Package 'CoreGx'

June 14, 2019

Type Package

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

Version 0.1.0

Date 2019-05-29

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>.

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0)

Imports lsa, methods, piano, Biobase, stats, rlang

Suggests PharmacoGx

License GPL-3

RoxygenNote 6.1.1

NeedsCompilation no

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

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

Repository CRAN

Date/Publication 2019-06-14 15:10:07 UTC

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

`amcc` *Calculate an Adaptive Matthews Correlation Coefficient*

Description

This function calculates an Adaptive Matthews Correlation Coefficient (AMCC) for two vectors of values identical 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,
  setseed = 12345, 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.

setseed	Allows setting a consitent seed for reproducibility of permutation testing results. Defaults to 12345.
nthread	Number of threads to parallize over. Both the AMCC calculation and the permutation testing is done in parallel.

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 Generic*

Description

Generic for cellInfo method

Usage

```
cellInfo(cSet)
```

Arguments

cSet	The CoreSet to retrieve cell info from

Value

a data.frame with the cell annotations

Examples

```
data(Cleveland_small)
cellInfo(Cleveland_small)
```

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(Cleveland_small) <- cellInfo(Cleveland_small)
```

cellNames *cellNames Generic*

Description

A generic for the cellNames method

Usage

```
cellNames(cSet)
```

Arguments

cSet	The CoreSet to return cell names from

Value

A vector of the cell names used in the CoreSet

Examples

```
cellNames(Cleveland_small)
```

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(Cleveland_small) <- cellNames(Cleveland_small)
```

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 pset object passed in.

Examples

```
checkCSetStructure(Cleveland_small)
```

Cleveland_small *Cleaveland_mut RadioSet subsetted and cast as CoreSet*

Description

Documentation for this dataset will be added at a later date. For now I just need this package to pass the CRAN checks! This dataset powers the exampe usage in the roxygen2 documentation for CoreGx.

Usage

```
data(Cleveland_small)
```

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("gsea", "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 'gsea' 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. For a much more detailed instruction on creating CoreSets, please see the "CreatingCoreSet" vignette.

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 ExpressionSet objects containing molecular profiles
`cell`	A `data.frame` containg 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

Examples

```
## For help creating a CoreSet object, please see the following vignette:
browseVignettes("PharmacoGx")
```

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 objects 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 data 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 two seperate implementations of the CSet class allows the packages to apply the correct model for the given data.

Usage

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

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

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

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

## S4 method for signature 'CoreSet,character'
molecularProfiles(cSet, mDataType)

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

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

## S4 replacement method for signature 'CoreSet,character,data.frame'
```

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

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

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

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

## 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(cSet)

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

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

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

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

## S4 method for signature 'CoreSet'
cSetName(cSet)

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

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

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

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

Arguments

cSet	A CoreSet object
object	A CoreSet object
value	A replacement value
mDataType	A character with the type of molecular data to return/update

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 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
- featureInfo: Return the feature info for the given 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
- dateCreated: Return the date the CoreSet was created
- cSetName: 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 4 `Biobase::ExpressionSet` type object for holding data for RNA, DNA, SNP and Copy Number Variation measurements respectively, with associated `fData` and `pData` 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` *Computes the cosine similarity and significance using permutation test*

Description

Computes the cosine similarity and significance using permutation test

Usage

```
cosinePerm(x, y, nperm = 1000, alternative = c("two.sided", "less",
  "greater"), include.perm = FALSE, setseed = 12345, 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 comput ethe 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'
`setseed`	[integer] is the seed specified by the user. Defaults is '12345'
`nthread`	[integer] is the number of threads to be used to perform the permutations in parallel

Value

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)
```

---

cSetName *cSetName Generic*

Description

A generic for the cSetName method

Usage

```
cSetName(cSet)
```

Arguments

cSet	A CoreSet

Value

The name of the CoreSet

Examples

```
cSetName(Cleveland_small)
```

---

dateCreated *dateCreated Generic*

Description

A generic for the dateCreated method

Usage

