Package ‘PharmacoGx’

March 23, 2017

Type Package

Title Analysis of Large-Scale Pharmacogenomic Data

Version 1.4.3

Date 2016-08-19

Author Petr Smirnov, Zhaleh Safikhani, Mark Freeman, Benjamin Haibe-Kains

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

Description Contains a set of functions to perform large-scale analysis of pharmacogenomic data.

License Artistic-2.0

Suggests xtable

Encoding UTF-8

Imports Biobase, piano, magicaxis, RColorBrewer, parallel, caTools, methods, downloader, stats, utils, graphics, grDevices, lsa

Depends R (>= 3.3)

RoxygenNote 5.0.1

biocViews GeneExpression, Pharmacogenetics, Pharmacogenomics, Software, Classification

BugReports https://github.com/bhklab/PharmacoGx/issues

NeedsCompilation no

R topics documented:

amce ................................................................. 3
availablePSets ..................................................... 4
CCLEsmall .......................................................... 4
cellInfo ........................................................... 5
cellInfo<- ......................................................... 5
cellNames ......................................................... 6
cellNames<- ....................................................... 6
checkPSetStructure ............................................... 7
CMAPsmall ........................................................ 7
computeABC ....................................................... 8
computeAmax ...................................................... 9
computeAUC ....................................................... 10
computeIC50 ....................................................... 11
computeSlope ........................................... 12
cosinePerm ........................................... 14
dateCreated .......................................... 15
dim.PharmacoSet-method .............................. 15
downloadPertSig ....................................... 16
downloadPSet .......................................... 16
drugDoseResponseCurve ...................... 17
drugInfo ................................................ 19
drugInfo<- ........................................... 19
drugNames ............................................. 20
drugNames<- .......................................... 20
drugPerturbationSig ......................... 21
drugSensitivitySig ................................. 22
featureInfo .......................................... 23
featureInfo<- ........................................ 24
fNames ................................................ 24
GDSCsmall ............................................ 25
gwc ...................................................... 25
HGACgenes ........................................... 26
intersectList ......................................... 27
intersectPSet ......................................... 27
logLogisticRegression ...................... 28
mcc ...................................................... 30
mDataNames ........................................... 30
molecularProfiles .................................... 31
molecularProfiles<- ................................. 32
pertNumber ........................................... 32
pertNumber<- .......................................... 33
PharmacoSet .......................................... 33
PharmacoSet-class .................................... 34
phenoInfo ............................................. 38
phenoInfo<- .......................................... 38
pSetName ............................................. 39
sensitivityInfo ....................................... 39
sensitivityInfo<- .................................... 40
sensitivityMeasures ............................... 40
sensitivityProfiles .................................. 41
sensitivityProfiles<- ............................... 41
sensNumber ........................................... 42
sensNumber<- .......................................... 42
show,PharmacoSet-method .................... 43
show,PharmacoSig-method ..................... 43
showSigAnnot ........................................ 43
subsetTo ............................................. 44
summarizeMolecularProfiles .................. 46
summarizeSensitivityProfiles ............... 47
unionList ............................................. 48
[,]PharmacoSet-method ............................. 48
Index ................................................... 50
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 permutation to use for estimating significance. If 0, then no p-value is calculated.
- `setseed`: Allows setting a consistent seed for reproducibility of permutation testing results. Defaults to 12345.
- `nthread`: Number of threads to parallelize 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

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

Return a table of PharmacoSets available for download

Description

The function fetches a table of all PharmacoSets available for download from the PharmacoGx server. The table includes the names of the PharmacoSet, the types of data available in the object, and the date of last update.

Usage

availablePSets(saveDir = file.path(".", "PSets"), myfn = "PSets.csv", verbose = TRUE)

Arguments

* saveDir: character, Directory to save the table of PSets
* myfn: character, The filename for the table of PSets
* verbose: bool, Should status messages be printed during download.

Value

A data.frame with details about the available PharmacoSet objects

Examples

if (interactive()){ availablePSets() }

CCLEsmall

Cancer Cell Line Encyclopedia (CCLE) Example PharmacoSet

Description

A small example version of the CCLE PharmacoSet, used in the documentation examples. All credit for the data goes to the CCLE group at the Broad Institute. This is not a full version of the dataset, most of of the dataset was removed to make runnable example code. For the full dataset, please download using the downloadPSet function.

Usage

data(CCLEsmall)

Format

PharmacoSet object
References


cellInfo

description
cellInfo Generic

Description

Generic for cellInfo method

Usage

cellInfo(pSet)

Arguments

pSet The PharmacoSet to retrieve cell info from

Value

a data.frame with the cell annotations

Examples

data(CCLEsmall)
cellInfo(CCLEsmall)

cellInfo<- Generic
description
cellInfo<- Generic

Description

Generic for cellInfo replace method

Usage

cellInfo(object) <- value

Arguments

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

Value

Updated PharmacoSet
Examples

data(CCLEsmall)
cellInfo(CCLEsmall) <- cellInfo(CCLEsmall)

cellNames <-
cellNames Generic

Description
A generic for the cellNames method

Usage

cellNames(pSet)

Arguments

pSet The PharmacoSet to return cell names from

Value
A vector of the cell names used in the PharmacoSet

Examples

data(CCLEsmall)
cellNames(CCLEsmall)

---

cellNames<- <- Generic

Description
A generic for the cellNames replacement method

Usage

cellNames(object) <- value

Arguments

object The PharmacoSet to update
value A character vector of the new cell names

Value
Updated PharmacoSet
checkPSetStructure

Examples

data(CCLEsmall)
cellNames(CCLEsmall) <- cellNames(CCLEsmall)

checkPSetStructure  A function to verify the structure of a PharmacoSet

Description

This function checks the structure of a PharmacoSet, 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

checkPSetStructure(pSet, plotDist = FALSE, result.dir = ".")

Arguments

  pSet          A PharmacoSet 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

data(CCLEsmall)

  checkPSetStructure(CCLEsmall)

CMAPsmall  Connectivity Map Example PharmacoSet

Description

A small example version of the Connectivity Map PharmacoSet, used in the documentation examples. All credit for the data goes to the Connectivity Map group at the Broad Institute. This is not a full version of the dataset, most of the dataset was removed to make runnable example code. For the full dataset, please download using the downloadPSet function.

Usage

data(CMAPsmall)
computeABC

Fits dose-response curves to data given by the user and returns the ABC of the fitted curves.

Description

Fits dose-response curves to data given by the user and returns the ABC of the fitted curves.

