Package 'PharmacoGx'

March 30, 2021

Type Package

Title Analysis of Large-Scale Pharmacogenomic Data

Version 2.2.4

Date 2020-11-23

Description Contains a set of functions to perform large-scale analysis of pharmaco-genomic data. These include the PharmacoSet object for storing the results of pharmacogenomic experiments, as well as a number of functions for computing common summaries of drug-dose response and correlating them with the molecular features in a cancer cell-line.

License Artistic-2.0

Suggests pander, rmarkdown, knitr, knitcitations, crayon, testthat, BiocGenerics

Encoding UTF-8

Imports Biobase, S4Vectors, SummarizedExperiment, BiocParallel, ggplot2, magicaxis, RColorBrewer, parallel, caTools, methods, downloader, stats, utils, graphics, grDevices, reshape2, jsonlite, data.table

Depends R (>= 3.6), CoreGx

RoxygenNote 7.1.1

VignetteBuilder knitr

biocViews GeneExpression, Pharmacogenetics, Pharmacogenomics, Software, Classification

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

Collate 'ComputeGR.R' 'GR.R' 'GWC.R' 'LogLogisticRegression.R' 'MatthewCor.R' 'SanityCheck.R' 'adaptiveMatthewCor.R' 'allGenerics.R' 'callingWaterfall.R' 'class-PharmacoSet.R' 'class-SignatureClass.R' 'computeABC.R' 'computeAUC.R' 'computeAUC_old.R' 'computeAmax.R' 'computeDSS.R' 'computeDrugSensitivity.R' 'computeIC50.R' 'computeICn.R' 'computeSlope.R' 'connectivityScore.R' 'cosinePerm.R' 'datasets.R' 'downloadPSet.R' 'downloadSignatures.R' 'drugDoseResponseCurve.R' 'drugPerturbationSig.R' 'filterNoisyCurves.R' 'geneDrugPerturbation.R' 'geneDrugSensitivity.R' 'getRawSensitivityMatrix.R' 'globals.R' 'intersectPSets.R' 'methods-[.R' 'methods-cellInfo.R'

'methods-cellNames.R' 'methods-dateCreated.R'
'methods-drugInfo.R' 'methods-drugNames.R'
'methods-drugSensitivitySig.R' 'methods-fNames.R'
'methods-featureInfo.R' 'methods-intersect.R'
'methods-mDataNames.R' 'methods-molecularProfiles.R'
'methods-molecularProfilesSlot.R' 'methods-name.R'
'methods-pertNumber.R' 'methods-phenoInfo.R'
'methods-sensNumber.R' 'methods-sensitivityInfo.R'
'methods-sensitivityMeasures.R' 'methods-sensitivityProfiles.R'
'methods-sensitivityRaw.R' 'methods-sensitivitySlot.R'
'methods-sensitivitySlotToLongTable.R' 'methods-subsetTo.R'
'methods-summarizeMolecularProfiles.R'
'methods-summarizeSensitivityProfiles.R' 'plotPSig.R'
'rankGeneDrugPerturbation.R' 'rankGeneDrugSensitivity.R'

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

git_branch RELEASE_3_12

git_last_commit 30c7992

git_last_commit_date 2021-02-26

Date/Publication 2021-03-29

Author Petr Smirnov [aut],
Zhaleh Safikhani [aut],
Christopher Eeles [aut],
Mark Freeman [aut],
Benjamin Haibe-Kains [aut, cre]

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

R topics documented:

amcc . . . 3
availablePSets . . . 4
callingWaterfall . . . 5
CCLEsmall . . . 6
cellInfo<-,PharmacoSet,data.frame-method . . . 6
checkPsetStructure . . . 7
CMAPsmall . . . 8
computeABC . . . 8
computeAmax . . . 9
computeAUC . . . 10
computeIC50 . . . 11
computeSlope . . . 13
connectivityScore . . . 14
cosinePerm . . . 15
dim,PharmacoSet-method . . . 16
downloadPertSig . . . 16
downloadPSet . . . 17
drugDoseResponseCurve . . . 18
drugInfo . . . 20
drugInfo<- . . . 21
drugNames . . . 21
drugNames<- . . . 22

drugPerturbationSig . . . 22
drugSensitivitySig,PharmacoSet-method . . . 23
filterNoisyCurves . . . 25
fNames<-,PharmacoSet,character,character-method . . . 26
GDSCsmall . . . 27
geneDrugSensitivity . . . 27
gwc . . . 28
HDAC_genes . . . 29
intersectPSet . . . 30
logLogisticRegression . . . 31
mcc . . . 32
PharmacoSet . . . 33
PharmacoSet-class . . . 34
PharmacoSig . . . 42
plot.PharmacoSig . . . 43
sensitivitySlotToLongTable,PharmacoSet-method . . . 44
show,PharmacoSet-method . . . 44
show,PharmacoSig-method . . . 45
showSigAnnot,PharmacoSig-method . . . 45
subsetTo,PharmacoSet-method . . . 46
summarizeSensitivityProfiles,PharmacoSet-method . . . 47
[,PharmacoSet,ANY,ANY,ANY-method . . . 48

