevelandSmall_cSet, "rna") <- featureInfo(clevelandSmall_cSet, "rna")
```

fNames *fNames Generic*

Description

A generic for the fNames method

Usage

```
fNames(object, mDataType, ...)
```

Arguments

object	The CoreSet
mDataType	The molecular data type to return feature names for
...	Fallthrough arguements for defining new methods

Value

A character vector of the feature names

Examples

```
fNames(clevelandSmall_cSet, "rna")
```

fNames<- *fNames<- Generic*

Description

A generic for the fNames replacement method

Usage

```
fNames(object, mDataType) <- value
```

Arguments

object	The CoreSet to update
mDataType	The molecular data type to update
value	A character vector of the new cell names

Value

Updated CoreSet

Examples

```
data(clevelandSmall_cSet)
fNames(clevelandSmall_cSet, "rna") <- fNames(clevelandSmall_cSet, "rna")
```

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?")
```

---

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')
```

---

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.

`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)
```

mDataNames *mDataNames Generic*

Description

A generic for the mDataNames method

Usage

```
mDataNames(object, ...)
```

Arguments

object	CoreSet object
...	Fallthrough arguements for defining new methods

Value

Vector of names of the molecular data types

Examples

```
mDataNames(clevelandSmall_cSet)
```

mDataNames<- *mDataNames<- Generic*

Description

A generic for the mDataNames method

Usage

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

Arguments

object	CoreSet object
...	Fallthrough arguements for defining new methods
value	A character vector with names to be assigned to each list item in the 'molecularProfiles' slot

Value

An updated copy of the CoreSet object

Examples

```
mDataNames(clevelandSmall_cSet) <- mDataNames(clevelandSmall_cSet)
```

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

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.

molecularProfiles *molecularProfiles Generic*

Description

Generic for molecularProfiles method

Usage

```
molecularProfiles(object, mDataType, assay, ...)
```

Arguments

object	The CoreSet to retrieve molecular profiles from
mDataType	character The type of molecular data
assay	character Name of the desired assay; if excluded defaults to first assay in the SummarizedExperiment for the given mDataType. Use assayNames(molecularProfiles(object,c to check which assays are available for a given molecular datatype.
...	Fallthrough arguements for defining new methods

Value

a matrix of data for the given mDataType and assay

Examples

```
data(clevelandSmall_cSet)
molecularProfiles(clevelandSmall_cSet, "rna")
```

molecularProfiles<- *molecularProfiles<- Generic*

Description

Generic for molecularProfiles replace method

Usage

```
molecularProfiles(object, mDataType, assay) <- value
```

Arguments

object	The CoreSet to replace molecular profiles in
mDataType	The type of molecular data to be updated
assay	character Name or index of the assay data to return
value	A matrix with the new profiles

Value

Updated CoreSet

Examples

```
data(clevelandSmall_cSet)
molecularProfiles(clevelandSmall_cSet, "rna") <- molecularProfiles(clevelandSmall_cSet, "rna")
```

molecularProfilesSlot *molecularProfilesSlot Generic*

Description

molecularProfilesSlot Generic

Usage

```
molecularProfilesSlot(object, ...)
```

Arguments

object	A CoreSet from which to return a list of all availble SummarizedExperiment objects
...	A list of additional parameters; included to allow adding arguments to methods on this generc

Value

A list containing the molecularProfiles from a cSet

Generic for molecularProfilesSlot

Examples

```
data(clevelandSmall_cSet)
molecularProfilesSlot(clevelandSmall_cSet)
```

`molecularProfilesSlot<-` *molecularProfilesSlot<-*

Description

Replace method for the molecular profiles slot of a cSet

Usage

```
molecularProfilesSlot(object) <- value
```

Arguments

`object`	A `CoreSet` object for which values will be replaced
`value`	A `list` containing molecular profiles as SummarizedExperiments

Value

A copy of the `CoreSet` with the molecularProfiles slot updated

Examples

```
data(clevelandSmall_cSet)
molecularProfilesSlot(clevelandSmall_cSet) <- molecularProfilesSlot(clevelandSmall_cSet)
```

`name` *name Generic*

Description

A generic for the name method

Usage

```
name(object, ...)
```

Arguments

`object`	A `CoreSet`
`...`	Fallthrough arguements for defining new methods

Value

The name of the CoreSet

Examples

```
name(clevelandSmall_cSet)
```

---

name<- *name<- Generic*

Description

A generic for the name<- method

Usage

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

Arguments

object	A CoreSet object
...	Fallthrough arguements for defining new methods
value	A character string with the name to assign to the cSet