Usage

computeABC(conc1, conc2, viability1, viability2, Hill_fit1, Hill_fit2, conc_as_log = FALSE, viability_as_pct = TRUE, trunc = TRUE, verbose = TRUE)

Arguments

conc1 [vector] is a vector of drug concentrations.
conc2 [vector] is a vector of drug concentrations.
viability1 [vector] is a vector whose entries are the viability values observed in the presence of the drug concentrations whose logarithms are in the corresponding entries of conc1, expressed as percentages of viability in the absence of any drug.
viability2 [vector] is a vector whose entries are the viability values observed in the presence of the drug concentrations whose logarithms are in the corresponding entries of conc2, expressed as percentages of viability in the absence of any drug.
Hill_fit1 [list or vector] In the order: c("Hill Slope", "E_inf", "EC50"), the parameters of a Hill Slope as returned by logLogisticRegression. If conc_as_log is set then the function assumes logEC50 is passed in, and if viability_as_pct flag is set, it assumes E_inf is passed in as a percent. Otherwise, E_inf is assumed to be a decimal, and EC50 as a concentration.
Hill_fit2 [list or vector] In the order: c("Hill Slope", "E_inf", "EC50"), the parameters of a Hill Slope as returned by logLogisticRegression. If conc_as_log is set then the function assumes logEC50 is passed in, and if viability_as_pct flag is set, it assumes E_inf is passed in as a percent. Otherwise, E_inf is assumed to be a decimal, and EC50 as a concentration.
conc_as_log [logical], if true, assumes that log10-concentration data has been given rather than concentration data.
viability_as_pct [logical], if false, assumes that viability is given as a decimal rather than a percentage, and returns ABC as a decimal. Otherwise, viability is interpreted as percent, and AUC is returned 0-100.
trunc [logical], if true, causes viability data to be truncated to lie between 0 and 1 before curve-fitting is performed.
verbose [logical], if true, causes warnings thrown by the function to be printed.
computeAmax

Value

The numeric area of the absolute difference between the two hill slopes

Examples

dose1 <- c("0.0025","0.008","0.025","0.08","0.25","0.8","2.53","8")
viability1 <- c("108.67","111","102.16","100.27","90","87","74","57")
dose2 <- c("0.0025","0.008","0.025","0.08","0.25","0.8","2.53","8")
viability2 <- c("100.94","112.5","86","104.16","75","68","48","29")
computeABC(dose1, dose2, viability1, viability2)

computeAmax

Fits dose-response curves to data given by the user and returns the Amax of the fitted curve. Amax: 100 - viability at maximum concentration (in fitted curve)

Description

Fits dose-response curves to data given by the user and returns the Amax of the fitted curve. Amax: 100 - viability at maximum concentration (in fitted curve)

Usage

computeAmax(concentration, viability, trunc = TRUE, verbose = FALSE)

Arguments

concentration

[vector] is a vector of drug concentrations.

viability

[vector] is a vector whose entries are the viability values observed in the presence of the drug concentrations whose logarithms are in the corresponding entries of the log_conc, expressed as percentages of viability in the absence of any drug.

trunc

[logical], if true, causes viability data to be truncated to lie between 0 and 1 before curve-fitting is performed.

verbose

[logical] should warnings be printed

Value

The numerical Amax

Examples

dose <- c("0.0025","0.008","0.025","0.08","0.25","0.8","2.53","8")
viability <- c("108.67","111","102.16","100.27","90","87","74","57")
computeAmax(dose, viability)
computeAUC  Computes the AUC for a Drug Dose Viability Curve

Description

Returns the AUC (Area Under the drug response Curve) given concentration and viability as input, normalized by the concentration range of the experiment. The area returned is the response (1-Viability) area, i.e. area under the curve when the response curve is plotted on a log10 concentration scale, with high AUC implying high sensitivity to the drug. The function can calculate both the area under a fitted Hill Curve to the data, and a trapz numeric integral of the actual data provided. Alternatively, the parameters of a Hill Slope returned by logLogisticRegression can be passed in if they already known.

Usage

```
computeAUC(concentration, viability, Hill_fit, conc_as_log = FALSE,
            viability_as_pct = TRUE, trunc = TRUE, area.type = c("Fitted",
            "Actual"), verbose = TRUE)
```

Arguments

- **concentration** [vector] is a vector of drug concentrations.
- **viability** [vector] is a vector whose entries are the viability values observed in the presence of the drug concentrations whose logarithms are in the corresponding entries of conc, where viability 0 indicates that all cells died, and viability 1 indicates that the drug had no effect on the cells.
- **Hill_fit** [list or vector] In the order: c("Hill Slope", "E_inf", "EC50"), the parameters of a Hill Slope as returned by logLogisticRegression. If conc_as_log is set then the function assumes logEC50 is passed in, and if viability_as_pct flag is set, it assumes E_inf is passed in as a percent. Otherwise, E_inf is assumed to be a decimal, and EC50 as a concentration.
- **conc_as_log** [logical], if true, assumes that log10-concentration data has been given rather than concentration data.
- **viability_as_pct** [logical], if false, assumes that viability is given as a decimal rather than a percentage, and returns AUC as a decimal. Otherwise, viability is interpreted as percent, and AUC is returned 0-100.
- **trunc** [logical], if true, causes viability data to be truncated to lie between 0 and 1 before curve-fitting is performed.
- **area.type** Should the area be computed using the actual data ("Actual"), or a fitted curve ("Fitted")
- **verbose** [logical], if true, causes warnings thrown by the function to be printed.

Value

Numeric AUC value
computeIC50

Computes the ICn for any n in 0-100 for a Drug Dose Viability Curve

Description

Returns the ICn for any given nth percentile when given concentration and viability as input, normalized by the concentration range of the experiment. A Hill Slope is first fit to the data, and the ICn is inferred from the fitted curve. Alternatively, the parameters of a Hill Slope returned by logLogisticRegression can be passed in if they already known.

Usage

```r
computeIC50(conc, viability, Hill_fit, conc_as_log = FALSE, viability_as_pct = TRUE, verbose = TRUE, trunc = TRUE)
computeICn(conc, viability, Hill_fit, n, conc_as_log = FALSE, viability_as_pct = TRUE, verbose = TRUE, trunc = TRUE)
```

Arguments

- `conc` [vector]: is a vector of drug concentrations.
- `viability` [vector]: is a vector whose entries are the viability values observed in the presence of the drug concentrations whose logarithms are in the corresponding entries of `conc`, where viability 0 indicates that all cells died, and viability 1 indicates that the drug had no effect on the cells.
- `Hill_fit` [list or vector]: In the order: c("Hill Slope", "E_inf", "EC50"), the parameters of a Hill Slope as returned by logLogisticRegression. If `conc_as_log` is set then the function assumes logEC50 is passed in, and if `viability_as_pct` flag is set, it assumes E_inf is passed in as a percent. Otherwise, E_inf is assumed to be a decimal, and EC50 as a concentration.
- `conc_as_log` [logical]: if true, assumes that log10-concentration data has been given rather than concentration data, and that log10(ICn) should be returned instead of ICn.
- `viability_as_pct` [logical]: if false, assumes that viability is given as a decimal rather than a percentage, and that E_inf passed in as decimal.
- `verbose` [logical]: if true, causes warnings thrown by the function to be printed.
- `trunc` [logical]: if true, causes viability data to be truncated to lie between 0 and 1 before curve-fitting is performed.
- `n` [numeric]: The percentile concentration to compute. If `viability_as_pct` set, assumed to be percentage, otherwise assumed to be a decimal value.