Index **49**

`amcc` *Adaptive Matthews Correlation Coefficient*

Description

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

Usage

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

Arguments

`x`	Two paired vectors of values. Could be replicates of observations for the same experiments for example.
`y`	Two paired vectors of values. Could be replicates of observations for the same experiments for example.
`step.prct`	Instead of testing all possible splits of the data, it is possible to test steps of a percentage size of the total number of ranks in x/y. If this variable is 0, function defaults to testing all possible splits.

min.cat	The minimum number of members per category. Classifications with less members fitting into both categories will not be considered.
nperm	The number of perumatation to use for estimating significance. If 0, then no p-value is calculated.
nthread	Number of threads to parallize over. Both the AMCC calculation and the permutation testing is done in parallel.

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)
```

---

availablePSets *Return a table of PharmacoSets available for download*

Description

The function fetches a table of all PharmacoSets available for download. The table includes the dataset names, version information for the data in the PSet, the date of last update, the name of the PSet, and references for the data contained within, a DOI for the data, and a direct download link. Download can also be done using the downloadPSet function.

Usage

```
availablePSets(canonical = TRUE)
```

Arguments

canonical	['logical'] Should available PSets show only official PSets, or should user generated PSets be included?

Details

Much more information on the processing of the data and data provenance can be found at: www.orcestra.ca

Value

A data.frame with details about the available PharmacoSet objects

Examples

```
if (interactive()){
availablePSets()
}
```

`callingWaterfall` *Drug sensitivity calling using waterfall plots*

Description

1. Sensitivity calls were made using one of IC50, ActArea or Amax

Usage

```
callingWaterfall(
  x,
  type = c("IC50", "AUC", "AMAX"),
  intermediate.fold = c(4, 1.2, 1.2),
  cor.min.linear = 0.95,
  name = "Drug",
  plot = FALSE
)
```

Arguments

`x`	What type of object does this take in?
`type`	ic50: IC50 values in micro molar (positive values) actarea: Activity Area, that is area under the drug activity curve (positive values) amax: Activity at max concentration (positive values)
`intermediate.fold`	vector of fold changes used to define the intermediate sensitivities for ic50, actarea and amax respectively
`cor.min.linear`	`numeric` The minimum linear correlation to require?
`name`	`character` The name of the output to use in plot
`plot`	`boolean` Whether to plot the results

Details

2. Sort log IC50s (or ActArea or Amax) of the cell lines to generate a "waterfall distribution"

3. Identify cutoff:

3.1 If the waterfall distribution is non-linear (pearson cc to the linear fit <=0.95), estimate the major inflection point of the log IC50 curve as the point on the curve with the maximal distance to a line drawn between the start and end points of the distribution.

3.2 If the waterfall distribution appears linear (pearson cc to the linear fit > 0.95), then use the median IC50 instead.

4. Cell lines within a 4-fold IC50 (or within a 1.2-fold ActArea or 20 difference) difference centered around this inflection point are classified as being "intermediate", cell lines with lower IC50s (or ActArea/Amax values) than this range are defined as sensitive, and those with IC50s (or ActArea/Amax) higher than this range are called "insensitive".

5. Require at least x sensitive and x insensitive cell lines after applying these criteria (x=5 in our case).

Value

factor Containing the drug sensitivity status of each cellline.

Examples

```
# Dummy example
1 + 1
```

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

Barretina et al. The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity. Nature, 2012

---

cellInfo<-,PharmacoSet,data.frame-method
cellInfo<- Generic

Description

Generic for cellInfo replace method

Usage

```
## S4 replacement method for signature 'PharmacoSet,data.frame'
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)
```

---

checkPsetStructure *A function to verify the structure of a PharmacoSet*

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

```
checkPsetStructure(object, plotDist = FALSE, result.dir = ".")
```

Arguments

object	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 object 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 of the dataset was removed to make runnable example code. For the full dataset, please download using the downloadPSet function.

Usage

```
data(CMAPsmall)
```

Format

PharmacoSet object

References

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

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.

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 concentarion (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 concentarion (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-Viablility) 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

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)
```

---

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

```
computeIC50(
  concentration,
  viability,
  Hill_fit,
  conc_as_log = FALSE,
  viability_as_pct = TRUE,
  verbose = TRUE,
  trunc = TRUE
)

computeICn(
  concentration,
  viability,
  Hill_fit,
  n,
  conc_as_log = FALSE,
  viability_as_pct = TRUE,
  verbose = TRUE,
  trunc = 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, 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.

Value

a numeric value for the concentration of the nth precentile 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] Viablities corresponding to the concentration range passed as first parameter. The range of viablity values by definition should be between 0 and 100. But the viabalities greater than 100 and lower than 0 are also accepted.
`trunc`	[binary] A flag that identify if the viabality values should be truncated to be in the range of (0,100)
`verbose`	[boolean] If 'TRUE' the function will retrun warnings and other infomrative 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