```
dateCreated(cSet)
```

Arguments

cSet	A CoreSet

Value

The date the CoreSet was created

Examples

```
dateCreated(Cleveland_small)
```

---

distancePointLine *Calculate shortest distance between point and line*

Description

This function calculates the shortest distance between a point and a line in 2D space.

Usage

```
distancePointLine(x, y, a, b, c)
```

Arguments

x	x-coordinate of point
y	y-coordinate of point
a	coefficient in line equation a * x + b * y + c = 0
b	coefficient in line equation a * x + b * y + c = 0
c	coefficient in line equation a * x + b * y + c = 0

Examples

```
distancePointLine(0, 0, 1, -1, 1)
```

distancePointSegment *Calculate shortest distance between point and line segment*

Description

This function calculates the shortest distance between a point and a line segment in 2D space.

Usage

```
distancePointSegment(x, y, x1, y1, x2, y2)
```

Arguments

x	x-coordinate of point
y	y-coordinate of point
x1	x-coordinate of one endpoint of the line segment
y1	y-coordinate of line segment endpoint with x-coordinate x1
x2	x-coordinate of other endpoint of line segment
y2	y-coordinate of line segment endpoint with x-coordinate x2

Examples

```
distancePointSegment(0, 0, -1, 1, 1, -1)
```

examineGOF *Getter for attributes of an object*

Description

Getter for attributes of an object

Usage

```
examineGOF(pars)
```

Arguments

pars	The object for which attributes are to be returned

Value

A named vector where index ‘Rsquare‘ contains the attributes of the object

featureInfo *featureInfo Generic*

Description

Generic for featureInfo method

Usage

```
featureInfo(cSet, mDataType)
```

Arguments

cSet	The CoreSet to retrieve feature annotations from
mDataType	the type of molecular data

Value

a data.frame with the experiment info

Examples

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

---

fNames *fNames Generic*

Description

A generic for the fNames method

Usage

```
fNames(cSet, mDataType)
```

Arguments

cSet	The CoreSet
mDataType	The molecular data type to return feature names for

Value

A character vector of the feature names

Examples

```
fNames(Cleveland_small, "rna")
```

---

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

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(Cleveland_small)
x <- molecularProfiles(Cleveland_small,"rna")[,1]
y <- molecularProfiles(Cleveland_small,"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)
```

---

intersectList — *Utility to find the intersection between a list of more than two vectors or lists*

Description

This function extends the native intersect function to work on two or more arguments.

Usage

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

Arguments

...	A list of or any number of vector like objects of the same mode, which could also be operated on by the native R set operations

Value

A vector like object of the same mode as the first argument, containing only the intersection common to all arguments to the function

Examples

```
list1 <- list('a', 'b', 'c')
list2 <- list('a', 'c')
list3 <- list('a', 'c', 'd')
listAll <- intersectList(list1, list2, list3)
listAll
```

---

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, setseed = 12345, nthread = 1)
```

Arguments

x, y	factor of the same length with the same number of levels
nperm	number of permutations for significance estimation. If 0, no permutation testing is done
setseed	seed for permutation testing
nthread	can parallelize permutation texting using parallel's mclapply

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(cSet)
```

Arguments

cSet	CoreSet object

Value

Vector of names of the molecular data types

Examples

```
mDataNames(Cleveland_small)
```

mDataNames,CoreSet-method *mDataNames*

Description

Returns the molecular data names for the CoreSet.