Examples

```r
dose <- c("0.0025", "0.008", "0.025", "0.08", "0.25", "0.8", "2.53", "8")
viability <- c("108.67", "111", "102.16", "100.27", "90", "87", "74", "57")
computeAUC(dose, viability)
```
computeSlope

Value

a numeric value for the concentration of the nth percentile viability reduction

Functions

- computeIC50: Returns the IC50 of a Drug Dose response curve

Examples

dose <- c("0.0025", "0.008", "0.025", "0.08", "0.25", "0.8", "2.53", "8")
viability <- c("108.67", "111", "102.16", "100.27", "90", "87", "74", "57")
computeIC50(dose, viability)
computeICn(dose, viability, n=10)

computeSlope

Return Slope (normalized slope of the drug response curve) for an experiment of a pSet by taking its concentration and viability as input.

Description

Return Slope (normalized slope of the drug response curve) for an experiment of a pSet by taking its concentration and viability as input.

Usage

computeSlope(concentration, viability, trunc = TRUE, verbose = TRUE)

Arguments

- concentration [vector] A concentration range that the AUC should be computed for that range. Concentration range by default considered as not logarithmic scaled. Converted to numeric by function if necessary.
- viability [vector] Viabilities corresponding to the concentration range passed as first parameter. The range of viability values by definition should be between 0 and 100. But the viabilities greater than 100 and lower than 0 are also accepted.
- trunc [binary] A flag that identify if the viability values should be truncated to be in the range of (0,100)
- verbose [boolean] If 'TRUE' the function will retrun warnings and other informative messages.

Value

Returns the normalized linear slope of the drug response curve

Examples

dose <- c("0.0025", "0.008", "0.025", "0.08", "0.25", "0.8", "2.53", "8")
viability <- c("108.67", "111", "102.16", "100.27", "90", "87", "74", "57")
computeSlope(dose, viability)
**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**

```r
connectivityScore(x, y, method = c("gsea", "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**


Examples

```r
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)
x <- cbind(xValue, xSig)
y <- 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)
```

### Description

Computes the cosine similarity and significance using permutation test

### Usage

```r
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 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'
- `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

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

**Description**
A generic for the `dateCreated` method

**Usage**
dateCreated(pSet)

**Arguments**
pSet A `PharmacoSet`

**Value**
The date the `PharmacoSet` was created

**Examples**

data(CCLEsmall)
dateCreated(CCLEsmall)

---

**dim,PharmacoSet-method**

*Get the dimensions of a `PharmacoSet`*

**Description**
Get the dimensions of a `PharmacoSet`

**Usage**

```
## S4 method for signature 'PharmacoSet'
dim(x)
```

**Arguments**
x `PharmacoSet`

**Value**
A named vector with the number of Cells and Drugs in the `PharmacoSet`
downloadPertSig  Download Drug Perturbation Signatures

Description
This function allows you to download an array of drug perturbation signatures, as would be computed by the drugPerturbationSig function, for the available perturbation PharmacoSets. This function allows the user to skip these very lengthy calculation steps for the datasets available, and start their analysis from the already computed signatures.

Usage

downloadPertSig(name, saveDir = file.path(".", "PSets", "Sigs"), myfn = NULL, verbose = TRUE)

Arguments
- **name**: Character string, the name of the PharmacoSet for which to download signatures. The name should match the names returned in the availablePSets table.
- **saveDir**: Character string with the folder path where the PharmacoSet should be saved. Defaults to "./PSets/Sigs/". Will create directory if it does not exist.
- **myfn**: character string, the file name to save the dataset under. Defaults to NULL.
- **verbose**: bool Should status messages be printed during download. Defaults to TRUE.

Value
An array type object containing the signatures.

Examples

```
if (interactive()){
  downloadPertSig("CMAP")
}
```

downloadPSet  Download a PharmacoSet object

Description
This function allows you to download a PharmacoSet object for use with this package. The PharmacoSets have been extensively curated and organised within a PharmacoSet class, enabling use with all the analysis tools provided in PharmacoGx.

Usage

downloadPSet(name, saveDir = file.path(".", "PSet"), pSetFileName = NULL, verbose = TRUE)
**drugDoseResponseCurve**

**Arguments**

- **name** [Character] String, the name of the PharmacoSet to download.
- **saveDir** [Character] String with the folder path where the PharmacoSet should be saved. Defaults to `.\Psets\`. Will create directory if it does not exist.
- **pSetFileName** [Character] String, the file name to save the dataset under.
- **verbose** [bool] Should status messages be printed during download. Defaults to TRUE.

**Value**

A PSet object with the dataset, downloaded from our server.

**Examples**

```r
if (interactive()){
  downloadPSet("CMAP")
}
```

---

**drugDoseResponseCurve**  
Plot drug response curve of a given drug and a given cell for a list of pSets (objects of the PharmacoSet class).

**Description**

Given a list of PharmacoSets, the function will plot the drug_response curve, for a given drug/cell pair. The y axis of the plot is the viability percentage and x axis is the log transformed concentrations. If more than one pSet is provided, a light grey area would show the common concentration range between pSets. User can ask for type of sensitivity measurement to be shown in the plot legend. The user can also provide a list of their own concentrations and viability values, as in the examples below, and it will be treated as experiments equivalent to values coming from a pset. The names of the concentration list determine the legend labels.

**Usage**

```r
drugDoseResponseCurve(drug, cellline, pSets = list(), concentrations = list(), viabilities = list(), conc_as_log = FALSE, viability_as_pct = TRUE, trunc = TRUE, legends.label = c("ic50_published", "gi50_published", "auc_published", "auc_recomputed", "ic50_recomputed"), ylim = c(0, 100), xlim, mycol, title, plot.type = c("Fitted", "Actual", "Both"), summarize.replicates = TRUE, lwd = 0.5, cex = 0.7, cex.main = 0.9, legend.loc = "topright", verbose = TRUE)
```

**Arguments**

- **drug** [string] A drug name for which the drug response curve should be plotted. If the plot is desirable for more than one pharmaco set, a unique drug id should be provided.
- **cellline** [string] A cell line name for which the drug response curve should be plotted. If the plot is desirable for more than one pharmaco set, a unique cell id should be provided.
drugDoseResponseCurve

pSets: [list] a list of PharmacoSet objects, for which the function should plot the curves.

concentrations, viabilities: [list] A list of concentrations and viabilities to plot, the function assumes that concentrations[[i]] is plotted against viabilities[[i]]. The names of the concentration list are used to create the legend labels.

conc_as_log: [logical], if true, assumes that log10-concentration data has been given rather than concentration data, and that log10(ICn) should be returned instead of ICn. Applies only to the concentrations parameter.

viability_as_pct: [logical], if false, assumes that viability is given as a decimal rather than a percentage, and that E_inf passed in as decimal. Applies only to the viabilities parameter.

trunc: [bool] Should the viability values be truncated to lie in [0-100] before doing the fitting?