```
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 'fgsea' and 'gwc'
`nperm`	`numeric`, how many permutations should be done to determine significance through permutation testing? The minimum is 100, default is 1e4.
`nthread`	`numeric`, how many cores to run parallel processing on.
`gwc.method`	`character`, should gwc use a weighted spearman or pearson statistic?
`...`	Additional arguments passed down to gsea and gwc functions

Value

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

References

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

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

Examples

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

---

cosinePerm *Cosine Permuations*

Description

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

Usage

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

Arguments

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

Value

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

Examples

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

`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	A character string, the name of the PharmacoSet for which to download signatures. The name should match the names returned in the 'PSet Name' column of 'availablePSets(canonical=FALSE)'.
saveDir	A character string with the folder path where the PharmacoSet should be saved. Defaults to "./PSets/Sigs/". Will create directory if it does not exist.
verbose	bool Should status messages be printed during download. Defaults to TRUE.

Value

An array type object contaning 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 PharacoSet class, enabling use with all the analysis tools provided in PharmacoGx. User availablePSets to discover which PSets are available.

Usage

```
downloadPSet(
  name,
  saveDir = tempdir(),
  pSetFileName = NULL,
  verbose = TRUE,
  timeout = 600
)
```

Arguments

name	Character string, the name of the PhamracoSet to download. Note that this is not the dataset name, but the PSet name - dataset names are not guaranteed to be unique.
saveDir	Character string with the folder path where the PharmacoSet should be saved. Defaults to 'tempdir()'. 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.
timeout	numeric Parameter that lets you extend R's default timeout for downloading large files. Defaults for this function to 600.

Value

A PSet object with the dataset

Warning

BREAKING CHANGES - this function now defaults to ‘tempdir()‘ as the download path! You must specify a saveDir or manually save the PSet if you want your download to persist past your current R session.‘

Examples

```
if (interactive()){
downloadPSet("CTRPv2", saveDir=file.path(".", "pSets"))
}
```

`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 gray area would show the common concentration range between pSets. User can ask for type of sensitivity measurment 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

```
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.
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 measurment 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 measerments 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 comncentrations between all the pSets would be used as plot xlim.
mycol	[vector] A vector with the same lenght of the pSets parameter which will determine the color of the curve for the pharmaco sets. If this parameter is missed default colors from Rcolorbrewer 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 logl 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

```
if (interactive()) {
# Manually enter the plot parameters
drugDoseResponseCurve(concentrations=list("Experiment 1"=c(.008, .04, .2, 1)),
 viabilities=list(c(100,50,30,1)), plot.type="Both")

# Generate a plot from one or more PSets
data(GDSCsmall)
drugDoseResponseCurve(drug="Doxorubicin", cellline="22RV", pSets=GDSCsmall)
}
```

---

drugInfo *drugInfo Generic*

Description

Generic for drugInfo getter method

Usage

```
drugInfo(object)
```

Arguments

object	The PharmacoSet to retrieve drug info from

Value

A [‘data.frame‘] of annotations for drugs in the object

drugInfo<- *drugInfo<- Generic*

Description

Generic for drugInfo replace method

Usage

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

Arguments

object	The [‘PharmacoSet‘] to replace drug info
value	A [‘data.frame‘] with the new drug annotations

Value

The [‘object‘] with updated drug annotations

drugNames *drugNames Generic*

Description

A generic for the drugNames method

Usage

```
drugNames(object)
```

Arguments

object	The [‘PharmacoSet‘] to return drug names from

Value

A [‘character‘] vector of drug names in the object

Examples

```
data(CCLEsmall)
drugNames(CCLEsmall)
```

drugNames<- *drugNames<- Generic*

Description

A generic for the drugNames replacement method

Usage

```
drugNames(object) <- value
```

Arguments

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

Value

The [‘object‘] with updated drug names

Examples

```
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

```
drugPerturbationSig(
  pSet,
  mDataType,
  drugs,
  cells,
  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.
cells	[character] a vector of cell names to use in computing 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 diagnostive 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

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

---

drugSensitivitySig,PharmacoSet-method

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

