Package ‘CoreGx’

October 17, 2020

Type Package

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

Version 1.0.2

Date 2020-04-24

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 'PharmaSet' 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, methods, stats, utils, graphics, grDevices, lsa

Suggests pander, BiocStyle, rmarkdown, knitr

License GPL-3

RoxygenNote 7.1.0

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

git_branch RELEASE_3_11

git_last_commit b528829

git_last_commit_date 2020-06-09

Date/Publication 2020-10-16

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:

`.getSupportVec` *.getSupportVec*

Description

.getSupportVec

Usage

```
.getSupportVec(x, output_length = 1001)
```

Arguments

`x`	An input vector of dosages
`output_length`	The length of the returned support vector

Value

`numeric` A numeric vector of interpolated concentrations

`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)
```

`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

```
data(clevelandSmall_cSet)
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
`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.

---

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.

Usage

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

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

## S4 replacement method for signature 'CoreSet,data.frame'
sensitivityInfo(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'
sensitivityMeasures(object)

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

## S4 replacement method for signature 'CoreSet,character'
cellNames(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'
dateCreated(object)

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

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

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

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

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

Arguments

`object`	A `CoreSet` object
`value`	A replacement value
`mDataType`	A `character` with the type of molecular data to return/update
`assay`	`character` Name of the desired assay; if excluded defaults to first assay in the SummarizedExperiment for the given mDataType. Use `assayNames(molecularProfiles(object,` to check which assays are available for a given molecular datatype.

Value

An object of the CoreSet class

Methods (by generic)

- `cellInfo`: Returns the annotations for all the cell lines tested on in the CoreSet
- `cellInfo<-`: Update the cell line annotations
- `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
- `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
- `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
- `sensitivityInfo`: Return the drug dose sensitivity experiment info
- `sensitivityInfo<-`: Update the sensitivity experiment info
- `sensitivityProfiles`: Return the phenotypic data for the drug dose sensitivity
- `sensitivityProfiles<-`: Update the phenotypic data for the drug dose sensitivity
- `sensitivityProfiles<-`: Update the phenotypic data for the drug dose sensitivity
- `sensitivityMeasures`: Returns the available sensitivity profile summaries, for example, whether there are IC50 values available
- `cellNames`: Return the cell names used in the dataset
- `cellNames<-`: Update the cell names used in the dataset
- `fNames`: Return the feature names used in the dataset
- `fNames<-`: Update the feature names used in a molecular profile
- `dateCreated`: Return the date the CoreSet was created
- `name`: Return the name of the CoreSet
- `pertNumber`: Return the summary of available perturbation experiments
- `sensNumber`: Return the summary of available sensitivity experiments
- `pertNumber<-`: Update the summary of available perturbation experiments
- `sensNumber<-`: Update the summary of available sensitivity experiments

Slots

`annotation` A `list` of annotation data about the CoreSet, including the `$name` and the session information for how the object was creating, 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

---

`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)
```

`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)
```

`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")
```

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)
```

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,CoreSet-method`
mDataNames

Description

Returns the molecular data names for the CoreSet.

Usage

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

Arguments

`object`	CoreSet object

Value

Vector of names of the molecular data types

Examples

```
data(clevelandSmall_cSet)
mDataNames(clevelandSmall_cSet)
```

`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,` 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)
```

`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")
```

`sensitivityInfo` *sensitivityInfo Generic*

Description

Generic for sensitivityInfo method

Usage

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

Arguments

`object`	The `CoreSet` to retrieve sensitivity experiment annotations from
`...`	Fallthrough arguments for defining new methods

Value

a `data.frame` with the experiment info

Examples

```
sensitivityInfo(clevelandSmall_cSet)
```

`sensitivityInfo<-` *sensitivityInfo<- Generic*

Description

A generic for the sensitivityInfo replacement method

Usage

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

Arguments

`object`	The `CoreSet` to update
`value`	A `data.frame` with the new sensitivity annotations

Value

Updated `CoreSet`

Examples

```
sensitivityInfo(clevelandSmall_cSet) <- sensitivityInfo(clevelandSmall_cSet)
```

`sensitivityMeasures` *sensitivityMeasures Generi*

Description

A generic for the sensitivityMeasures method

Usage

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

Arguments

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

Value

A `character` vector of all the available sensitivity measures

Examples

```
sensitivityMeasures(clevelandSmall_cSet)
```

`sensitivityProfiles` *sensitivityProfiles Generic*

Description

Generic for sensitivityProfiles method

Usage

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

Arguments

`object`	The `CoreSet` to retrieve sensitivity experiment data from
`...`	Fallthrough arguements for defining new methods

Value

a `data.frame` with the experiment info

Examples

```
sensitivityProfiles(clevelandSmall_cSet)
```

sensitivityProfiles<- *sensitivityProfiles<- Generic*

Description

A generic for the sensitivityProfiles replacement method

Usage

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

Arguments

object	The CoreSet to update
value	A data.frame with the new sensitivity profiles. If a matrix object is passed in, converted to data.frame before assignment

Value

Updated CoreSet

Examples

```
sensitivityProfiles(clevelandSmall_cSet) <- sensitivityProfiles(clevelandSmall_cSet)
```

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)
```

Index