legends.label: [vector] A vector of sensitivity measurement types which could be any combination of ic50_published, auc_published, auc_recomputed and auc_recomputed_star. A legend will be displayed on the top right of the plot which each line of the legend is the values of requested sensitivity measurements for one of the requested pSets. If this parameter is missed no legend would be provided for the plot.

ylim: [vector] A vector of two numerical values to be used as ylim of the plot. If this parameter would be missed c(0,100) would be used as the ylim of the plot.

xlim: [vector] A vector of two numerical values to be used as xlim of the plot. If this parameter would be missed the minimum and maximum concentrations between all the pSets would be used as plot xlim.

mycol: [vector] A vector with the same length of the pSets parameter which will determine the color of the curve for the pharmaco sets. If this parameter is missed default colors from Reolorbrewer package will be used as curves color.

title: [character] The title of the graph. If no title is provided, then it defaults to 'Drug': 'Cell Line'.

plot.type: [character] Plot type which can be the actual one ("Actual") or the one fitted by log logistic regression ("Fitted") or both of them ("Both"). If this parameter is missed by default actual curve is plotted.

summarize.replicates: [character] If this parameter is set to true replicates are summarized and replicates are plotted individually otherwise.

lwd: [numeric] The line width to plot with.

cex: [numeric] The cex parameter passed to plot.

cex.main: [numeric] The cex.main parameter passed to plot, controls the size of the titles.

legend.loc: And argument passable to xy.coords for the position to place the legend.

verbose: [boolean] Should warning messages about the data passed in be printed?

Value

Plots to the active graphics device and returns and invisible NULL.
Examples

```r
if (interactive()) {
  drugDoseResponseCurve(concentrations=list("Experiment 1"=c(.008, .04, .2, 1)),
                         viabilities=list(c(100,50,30,1)), plot.type="Both")
}
```

---

**drugInfo**

**Description**

Generic for drugInfo method

**Usage**

```
drugInfo(pSet)
```

**Arguments**

- **pSet**: The PharmacoSet to retrieve drug info from

**Value**

a data.frame with the drug annotations

**Examples**

```r
data(CCLEsmall)
drugInfo(CCLEsmall)
```

---

**drugInfo<-**

**drugInfo<- Generic**

**Description**

Generic for drugInfo replace method

**Usage**

```
drugInfo(object) <- value
```

**Arguments**

- **object**: The PharmacoSet to replace drug info in
- **value**: A data.frame with the new drug annotations

**Value**

Updated PharmacoSet
Examples

data(CCLEsmall)
drugInfo(CCLEsmall) <- drugInfo(CCLEsmall)

drugNames <-

---

drugNames  

Description
A generic for the drugNames method

Usage

drugNames(pSet)

Arguments

<table>
<thead>
<tr>
<th>pSet</th>
</tr>
</thead>
<tbody>
<tr>
<td>The PharmacoSet to return drug names from</td>
</tr>
</tbody>
</table>

Value
A vector of the drug names used in the PharmacoSet

Examples

data(CCLEsmall)
drugNames(CCLEsmall)

---

drugNames<- <-

---

Description
A generic for the drugNames replacement method

Usage

drugNames(object) <- value

Arguments

<table>
<thead>
<tr>
<th>object</th>
</tr>
</thead>
<tbody>
<tr>
<td>The PharmacoSet to update</td>
</tr>
</tbody>
</table>

| value  |
| A character vector of the new drug names |

Value
Updated PharmacoSet
**drugPerturbationSig**

**Examples**

```r
data(CCLEsmall)
drugNames(CCLEsmall) <- drugNames(CCLEsmall)
```

**drugPerturbationSig**  
*Creates a signature representing gene expression (or other molecular profile) change induced by administrating a drug, for use in drug effect analysis.*

**Description**

Given a PharmacoSet of the perturbation experiment type, and a list of drugs, the function will compute a signature for the effect of drug concentration on the molecular profile of a cell. The algorithm uses a regression model which corrects for experimental batch effects, cell specific differences, and duration of experiment to isolate the effect of the concentration of the drug applied. The function returns the estimated coefficient for concentration, the t-stat, the p-value and the false discovery rate associated with that coefficient, in a 3 dimensional array, with genes in the first direction, drugs in the second, and the selected return values in the third.

**Usage**

```r
drugPerturbationSig(pSet, mDataType, drugs, features, nthread = 1,
                   returnValues = c("estimate", "tstat", "pvalue", "fdr"), verbose = FALSE)
```

**Arguments**

- `pSet`  
  [PharmacoSet] a PharmacoSet of the perturbation experiment type
- `mDataType`  
  [character] which one of the molecular data types to use in the analysis, out of dna, rna, rnaseq, snp, cnv
- `drugs`  
  [character] a vector of drug names for which to compute the signatures. Should match the names used in the PharmacoSet.
- `features`  
  [character] a vector of features for which to compute the signatures. Should match the names used in correspondant molecular data in PharmacoSet.
- `nthread`  
  [numeric] if multiple cores are available, how many cores should the computation be parallelized over?
- `returnValues`  
  [character] Which of estimate, t-stat, p-value and fdr should the function return for each gene drug pair?
- `verbose`  
  [bool] Should diagnostic messages be printed? (default false)

**Value**

list  a 3D array with genes in the first dimension, drugs in the second, and return values in the third.

**Examples**

```r
data(CMAPsmall)
drug.perturbation <- drugPerturbationSig(CMAPsmall, mDataType="rna", nthread=1)
print(drug.perturbation)
```
**drugSensitivitySig**  

Creates a signature representing the association between gene expression (or other molecular profile) and drug dose response, for use in drug sensitivity analysis.

**Description**

Given a PharmacoSet of the sensitivity experiment type, and a list of drugs, the function will compute a signature for the effect gene expression on the molecular profile of a cell. The function returns the estimated coefficient, the t-stat, the p-value and the false discovery rate associated with that coefficient, in a 3 dimensional array, with genes in the first direction, drugs in the second, and the selected return values in the third.

**Usage**

```
drugSensitivitySig(pSet, mDataType, drugs, features,  
sensitivity.measure = "auc_recomputed", molecular.summary.stat = c("mean",  
"median", "first", "last", "or", "and"),  
sensitivity.summary.stat = c("mean", "median", "first", "last"),  
returnValues = c("estimate", "pvalue", "fdr"), sensitivity.cutoff,  
standardize = c("SD", "rescale", "none"), nthread = 1, verbose = TRUE,  
...)
```

**Arguments**

- **pSet**  
  [PharmacoSet] a PharmacoSet of the perturbation experiment type
- **mDataType**  
  [character] which one of the molecular data types to use in the analysis, out of dna, rna, rnaseq, snp, cnv
- **drugs**  
  [character] a vector of drug names for which to compute the signatures. Should match the names used in the PharmacoSet.
- **features**  
  [character] a vector of features for which to compute the signatures. Should match the names used in correspondant molecular data in PharmacoSet.
- **sensitivity.measure**  
  [character] which measure of the drug dose sensitivity should the function use for its computations? Use the sensitivityMeasures function to find out what measures are available for each PSet.
- **molecular.summary.stat**  
  What summary statistic should be used to summarize duplicates for cell line molecular profile measurements?
- **sensitivity.summary.stat**  
  What summary statistic should be used to summarize duplicates for cell line sensitivity measurements?
- **returnValues**  
  [character] Which of estimate, t-stat, p-value and fdr should the function return for each gene drug pair?
- **sensitivity.cutoff**  
  Allows to provide upper and lower bounds to sensitivity measures in the cases where the values exceed physical values due to numerical or other errors.