```
## S4 method for signature 'PharmacoSet'
drugSensitivitySig(
  object,
  mDataType,
  drugs,
  features,
  cells,
  tissues,
  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"),
  molecular.cutoff = NA,
  molecular.cutoff.direction = c("less", "greater"),
  nthread = 1,
  verbose = TRUE,
  ...
)
```

Arguments

`object`	`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.
`cells`	`character` allows choosing exactly which cell lines to include for the signature fitting. Should be a subset of cellNames(pSet)
`tissues`	`character` a vector of which tissue types to include in the signature fitting. Should be a subset of cellInfo(pSet)$tissueid
`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`	`character` What summary statistic should be used to summarize duplicates for cell line molecular profile measurements?
`sensitivity.summary.stat`	`character` 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`	`numeric` Allows the user to binarize the sensitivity data using this threshold.

standardize	character One of "SD", "rescale", or "none", for the form of standardization of the data to use. If "SD", the 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.
molecular.cutoff	Allows the user to binarize the sensitivity data using this threshold.
molecular.cutoff.direction	character One of "less" or "greater", allows to set direction of binarization.
nthread	numeric if multiple cores are available, how many cores should the computation be parallelized over?
verbose	logical 'TRUE' if the warnings and other informative message shoud 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

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

`filterNoisyCurves` *Viability measurements in dose-reponse curves must remain stable or decrease monotonically reflecting response to the drug being tested. filterNoisyCurves flags dose-response curves that strongly violate these assumptions.*

Description

Viability measurements in dose-reponse curves must remain stable or decrease monotonically reflecting response to the drug being tested. filterNoisyCurves flags dose-response curves that strongly violate these assumptions.

Usage

```
filterNoisyCurves(
  pSet,
  epsilon = 25,
  positive.cutoff.percent = 0.8,
  mean.viablity = 200,
  nthread = 1
)
```

Arguments

pSet	[PharmacoSet] a PharmacoSet object
epsilon	[numeric] a value indicates assumed threshold for the distance between to consecutive viability values on the drug-response curve in the analysis, out of dna, rna, rnaseq, snp, cnv
positive.cutoff.percent	[numeric] This value indicates that function may violate epsilon rule for how many points on drug-response curve
mean.viablity	[numeric] average expected viability value
nthread	[numeric] if multiple cores are available, how many cores should the computation be parallelized over?

Value

a list with two elements 'noisy' containing the rownames of the noisy curves, and 'ok' containing the rownames of the non-noisy curves

Examples

```
data(GDSCsmall)
filterNoisyCurves(GDSCsmall)
```

---

fNames<-,PharmacoSet,character,character-method
fNames<-

Description

Setter for the feature names of a [‘SummarizedExperiment‘] in the molecularProfiles slot

Usage

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

Arguments

object	The [‘PharmacoSet‘] object to update
mDataType	[‘character‘] The molecular data type to update
value	A [‘character‘] vector of the new cell names

Value

Updated [‘PharmacoSet‘]

Examples

```
data(CCLEsmall)
fNames(CCLEsmall, 'rna') <- 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 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

Garnett et al. Systematic identification of genomic markers of drug sensitivity in cancer cells. Nature, 2012.

geneDrugSensitivity *Calcualte The Gene Drug Sensitivity*

Description

TODO:: Write a description!

Usage

```
geneDrugSensitivity(
  x,
  type,
  batch,
  drugpheno,
  interaction.typexgene = FALSE,
  model = FALSE,
  standardize = c("SD", "rescale", "none"),
  verbose = FALSE
)
```

Arguments

x	A numeric vector of gene expression values
type	A vector of factors specifying the cell lines or type types
batch	A vector of factors specifying the batch
drugpheno	A numeric vector of drug sensitivity values (e.g., IC50 or AUC)
interaction.typexgene	boolean Should interaction between gene expression and cell/type type be computed? Default set to FALSE
model	boolean Should the full linear model be returned? Default set to FALSE
standardize	character One of 'SD', 'rescale' or 'none'
verbose	boolean Should the function display messages?

Value

A vector reporting the effect size (estimateof the coefficient of drug concentration), standard error (se), sample size (n), t statistic, and F statistics and its corresponding p-value.

---

gwc *GWC Score*

Description

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

Usage

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

Arguments

x1	numeric vector of effect sizes (e.g., fold change or t statitsics) for the first experiment
p1	numeric vector of p-values for each corresponding effect size for the first experiment
x2	numeric effect size (e.g., fold change or t statitsics) for the second experiment
p2	numeric vector of p-values for each corresponding effect size for the second experiment

method.cor	character string identifying if a pearson or spearman correlation should be used
nperm	numeric how many permutations should be done to determine
truncate.p	numeric Truncation value for extremely low p-values
...	Other passed down to internal functions

Value

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

Examples

```
data(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)
```

HDAC_genes *HDAC Gene Signature*

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

```
data(HDAC_genes)
```

Format

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

References

Glaser et al. Gene expression profiling of multiple histone deacetylase (HDAC) inhibitors: defining a common gene set produced by HDAC inhibition in T24 and MDA carcinoma cell lines. Molecular cancer therapeutics, 2003.