Value

The name of the CoreSet

Examples

```
name(clevelandSmall_cSet) <- "Cleveland Small"
```

---

pertNumber *pertNumber Generic*

Description

A generic for the pertNumber method

Usage

```
pertNumber(object, ...)
```

Arguments

object	A CoreSet
...	Fallthrough arguements for defining new methods

Value

A 3D array with the number of perturbation experiments per drug and cell line, and data type

Examples

```
pertNumber(clevelandSmall_cSet)
```

---

pertNumber<- *pertNumber<- Generic*

Description

A generic for the pertNumber method

Usage

```
pertNumber(object) <- value
```

Arguments

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

Value

The updated CoreSet

Examples

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

---

phenoInfo *phenoInfo Generic*

Description

Generic for phenoInfo method

Usage

```
phenoInfo(object, mDataType, ...)
```

Arguments

object	The CoreSet to retrieve rna annotations from
mDataType	the type of molecular data
...	Fallthrough argument for defining new parameters in other S4 methods

Value

a data.frame with the experiment info

Examples

```
phenoInfo(clevelandSmall_cSet, mDataType="rna")
```

---

phenoInfo<- *phenoInfo<- Generic*

Description

Generic for phenoInfo replace method

Usage

```
phenoInfo(object, mDataType) <- value
```

Arguments

object	The CoreSet to retrieve molecular experiment annotations from
mDataType	the type of molecular data
value	a dataframe with the new experiment annotations

Value

The updated CoreSet

Examples

```
data(clevelandSmall_cSet)
phenoInfo(clevelandSmall_cSet, mDataType="rna") <- phenoInfo(clevelandSmall_cSet, mDataType="rna")
```

---

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

sensitivitySlot *sensitivitySlot Generic*

Description

sensitivitySlot Generic

Usage

```
sensitivitySlot(object, ...)
```

Arguments

object	A CoreSet to extract the raw sensitivity data from
...	A list to allow new parameters in specific methods

Value

A list of the sensitivity slot contents

Examples

```
data(clevelandSmall_cSet)
sensitivitySlot(clevelandSmall_cSet)
```

sensitivitySlot<- *sensitivitySlot<- Replacement Generic*

Description

sensitivitySlot<- Replacement Generic

Usage

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

Arguments

object	A CoreSet to extract the raw sensitivity data from
...	A list to allow new parameters in specific methods
value	A list of new sensitivity slot data for the rSet

Value

A copy of the CoreSet containing the updated sensitivty slot

Examples

```
data(clevelandSmall_cSet)
sensitivitySlot(clevelandSmall_cSet) <- sensitivitySlot(clevelandSmall_cSet)
```

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?")
```

sensNumber *sensNumber Generic*

Description

A generic for the sensNumber method

Usage

```
sensNumber(object, ...)
```

Arguments

object	A CoreSet
...	Fallthrough arguements for defining new methods

Value

A data.frame with the number of sensitivity experiments per drug and cell line

Examples

```
sensNumber(clevelandSmall_cSet)
```

sensNumber<- *sensNumber<- Generic*

Description

A generic for the sensNumber method

Usage

```
sensNumber(object) <- value
```

Arguments

object	A CoreSet
value	A new data.frame with the number of sensitivity experiments per drug and cell line

Value

The updated CoreSet

Examples

```
sensNumber(clevelandSmall_cSet) <- sensNumber(clevelandSmall_cSet)
```

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)
```

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, 'CAOV3', 'ABT-888')
# Numeric
subset(merckLongTable, 1, c(1, 2))
# Logical
subset(merckLongTable, rowData(merckLongTable)$cell_line1 == 'A2058')
# Call
subset(merckLongTable, cell_line1 == 'A2058',
  drug1 == 'Dasatinib' & drug2 != '5-FU')
```

---

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['CAOV3', 'ABT-888']
# Numeric
merckLongTable[1, c(1, 2)]
# Logical
merckLongTable[rowData(merckLongTable)$cell_line1 == 'A2058', ]
# Call
merckLongTable[.(cell_line1 == 'A2058'),
  .(drug1 == 'Dasatinib' & drug2 != '5-FU')]
```

$,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$viability
```

Index