...
**standardize** [character] One of "SD", "rescale", or "none", for the form of standardization of the data to use. If "SD", the data is scaled so that SD = 1. If rescale, then the data is scaled so that the 95 interquantile range lies in [0,1]. If none no rescaling is done.

**nthread** [numeric] if multiple cores are available, how many cores should the computation be parallelized over?

**verbose** [boolean] 'TRUE' if the warnings and other informative message should be displayed

... additional arguments not currently fully supported by the function

**Value**

list a 3D array with genes in the first dimension, drugs in the second, and return values in the third.

**Examples**

```r
data(GDSCsmall)
drug.sensitivity <- drugSensitivitySig(GDSCsmall, mDataType="rna", nthread=1, features = fNames(GDSCsmall, "rna")[[1]])
print(drug.sensitivity)
```

---

**featureInfo**

**featureInfo Generic**

**Description**

Generic for featureInfo method

**Usage**

```r
featureInfo(pSet, mDataType)
```

**Arguments**

- **pSet** The PharmacoSet to retrieve feature annotations from
- **mDataType** the type of molecular data

**Value**

a data.frame with the experiment info

**Examples**

```r
data(CCLEsmall)
featureInfo(CCLEsmall, "rna")
```
featureInfo<- Generic

Description
Generic for featureInfo replace method

Usage
featureInfo(object, mDataType) <- value

Arguments
  object The PharmacoSet 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 PharmacoSet

Examples
data(CCLEsmall)
featureInfo(CCLEsmall, "rna") <- featureInfo(CCLEsmall, "rna")

fNames Generic

Description
A generic for the fNames method

Usage
fNames(pSet, mDataType)

Arguments
  pSet The PharmacoSet
  mDataType The molecular data type to return feature names for

Value
A character vector of the feature names

Examples
data(CCLEsmall)
fNames(CCLEsmall, "rna")
GDSCsmall

Genomics of Drug Sensitivity in Cancer Example PharmacoSet

Description

A small example version of the Genomics of Drug Sensitivity in Cancer Project PharmacoSet, used in the documentation examples. All credit for the data goes to the Genomics of Drug Sensitivity in Cancer Project group at the Sanger. This is not a full version of the dataset, most of the dataset was removed to make runnable example code. For the full dataset, please download using the downloadPSet function.

Usage

data(GDSCsmall)

Format

PharmacoSet object

References


---

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 statistics) 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 statistics) 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
Data frame containing gene identifiers and direction change.

**Value**

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

**Examples**

```r
data(CCLEsmall)
x <- molecularProfiles(CCLEsmall,"rna")[,1]
y <- molecularProfiles(CCLEsmall,"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)
```

**Description**

A gene signature for HDAC inhibitors, as detailed by Glaser et al. The signature is mapped from the probe to gene level using `probeGeneMapping`.

**Usage**

```r
data(HDAC_genes)
```

**Format**

A 13x2 data frame with gene identifiers in the first column and direction change in the second.

**References**

**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**

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

**intersectPSet**

*Intersects objects of the PharmacoSet class, subsetting them to the common drugs and/or cell lines as selected by the user.*

**Description**

Given a list of PharmacoSets, the function will find the common drugs, and/or cell lines, and return PharmacoSets that contain data only pertaining to the common drugs, and/or cell lines. The mapping between dataset drug and cell names is done using annotations found in the PharmacoSet object’s internal curation slot

**Usage**

```r
intersectPSet(pSets, intersectOn = c("drugs", "cell.lines", "concentrations"), cells, drugs, strictIntersect = FALSE, verbose = TRUE, nthread = 1)
```
Arguments

pSets [list] a list of PharmacoSet objects, of which the function should find the intersection
intersectOn [character] which identifiers to intersect on, drugs or cell lines
cells a vector of common cell lines between pSets. In case user is interested on getting intersection on certain cell lines, they can provide their list of cell lines
drugs a vector of common drugs between pSets. In case user is interested on getting intersection on certain drugs, they can provide their list of drugs.
strictIntersect [boolean] Should the intersection keep only the drugs and cell lines that have been tested on together?
verbose [boolean] Should the function announce its key steps?
nthread [numeric] The number of cores to use to run intersection on concentrations

Value

list a list of pSets, containing only the intersection

Examples

data(GDSCsmall)
data(CCLEsmall)
common <- intersectPSet(list('GDSC'=GDSCsmall, 'CCLE'=CCLEsmall), intersectOn = c("drugs", "cell.lines"))
common$CGP
common$CCLE

logLogisticRegression

logLogisticRegression  Fits curves of the form $E = E_{\text{inf}} + (1 - E_{\text{inf}}) / (1 + (c/EC50)^{HS})$ to dose-response data points $(c, E)$ given by the user and returns a vector containing estimates for $HS$, $E_{\text{inf}}$, and $EC50$.

Description

By default, logLogisticRegression uses an L-BFGS algorithm to generate the fit. However, if this fails to converge to solution, logLogisticRegression samples lattice points throughout the parameter space. It then uses the lattice point with minimal least-squares residual as an initial guess for the optimal parameters, passes this guess to drm, and re-attempts the optimization. If this still fails, logLogisticRegression uses the PatternSearch algorithm to fit a log-logistic curve to the data.

Usage

logLogisticRegression(conc, viability, density = c(2, 10, 2),
                      step = 0.5/density, precision = 0.05, lower_bounds = c(0, 0, -6),
                      upper_bounds = c(4, 1, 6), scale = 0.07, family = c("normal", "Cauchy"),
                      median_n = 1, conc_as_log = FALSE, viability_as_pct = TRUE,
                      trunc = TRUE, verbose = FALSE)
Arguments

conc [vector] is a vector of drug concentrations.

viability [vector] is a vector whose entries are the viability values observed in the presence of the drug concentrations whose logarithms are in the corresponding entries of the log_conc, where viability 0 indicates that all cells died, and viability 1 indicates that the drug had no effect on the cells.

density [vector] is a vector of length 3 whose components are the numbers of lattice points per unit length along the HS-, E_inf-, and base-10 logarithm of the EC50-dimensions of the parameter space, respectively.

step [vector] is a vector of length 3 whose entries are the initial step sizes in the HS, E_inf, and base-10 logarithm of the EC50 dimensions, respectively, for the PatternSearch algorithm.