`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

```
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, cell lines, or concentrations
`cells`	a `character`vector of common cell lines between pSets. In case user is intersted on getting intersection on certain cell lines, they can provide their list of cell lines
`drugs`	a `character` vector of common drugs between pSets. In case user is intersted 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

A `list` of pSets, contatining 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` *Fits curves of the form E = E_inf + (1 - E_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_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.
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 enteries are paired in order of the vectors.

Usage

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

Arguments

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

Details

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

Value

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

Examples

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

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
)
```

Arguments

name	A character string detailing the name of the dataset
molecularProfiles	A list of SummarizedExperiment objects containing molecular profiles for each molecular data type.
cell	A data.frame containing the annotations for all the cell lines profiled in the data set, across all data types
drug	A data.frame containg the annotations for all the drugs
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	A data.frame summarizing the available sensitivity/perturbation data
perturbationN	A data.frame summarizing the available sensitivity/perturbation data
curationDrug, curationCell, curationTissue	A data.frame mapping the names for drugs, cells and tissues used in the data set to universal identifiers used between different PharmacoSet 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 PharmacoSet

Examples

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

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 accomidate datasets containing both drug dose response data, as well as datasets contaning genetic profiles of cell lines pre and post treatement with compounds, known respecitively as sensitivity and perturbation datasets.

Return cell line metadata from a object

Get the names of all cell-lines available in a ‘PharmacoSet‘ object

Update the names of cell lines available in a ‘PharmacoSet‘ object

Get the data that a ‘PharmacoSet‘ object was updated

A generic for the sensNumber method

Retrieve information from the

Retrieve information from the

Get the names of all drugs available in a specified ‘PharmacoSet‘ object

Set the drug names available in a PharmacoSet object

Return the feature names for the specified molecular data type

Get the molecular profile data for the specified molecular data type

Update the molecular profile data for the specified datatype in the specified pSet object

Returns the molecular data names for the ‘PharmacoSet‘ object

Get the molecular profile data for the specified molecular data type

Update the molecular profile data for the specified datatype in the specified pSet object

Return the name of the PharmacoSet object

Return the name of the ‘PharmacoSet‘ object

Get the perturbation number for a specified ‘PharmcoSet‘ object

Set the perturbation number for a specified ‘PharmacoSet‘ object

Get the phenotype information for a specified molecular datatype

Update the phenotype information for a specified molecular data type in a specified pSet object