Usage

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

Arguments

cSet	CoreSet object

Value

Vector of names of the molecular data types

Examples

```
data(cleveland_small)
mDataNames(Cleveland_small)
```

`molecularProfiles` *molecularProfiles Generic*

Description

Generic for molecularProfiles method

Usage

```
molecularProfiles(cSet, mDataType)
```

Arguments

`cSet`	The `CoreSet` to retrieve molecular profiles from
`mDataType`	the type of molecular data

Value

a `data.frame` with the experiment info

Examples

```
molecularProfiles(Cleveland_small, "rna")
```

`molecularProfiles<-` *molecularProfiles<- Generic*

Description

Generic for molecularProfiles replace method

Usage

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

Arguments

`object`	The `CoreSet` to replace molecular profiles in
`mDataType`	The type of molecular data to be updated
`value`	A `matrix` with the new profiles

Value

Updated CoreSet

Examples

```
molecularProfiles(Cleveland_small, "rna") <- molecularProfiles(Cleveland_small, "rna")
```

---

pertNumber *pertNumber Generic*

Description

A generic for the pertNumber method

Usage

```
pertNumber(cSet)
```

Arguments

cSet	A CoreSet

Value

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

Examples

```
pertNumber(Cleveland_small)
```

---

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(Cleveland_small) <- pertNumber(Cleveland_small)
```

---

phenoInfo *phenoInfo Generic*

Description

Generic for phenoInfo method

Usage

```
phenoInfo(cSet, mDataType)
```

Arguments

cSet	The CoreSet to retrieve rna annotations from
mDataType	the type of molecular data

Value

a data.frame with the experiment info

Examples

```
phenoInfo(Cleveland_small, 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 data.frame with the new experiment annotations

Value

The updated CoreSet

Examples

```
phenoInfo(Cleveland_small, mDataType="rna") <- phenoInfo(Cleveland_small, mDataType="rna")
```

sensitivityInfo *sensitivityInfo Generic*

Description

Generic for sensitivityInfo method

Usage

```
sensitivityInfo(cSet)
```

Arguments

cSet	The CoreSet to retrieve sensitivity experiment annotations from

Value

a data.frame with the experiment info

Examples

```
sensitivityInfo(Cleveland_small)
```

`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(Cleveland_small) <- sensitivityInfo(Cleveland_small)
```

`sensitivityMeasures` *sensitivityMeasures Generic*

Description

A generic for the sensitivityMeasures method

Usage

```
sensitivityMeasures(cSet)
```

Arguments

`cSet`	The `CoreSet`

Value

A `character` vector of all the available sensitivity measures

Examples

```
sensitivityMeasures(Cleveland_small)
```

`sensitivityProfiles` *sensitivityProfiles Generic*

Description

Generic for sensitivityProfiles method

Usage

```
sensitivityProfiles(cSet)
```

Arguments

`cSet`	The `CoreSet` to retrieve sensitivity experiment data from

Value

a `data.frame` with the experiment info

Examples

```
sensitivityProfiles(Cleveland_small)
```

`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(Cleveland_small) <- sensitivityProfiles(Cleveland_small)
```

---

sensNumber *sensNumber Generic*

Description

A generic for the sensNumber method

Usage

```
sensNumber(cSet)
```

Arguments

cSet	A CoreSet

Value

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

Examples

```
sensNumber(Cleveland_small)
```

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(Cleveland_small) <- sensNumber(Cleveland_small)
```

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(Cleveland_small)
```

`symSetDiffList` *Utility to find the symmetric set difference of a list of two or more vectors or lists*

Description

The function finds the symmetric set differnces between all the arguments, defined as Union(args)-Intersection(args)

Usage

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

Arguments

`...`	A list of or any number of vector like objects of the same mode, which could also be operated on by the native R set operations

Value

A vector like object of the same mode as the first argument, containing only the symmetric set difference

Examples

```
list1 <- list('a', 'b', 'c')
list2 <- list('a', 'c')
list3 <- list('a', 'c', 'd')
listAll <- symSetDiffList(list1, list2, list3)
listAll
```

`unionList` *Utility to find the union between a list of more than two vectors or lists*

Description

This function extends the native union function to work on two or more arguments.

Usage

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

Arguments

...	A list of or any number of vector like objects of the same mode, which could also be operated on by the native R set operations

Value

A vector like object of the same mode as the first argument, containing all the elements of all arguments passed to the function

Examples

```
list1 <- list('a', 'b')
list2 <- list('a', 'c')
list3 <- list('c', 'd')
listAll <- unionList(list1, list2, list3)
listAll
```

Index