precision is a positive real number such that when the ratio of current step size to initial step size falls below it, the PatternSearch algorithm terminates. A smaller value will cause LogisticPatternSearch to take longer to complete optimization, but will produce a more accurate estimate for the fitted parameters.

lower_bounds [vector] is a vector of length 3 whose entries are the lower bounds on the HS, E_inf, and base-10 logarithm of the EC50 parameters, respectively.

upper_bounds [vector] is a vector of length 3 whose entries are the upper bounds on the HS, E_inf, and base-10 logarithm of the EC50 parameters, respectively.

density [vector] is a vector of length 3 whose components are the numbers of lattice points per unit length along the HS-, E_inf-, and base-10 logarithm of the EC50-dimensions of the parameter space, respectively.

step [vector] is a vector of length 3 whose entries are the initial step sizes in the HS, E_inf, and base-10 logarithm of the EC50 dimensions, respectively, for the PatternSearch algorithm.

precision is a positive real number such that when the ratio of current step size to initial step size falls below it, the PatternSearch algorithm terminates. A smaller value will cause LogisticPatternSearch to take longer to complete optimization, but will produce a more accurate estimate for the fitted parameters.

lower_bounds [vector] is a vector of length 3 whose entries are the lower bounds on the HS, E_inf, and base-10 logarithm of the EC50 parameters, respectively.

upper_bounds [vector] is a vector of length 3 whose entries are the upper bounds on the HS, E_inf, and base-10 logarithm of the EC50 parameters, respectively.

scale is a positive real number specifying the shape parameter of the Cauchy distribution.

family [character], if "cauchy", uses MLE under an assumption of Cauchy-distributed errors instead of sum-of-squared-residuals as the objective function for assessing goodness-of-fit of dose-response curves to the data. Otherwise, if "normal", uses MLE with a gaussian assumption of errors.

median_n If the viability points being fit were medians of measurements, they are expected to follow a median of family distribution, which is in general quite different from the case of one measurement. Median_n is the number of measurements the median was taken of. If the measurements are means of values, then both the Normal and the Cauchy distributions are stable, so means of Cauchy or Normal distributed variables are still Cauchy and normal respectively.

conc_as_log [logical], if true, assumes that log10-concentration data has been given rather than concentration data, and that log10(EC50) should be returned instead of EC50.

viability_as_pct [logical], if false, assumes that viability is given as a decimal rather than a percentage, and that E_inf should be returned as a decimal rather than a percentage.

trunc [logical], if true, causes viability data to be truncated to lie between 0 and 1 before curve-fitting is performed.

verbose [logical], if true, causes warnings thrown by the function to be printed.

Value

A vector containing estimates for HS, E_inf, and EC50.
Examples

dose <- c("0.0025","0.008","0.025","0.08","0.25","0.8","2.53","8")
viability <- c("108.67","111","102.16","100.27","90","87","74","57")
computeAUC(dose, viability)

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 entries 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 testing 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

Description

Returns the molecular data names for the PharmacoSet.

Usage

mDataNames(pSet)
molecularProfiles

Arguments

pSet PharmacoSet object

Value

Vector of names of the molecular data types

Examples

data(CCLEsmall)
mDataNames(CCLEsmall)

molecularProfiles molecularProfiles Generic

Description

Generic for molecularProfiles method

Usage

molecularProfiles(pSet, mDataType)

Arguments

pSet The PharmacoSet to retrieve molecular profiles from
mDataType the type of molecular data

Value

a data.frame with the experiment info

Examples

data(CCLEsmall)
molecularProfiles(CCLEsmall, "rna")
molecularProfiles<- Generic

Description
Generic for molecularProfiles replace method

Usage
molecularProfiles(object, mDataType) <- value

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

Value
Updated PharmacoSet

Examples
data(CCLEsmall)
molecularProfiles(CCLEsmall, "rna") <- molecularProfiles(CCLEsmall, "rna")

pertNumber     pertNumber Generic

Description
A generic for the pertNumber method

Usage
cartNumber(pSet)

Arguments
pSet        A PharmacoSet

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

Examples
data(CCLEsmall)
pertNumber(CCLEsmall)
Description

A generic for the `pertNumber` method

Usage

`pertNumber(object) <- value`

Arguments

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

Value

The updated `PharmacoSet`

Examples

data(CCLEsmall)
`pertNumber(CCLEsmall) <- pertNumber(CCLEsmall)`

PharmacoSet

**PharmacoSet constructor**

Description

A constructor that simplifies the process of creating `PharmacoSets`, 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 `PharmacoSet` methods. For a much more detailed instruction on creating `PharmacoSets`, please see the "CreatingPharmacoSet" vignette.

Usage

`PharmacoSet(name, molecularProfiles = list(), cell = data.frame(), drug = 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)), curationDrug = data.frame(), curationCell = data.frame(), curationTissue = data.frame(), datasetType = c("sensitivity", "perturbation", "both"), verify = TRUE)`
PharmacoSet-class

A Class to Contain PharmacoGenomic datasets together with their curations

Description

The PharmacoSet (PSet) class was developed to contain and organise large PharmacoGenomic datasets, and aid in their metanalysis. It was designed primarily to allow bioinformaticians and biologists to work with data at the level of genes, drugs and cell lines, providing a more naturally intuitive interface and simplifying analyses between several datasets. As such, it was designed to be flexible enough to hold datasets of two different natures while providing a common interface. The class can accommodate datasets containing both drug dose response data, as well as datasets containing genetic profiles of cell lines pre and post treatment with compounds, known respectively as sensitivity and perturbation datasets.
PharmacoSet-class

Usage

## S4 method for signature 'PharmacoSet'
cellInfo(pSet)

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

## S4 method for signature 'PharmacoSet'
drugInfo(pSet)

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

## S4 method for signature 'PharmacoSet'
phenoInfo(pSet, mDataType)

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

## S4 method for signature 'PharmacoSet'
molecularProfiles(pSet, mDataType)

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

## S4 method for signature 'PharmacoSet'
featureInfo(pSet, mDataType)

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

## S4 method for signature 'PharmacoSet'
sensitivityInfo(pSet)

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

## S4 method for signature 'PharmacoSet'
sensitivityProfiles(pSet)

## S4 replacement method for signature 'PharmacoSet,matrix'
sensitivityProfiles(object) <- value

## S4 method for signature 'PharmacoSet'
sensitivityMeasures(pSet)

## S4 method for signature 'PharmacoSet'
drugNames(pSet)
## S4 replacement method for signature 'PharmacoSet,character'

drugNames(object) <- value

## S4 method for signature 'PharmacoSet'
cellNames(pSet)

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

## S4 method for signature 'PharmacoSet'
fNames(pSet, mDataType)

## S4 method for signature 'PharmacoSet'
dateCreated(pSet)

## S4 method for signature 'PharmacoSet'
pSetName(pSet)

## S4 method for signature 'PharmacoSet'
pertNumber(pSet)

## S4 method for signature 'PharmacoSet'
sensNumber(pSet)

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

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

### Arguments

- **pSet**: A PharmacoSet object
- **object**: A PharmacoSet object
- **value**: A replacement value
- **mDataType**: A character with the type of molecular data to return/update