Get the sensitivity numbers for a ‘PharmacoSet‘ object

Get the senstivity information DataFrame from a PharmacoSet object

Set the sensitivityInfo DataFrame in a PharmacoSet object

Get the types of sensitivity measurements from a object object

Get the types of sensitivity measurements available in a PharmacoSet object

Get the sensitivityProfiles data.frame from a PharmacoSet object

Usage

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

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

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

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

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

```
## S4 method for signature 'PharmacoSet'
drugInfo(object)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Arguments

object	A PharmacoSet to extract the raw sensitivity data from
value	A list of new sensitivity slot data for the pSet
mDataType	the type of molecular data
assay	[‘character‘] Name or index of the assay data to return
dimension	[‘character‘] Optional name of the dimension to extract, either ’cells’ or ’drugs’. Only used if the sensitivity slot contains a ‘LongTable‘ object instead of a ‘list‘.
...	Additional arguments to the rowData or colData ‘LongTable‘ methods. Only used if the sensitivity slot contains a ‘LongTable‘ object instead of a ‘list‘.

Value

An object of the PharmacoSet class

a `data.frame` with the cell annotations

A vector of the cell names used in the PharmacoSet

Updated ['PharmacoSet']

['character'] The date the 'PharmacoSet' was created

The updated `PharmacoSet`

A ['data.frame'] containg annotatations for all drugs in the object

A ['PharmacoSet'] with updated drug annotations in the '@drug' slot

A ['character'] vector containg the names of drugs in the pSet

The updated ['PharmacoSet'] object

A ['character'] vector of the feature names

A ['data.frame'] with the feature annotations for the specified 'mDataType'

Updated `PharmacoSet`

Vector of names of the molecular data types

a ['matrix'] of data for the given mDataType and assay

Updated ['PharmacoSet']

A ['list'] of 'SummarizedExperiment' objects, named by molecular data type

['character'] The name of the 'PharmacoSet'

The name of the PharmacoSet

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

The updated ['PharmacoSet']

a ['data.frame'] with the phenotype information for the specified molecular data type

The updated `PharmacoSet`

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

a ['DataFrame'] with the experiment info

Updated `PharmacoSet`

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

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

a `data.frame` with the experiment info

['invisible'] Updates the 'PharmacoSet' object.

['invisible'] Updates the 'PharmacoSet' object.

A `array` containing the raw sensitivity data

A copy of the `PharmacoSet` containing the updated sensitivty data

A `list` of the sensitivity slot contents

A copy of the `PharmacoSet` containing the updated sensitivty slot

Methods (by generic)

- `cellInfo`:
- `cellNames`: Return the cell names used in the dataset
- `cellNames<-`: Update the cell names used in the dataset
- `dateCreated`: Return the date the PharmacoSet was created
- `sensNumber<-`: Update the summary of available sensitivity experiments
- `drugInfo`: Returns the annotations for all the drugs tested in the PharmacoSet
- `drugInfo<-`: Update the drug annotations
- `drugNames`: Return the names of the drugs used in the PharmacoSet
- `drugNames<-`: Update the drug names used in the dataset
- `fNames`: Return the feature names used in the dataset
- `featureInfo`: Return the feature info for the given molecular datatype
- `featureInfo<-`: Replace the gene info for the molecular data
- `mDataNames`: Returns the names of molecular data types in a PharmacoSet
- `molecularProfiles`: Return the given type of molecular data from the PharmacoSet
- `molecularProfiles<-`: Update the given type of molecular data from the PharmacoSet
- `molecularProfiles<-`: Update the given type of molecular data from the PharmacoSet
- `molecularProfilesSlot`: Getter for the molecular profiles slot
- `molecularProfilesSlot<-`: Setter for the molecular profiles slot
- `name`: Return the name of the PharmacoSet
- `name<-`: Return the name of the PharmacoSet
- `pertNumber`: Return the summary of available perturbation experiments
- `pertNumber<-`: Update the summary of available perturbation experiments
- `phenoInfo`: Return the experiment info from the given type of molecular data in PharmacoSet
- `phenoInfo<-`: Update the given type of molecular data experiment info in the PharmacoSet
- `sensNumber`: Return the summary of available sensitivity experiments
- `sensitivityInfo`: Return the drug dose sensitivity experiment info
- `sensitivityInfo<-`: Update the metadata for the treatment response experiments in the sensitivity slot.
- `sensitivityMeasures`: returns the available sensitivity profile summaries, for example, whether there are IC50 values available
- `sensitivityMeasures<-`: returns the available sensitivity profile summaries, for example, whether there are IC50 values available
- `sensitivityProfiles`: Return the sensitivity profile summary values for the treatment response experiment data in the sensitivity slot.
- `sensitivityProfiles<-`: Update the sensitivity profiles for a 'PharmacoSet' object.
- `sensitivityProfiles<-`: Update the sensitivity profiles for a 'PharmacoSet' object.
- `sensitivityRaw`: Retrive the raw dose and viability data from a pSet
- `sensitivityRaw<-`: Update the raw dose and viability data in a pSet object
- `sensitivitySlot`: Retrieve the contents of the sensitivity slot
- `sensitivitySlot<-`: Set the raw dose and viability data for an pSet and return and updated copty

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 `SummarizedExperiment` type object for holding data for RNA, DNA, SNP and CNV measurements, 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` containg 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` containting `$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

Examples

```
data(CCLEsmall)
cellInf <- cellInfo(CCLEsmall)

data(CCLEsmall)
cellNames(CCLEsmall)

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

data(CCLEsmall)
dateCreated(CCLEsmall)

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

data(CCLEsmall)
drugInf <- drugInfo(CCLEsmall)

data(CCLEsmall)
drugInf <- drugInfo(CCLEsmall)

data(CCLEsmall)
drugNames(CCLEsmall)

data(CCLEsmall)
drugNames(CCLEsmall) <- drugNames(CCLEsmall)

data(CCLEsmall)
fNames(CCLEsmall, "rna")
```

```
data(CCLEsmall)
featInf <- featureInfo(CCLEsmall, "rna")

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

data(CCLEsmall)
mDataNames(CCLEsmall)

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

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

data(CCLEsmall)
molProfSlot <- molecularProfilesSlot(CCLEsmall)

data(CCLEsmall)
molecularProfilesSlot(CCLEsmall) <- molecularProfilesSlot(CCLEsmall)

data(CCLEsmall)
name(CCLEsmall)

data(CCLEsmall)
name(CCLEsmall) <- 'CCLEsmall'

data(CCLEsmall)
pertNumber(CCLEsmall)

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

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

data(CCLEsmall)
phenoInfo(CCLEsmall, mDataType='rna') <- phenoInfo(CCLEsmall, mDataType='rna')

data(CCLEsmall)
sensNumber(CCLEsmall)

data(CCLEsmall)
sensInf <- sensitivityInfo(CCLEsmall)

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

data(CCLEsmall)
sensMeas <- sensitivityMeasures(CCLEsmall)

data(CCLEsmall)
sensMeas <- sensitivityMeasures(CCLEsmall)