### Value

An object of the PharmacoSet class

### Methods (by generic)

- **cellInfo**: Returns the annotations for all the cell lines tested on in the PharmacoSet
- **cellInfo<-**: Update the cell line annotations
- **drugInfo**: Returns the annotations for all the drugs tested in the PharmacoSet
- **drugInfo<-**: Update the drug annotations
- **phenoInfo**: Return the experiment info from the given type of molecular data in PharmacoSet
- **phenoInfo<-**: Update the the given type of molecular data experiment info in the PharmacoSet
- **molecularProfiles**: Return the given type of molecular data from the PharmacoSet
PharmacoSet-class

- molecularProfiles<-: Update the given type of molecular data from the PharmacoSet
- 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
- sensitivityMeasures: Returns the available sensitivity profile summaries, for example, whether there are IC50 values available
- drugNames: Return the names of the drugs used in the PharmacoSet
- drugNames<-: Update the drug names used in the dataset
- 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 PharmacoSet was created
- pSetName: Return the name of the PharmacoSet
- 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 PharmacoSet, 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 containing the annotations for all the cell lines profiled in the data set, across all data types
- drug: A data.frame containing the annotations for all the drugs 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 pair
- perturbation: A list containing $n, a data.frame summarizing the available perturbation data, curation: A list containing mappings for $drug, cell, tissue names used in the data set to universal identifiers used between different PharmacoSet objects
- datasetType: A character string of 'sensitivity', 'perturbation', or both detailing what type of data can be found in the PharmacoSet, for proper processing of the data
Description

Generic for phenoInfo method

Usage

phenoInfo(pSet, mDataType)

Arguments

pSet The PharmacoSet to retrieve rna annotations from
mDataType the type of molecular data

Value

a data.frame with the experiment info

Examples

data(CCLEsmall)
phenoInfo(CCLEsmall, mDataType="rna")

Description

Generic for phenoInfo replace method

Usage

phenoInfo(object, mDataType) <- value

Arguments

object The PharmacoSet to retrieve molecular experiment annotations from
mDataType the type of molecular data
value a data.frame with the new experiment annotations

Value

The updated PharmacoSet
**pSetName**

### Examples

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

---

**pSetName**

**pSetName Generic**

### Description

A generic for the `pSetName` method

### Usage

```r
pSetName(pSet)
```

### Arguments

- `pSet` A `PharmacoSet`

### Value

The name of the `PharmacoSet`

### Examples

```r
data(CCLEsmall)
pSetName(CCLEsmall)
```

---

**sensitivityInfo**

**sensitivityInfo Generic**

### Description

Generic for `sensitivityInfo` method

### Usage

```r
sensitivityInfo(pSet)
```

### Arguments

- `pSet` The `PharmacoSet` to retrieve sensitivity experiment annotations from

### Value

A `data.frame` with the experiment info
Examples

data(CCLEsmall)
sensitivityInfo(CCLEsmall)

Description

A generic for the sensitivityInfo replacement method

Usage

sensitivityInfo(object) <- value

Arguments

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

Value

Updated PharmacoSet

Examples

data(CCLEsmall)
sensitivityInfo(CCLEsmall) <- sensitivityInfo(CCLEsmall)

sensitivityMeasures

Description

A generic for the sensitivityMeasures method

Usage

sensitivityMeasures(pSet)

Arguments

pSet
The PharmacoSet

Value

A character vector of all the available sensitivity measures
sensitivityProfiles

Examples

data(CCLEsmall)
sensitivityMeasures(CCLEsmall)

---

sensitivityProfiles  sensitivityProfiles Generic

Description

Generic for sensitivityProfiles method

Usage

sensitivityProfiles(pSet)

Arguments

- `pSet`: The PharmacoSet to retrieve sensitivity experiment data from

Value

A `data.frame` with the experiment info

Examples

data(CCLEsmall)
sensitivityProfiles(CCLEsmall)

---

sensitivityProfiles<-  sensitivityProfiles<- Generic

Description

A generic for the sensitivityProfiles replacement method

Usage

sensitivityProfiles(object) <- value

Arguments

- `object`: The PharmacoSet to update
- `value`: A `data.frame` with the new sensitivity profiles

Value

Updated PharmacoSet
Examples

```r
data(CCLEsmall)
sensitivityProfiles(CCLEsmall) <- sensitivityProfiles(CCLEsmall)
```

---

### Description

A generic for the sensNumber method

### Usage

```r
sensNumber(pSet)
```

### Arguments

- **pSet**: A PharmacoSet

### Value

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

### Examples

```r
data(CCLEsmall)
sensNumber(CCLEsmall)
```

---

### Description

A generic for the sensNumber method

### Usage

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

### Arguments

- **object**: A PharmacoSet
- **value**: A new data.frame with the number of sensitivity experiments per drug and cell line

### Value

The updated PharmacoSet
show,PharmacoSet-method

Examples

data(CCLEsmall)
sensNumber(CCLEsmall) <- sensNumber(CCLEsmall)

show,PharmacoSet-method

Show a PharmacoSet

Description

Show a PharmacoSet

Usage

## S4 method for signature 'PharmacoSet'
show(object)

Arguments

object PharmacoSet

Value

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

Examples

data(CCLEsmall)
CCLEsmall

show,PharmacoSig-method

Show PharmacoGx Signatures

Description

Show PharmacoGx Signatures

Usage

## S4 method for signature 'PharmacoSig'
show(object)

Arguments

object PharmacoSig
showSigAnnot

Value

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

Examples

```r
data(GDSCsmall)
drug.sensitivity <- drugSensitivitySig(GDSCsmall, mDataType="rna",
nthread=1, features = fNames(GDSCsmall, "rna"))[1])
drug.sensitivity

showSigAnnot(drug.sensitivity)
```

---

showSigAnnot  \hspace{1cm} Show the Annotations of a signature object

Description

This function prints out the information about the call used to compute the drug signatures, and the session info for the session in which the computation was done. Useful for determining the exact conditions used to generate signatures.

Usage

```r
showSigAnnot(Sigs)
```

Arguments

Sigs  
An object of the PharmacoSig Class, as returned by `drugPerturbationSig` or `drugSensitivitySig`

Value

Prints the PharmacoGx Signatures annotations to the output stream, and returns invisible NULL.

Examples

```r
data(GDSCsmall)
drug.sensitivity <- drugSensitivitySig(GDSCsmall, mDataType="rna",
nthread=1, features = fNames(GDSCsmall, "rna"))[1])
showSigAnnot(drug.sensitivity)
```
subsetTo

A function to subset a PharmacoSet to data containing only specified drugs, cells and genes

Description

This is the preferred method of subsetting a PharmacoSet. This function allows abstraction of the data to the level of biologically relevant objects: drugs, genes and cells. The function will automatically go through all of the combined data in the PharmacoSet and ensure only the requested drugs, genes and cell lines are found in any of the slots. This allows quickly picking out all the experiments for a drug or cell of interest, as well removes the need to keep track of all the metadata conventions between different datasets.