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

```
data(GDSCsmall)
sensitivityProfiles(GDSCsmall) <- sensitivityProfiles(GDSCsmall)

data(GDSCsmall)
sensitivityProfiles(GDSCsmall) <- sensitivityProfiles(GDSCsmall)

data(CCLEsmall)
sensitivityRaw(CCLEsmall)

data(CCLEsmall)
sensitivityRaw(CCLEsmall) <- sensitivityRaw(CCLEsmall)

data(CCLEsmall)
sensitivitySlot(CCLEsmall)

data(CCLEsmall)
sensitivitySlot(CCLEsmall) <- sensitivitySlot(CCLEsmall)
```

`PharmacoSig` *Contructor for the PharmacoSig S4 class*

Description

Contructor for the PharmacoSig S4 class

Usage

```
PharmacoSig(
  Data = array(NA, dim = c(0, 0, 0)),
  PSetName = "",
  DateCreated = date(),
  SigType = "sensitivity",
  SessionInfo = sessionInfo(),
  Call = "No Call Recorded",
  Arguments = list()
)
```

Arguments

`Data`	[‘array‘] of data to build the signature from
`PSetName`	[‘character‘] vector containing name of PSet, defaults to ”
`DateCreated`	[‘date‘] date the signature was created, defaults to ‘date()‘
`SigType`	[‘character‘] vector specifying whether the signature is sensitivity or perturbation, defaults to ’sensitivity’
`SessionInfo`	[‘sessionInfo‘] object as retuned by ‘sesssionInfo()‘ function, defaults to ‘sessionInfo()‘
`Call`	[‘character‘ or ‘call‘] specifying the constructor call used to make the object, defaults to ’No Call Recorded’
`Arguments`	[‘list‘] a list of additional arguments to the constructure

Value

A ['PharmacoSig'] object build from the provided signature data

---

`plot.PharmacoSig` *Plots a PharmacoSig object into a Volcano Plot*

Description

Given a PharmacoSig, this will plot a volcano plot, with parameters to set cutoffs for a significant effect size, p value, to pick multiple testing correction strategy, and to change point colors. Built on top of ggplot, it will return the plot object which can be easily customized as any other ggplot.

Usage

```
## S3 method for class 'PharmacoSig'
plot(
  x,
  adjust.method,
  drugs,
  features,
  effect_cutoff,
  signif_cutoff,
  color,
  ...
)
```

Arguments

`x`	['PharmacoSig'] a PharmacoSig object, result of drugSensitivitySig or drugPerturbationSig
`adjust.method`	['character'] or ['boolean'] either FALSE for no adjustment, or one of the methods implemented by p.adjust. Defaults to FALSE for no correction
`drugs`	['character'] a vector of drug names for which to plot the estimated associations with gene expression
`features`	['character'] a vector of features for which to plot the estimated associations with drug treatment
`effect_cutoff`	the cutoff to use for coloring significant effect sizes.
`signif_cutoff`	the cutoff to use for coloring significance by p value or adjusted p values. Not on log scale.
`color`	one color if no cutoffs set for plotting. A vector of colors otherwise used to color points the in three categories above.
`...`	additional arguments, not currently used, but left here for consistency with plot

Value

returns a ggplot object, which by default will be evaluated and the plot displayed, or can be saved to a variable for further customization by adding ggplot elements to the returned graph

Examples

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

`sensitivitySlotToLongTable,PharmacoSet-method`
Reconstruct the data in the @sensitivity slot list into a LongTable object.

Description

Reconstruct the data in the @sensitivity slot list into a LongTable object.

Usage

```
## S4 method for signature 'PharmacoSet'
sensitivitySlotToLongTable(object)
```

Arguments

`object`	A ['PharmacoSet'] with a list in the sensitivity slot containing items raw, profiles, info and n.

Value

A ['LongTable'] with the data from the sensitivity slot.

`show,PharmacoSet-method`
Show a PharamcoSet

Description

Show a PharamcoSet

Usage

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

Arguments

`object`	`PharmacoSet`

Value

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

@importFrom CoreGx show @importFrom methods callNextMethod

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

Value

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

Examples

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

---

showSigAnnot,PharmacoSig-method

Show the Annotations of a signature object

Description

This funtion 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

```
## S4 method for signature 'PharmacoSig'
showSigAnnot(object)
```

Arguments

object	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

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

---

subsetTo,PharmacoSet-method

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

---

Description

This is the prefered method of subsetting a PharmacoSet. This function allows abstraction of the data to the level of biologically relevant objects: drugs and cells. The function will automatically go through all of the combined data in the PharmacoSet and ensure only the requested drugs 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

```
## S4 method for signature 'PharmacoSet'
subsetTo(
  object,
  cells = NULL,
  drugs = NULL,
  molecular.data.cells = NULL,
  keep.controls = TRUE,
  ...
)
```

Arguments

object	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
```

---

summarizeSensitivityProfiles,PharmacoSet-method

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 senstitivity data of a PharmacoSet into drug-cell line pairs

Usage