Usage

subsetTo(pSet, cells = NULL, drugs = NULL, molecular.data.cells = NULL, keep.controls = TRUE, ...)

Arguments

- **pSet**: A PharmacoSet to be subsetted
- **cells**: A list or vector of cell names as used in the dataset to which the object will be subsetted. If left blank, then all cells will be left in the dataset.
- **drugs**: A list or vector of drug names as used in the dataset to which the object will be subsetted. If left blank, then all drugs will be left in the dataset.
- **molecular.data.cells**: A list or vector of cell names to keep in the molecular data
- **keep.controls**: If the dataset has perturbation type experiments, should the controls be kept in the dataset? Defaults to true.
- ... Other arguments passed by other function within the package

Value

A PharmacoSet with only the selected drugs and cells

Examples

data(CCLEsmall)
CCLEdrugs <- drugNames(CCLEsmall)
CCLEcells <- cellNames(CCLEsmall)
PSet <- subsetTo(CCLEsmall, drugs = CCLEdrugs[1], cells = CCLEcells[1])
PSet
summarizeMolecularProfiles

Takes molecular data from a PharmacoSet, and summarises them into one entry per drug

Description

Given a PharmacoSet with molecular data, this function will summarize the data into one profile per cell line, using the chosen summary.stat. Note that this does not really make sense with perturbation type data, and will combine experiments and controls when doing the summary if run on a perturbation dataset.

Usage

summarizeMolecularProfiles(pSet, mDataType, cell.lines, features, summary.stat = c("mean", "median", "first", "last", "and", "or"), fill.missing = TRUE, summarize = TRUE, verbose = TRUE)

Arguments

- **pSet**: PharmacoSet The PharmacoSet to summarize
- **mDataType**: character which one of the molecular data types to use in the analysis, out of all the molecular data types available for the pset for example: rna, rnaseq, snp
- **cell.lines**: character The cell lines to be summarized. If any cell.line has no data, missing values will be created
- **features**: caracter A vector of the feature names to include in the summary
- **summary.stat**: character which summary method to use if there are repeated cell.lines? Choices are "mean", "median", "first", or "last" In case molecular data type is mutation or fusion "and" and "or" choices are available
- **fill.missing**: boolean should the missing cell lines not in the molecular data object be filled in with missing values?
- **summarize**: A flag which when set to FALSE disables summarizing and returns the data unchanged as a ExpressionSet
- **verbose**: boolean should messages be printed

Value

- **matrix**: An updated PharmacoSet with the molecular data summarized per cell line.

Examples

data(GDSCsmall)
GDSCsmall <- summarizeMolecularProfiles(GDSCsmall, mDataType = "rna", cell.lines=cellNames(GDSCsmall), summary.stat = "median", fill.missing = TRUE, verbose=TRUE)
GDSCsmall
summarizeSensitivityProfiles

Takes the sensitivity data from a PharmacoSet, and summarises them into a drug vs cell line table

Description

This function creates a table with cell lines as rows and drugs as columns, summarising the drug sensitivity data of a PharmacoSet into drug-cell line pairs.

Usage

summarizeSensitivityProfiles(pSet, sensitivity.measure = "auc_recomputed", cell.lines, drugs, summary.stat = c("mean", "median", "first", "last", "max", "min"), fill.missing = TRUE, verbose = TRUE)

Arguments

- **pSet** [PharmacoSet] The PharmacoSet from which to extract the data.
- **sensitivity.measure** [character] which sensitivity measure to use? Use the sensitivityMeasures function to find out what measures are available for each PSet.
- **cell.lines** character The cell lines to be summarized. If any cell lines has no data, it will be filled with missing values.
- **drugs** character The drugs to be summarized. If any drugs has no data, it will be filled with missing values.
- **summary.stat** character which summary method to use if there are repeated cell line-drug experiments? Choices are "mean", "median", "first", or "last".
- **fill.missing** boolean should the missing cell lines not in the molecular data object be filled in with missing values?
- **verbose** boolean Should the function print progress messages?

Value

matrix A matrix with cell lines going down the rows, drugs across the columns, with the selected sensitivity statistic for each pair.

Examples

data(GDSCsmall)
GDSCauc <- summarizeSensitivityProfiles(GDSCsmall, sensitivity.measure='auc_published')
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
```

[.PharmacoSet-method

Description

`.`

Usage

```r
## S4 method for signature 'PharmacoSet'
x[i, j, ..., drop = FALSE]
```

Arguments

- `x`: PSet
- `i`: Cell lines to keep in PSet
- `j`: Drugs to keep in PSet
- `...`: further arguments
- `drop`: A boolean flag of whether to drop single dimensions or not
Value

Returns the subsetted PSet
Index

*Topic datasets
- CCLEsmall, 4
- CMAPsmall, 7
- GDSCsmall, 25
- HDAC_genes, 26

PharmacoSet (PharmacoSet-class), 34

availablePSet, 4

CCLEsmall, 4
cellInfo, 5
cellInfo, PharmacoSet-method
(PharmacoSet-class), 34
cellInfo<-, 5
cellInfo<-, PharmacoSet, data.frame-method
(PharmacoSet-class), 34
cellNames, 6
cellNames, PharmacoSet-method
(PharmacoSet-class), 34
cellNames<-, 6
cellNames<-, PharmacoSet, character-method
(PharmacoSet-class), 34
checkPSetStructure, 7
CMAPsmall, 7
computeABC, 8
computeAmax, 9
computeAUC, 10
computeIC50, 11
computeICn (computeIC50), 11
computeSlope, 12
connectivityScore, 13
cosinePerm, 14
dateCreated, 15
dateCreated, PharmacoSet-method
(PharmacoSet-class), 34
dim, PharmacoSet-method, 15
downloadPertSig, 16
downloadPSet, 16
drugDoseResponseCurve, 17
drugInfo, 19
drugInfo, PharmacoSet-method
(PharmacoSet-class), 34
drugInfo<-, 19
drugInfo<-, PharmacoSet, data.frame-method
(PharmacoSet-class), 34
drugNames, 20
drugNames, PharmacoSet-method
(PharmacoSet-class), 34
drugNames<-, 20
drugNames<-, PharmacoSet, character-method
(PharmacoSet-class), 34
drugPerturbationSig, 21
drugSensitivitySig, 22
fNames, 24
fNames, PharmacoSet-method
(PharmacoSet-class), 34
GDSCsmall, 25
gwc, 25
HDAC_genes, 26
intersectList, 27
intersectPSet, 27
logLogisticRegression, 28
mcc, 30
mDataNames, 30
molecularProfiles, 31
molecularProfiles, PharmacoSet-method
(PharmacoSet-class), 34
molecularProfiles<-, 32
molecularProfiles<-, PharmacoSet, character, matrix-method
(PharmacoSet-class), 34
pertNumber, 32

50