```
## S4 method for signature 'PharmacoSet'
summarizeSensitivityProfiles(
  object,
  sensitivity.measure = "auc_recomputed",
  cell.lines,
  drugs,
  summary.stat = c("mean", "median", "first", "last", "max", "min"),
  fill.missing = TRUE,
  verbose = TRUE
)
```

Arguments

object [PharmacoSet] The PharmacoSet from which to extract the data

sensitivity.measure
[character] which sensitivity sensitivity.measure to use? Use the sensitivityMeasures function to find out what measures are available for each object.

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	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')
```

[,PharmacoSet,ANY,ANY,ANY-method
'['

Description

'['

Usage

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

Arguments

x	object
i	Cell lines to keep in object
j	Drugs to keep in object
...	further arguments
drop	A boolean flag of whether to drop single dimensions or not

Value

Returns the subsetted object

Examples

```
data(CCLEsmall)
CCLEsmall["WM1799", "Sorafenib"]
```

Index

∗ **datasets**
    CCLEsmall, 6
    CMAPsmall, 8
    GDSCsmall, 27
    HDAC_genes, 29
.PharmacoSet *(*PharmacoSet-class*)*, 34
[,PharmacoSet,ANY,ANY,ANY-method, 48

amcc, 3
availablePSets, 4

callingWaterfall, 5
CCLEsmall, 6
cellInfo,PharmacoSet-method
        *(*PharmacoSet-class*)*, 34
cellInfo<-,PharmacoSet,data.frame-method,
        6
cellNames,PharmacoSet-method
        *(*PharmacoSet-class*)*, 34
cellNames<-,PharmacoSet,character-method
        *(*PharmacoSet-class*)*, 34
checkPsetStructure, 7
CMAPsmall, 8
computeABC, 8
computeAmax, 9
computeAUC, 10
computeIC50, 11
computeICn *(*computeIC50*)*, 11
computeSlope, 13
connectivityScore, 14
cosinePerm, 15

dateCreated,PharmacoSet-method
        *(*PharmacoSet-class*)*, 34
dim,PharmacoSet-method, 16
downloadPertSig, 16
downloadPSet, 17
drugDoseResponseCurve, 18
drugInfo, 20
drugInfo,PharmacoSet-method
        *(*PharmacoSet-class*)*, 34
drugInfo<-, 21
drugInfo<-,PharmacoSet,data.frame-method
        *(*PharmacoSet-class*)*, 34
drugNames, 21
drugNames,PharmacoSet-method
        *(*PharmacoSet-class*)*, 34
drugNames<-, 22
drugNames<-,PharmacoSet,character-method
        *(*PharmacoSet-class*)*, 34
drugPerturbationSig, 22
drugSensitivitySig,PharmacoSet-method,
        23

featureInfo,PharmacoSet-method
        *(*PharmacoSet-class*)*, 34
featureInfo<-,PharmacoSet,character,DataFrame-method
        *(*PharmacoSet-class*)*, 34
filterNoisyCurves, 25
fNames,PharmacoSet,character-method
        *(*PharmacoSet-class*)*, 34
fNames<-,PharmacoSet,character,character-method,
        26

GDSCsmall, 27
geneDrugSensitivity, 27
gwc, 28

HDAC_genes, 29

intersectPSet, 30

logLogisticRegression, 31

mcc, 32
mDataNames,PharmacoSet-method
        *(*PharmacoSet-class*)*, 34
molecularProfiles,PharmacoSet-method
        *(*PharmacoSet-class*)*, 34
molecularProfiles<-,PharmacoSet,character,character,mat
        *(*PharmacoSet-class*)*, 34
molecularProfiles<-,PharmacoSet,character,missing,matri
        *(*PharmacoSet-class*)*, 34
molecularProfilesSlot,PharmacoSet-method
        *(*PharmacoSet-class*)*, 34
molecularProfilesSlot<-,PharmacoSet,list-method
        *(*PharmacoSet-class*)*, 34

name,PharmacoSet-method
        *(*PharmacoSet-class*)*, 34

name<-,PharmacoSet,character-method
(PharmacoSet-class), 34

pertNumber,PharmacoSet-method
(PharmacoSet-class), 34
pertNumber<-,PharmacoSet,array-method
(PharmacoSet-class), 34
PharmacoSet, 33
PharmacoSet-class, 34
PharmacoSig, 42
phenoInfo,PharmacoSet-method
(PharmacoSet-class), 34
phenoInfo<-,PharmacoSet,character,DataFrame-method
(PharmacoSet-class), 34
plot.PharmacoSig, 43

sensitivityInfo,PharmacoSet-method
(PharmacoSet-class), 34
sensitivityInfo<-,PharmacoSet,data.frame-method
(PharmacoSet-class), 34
sensitivityMeasures,PharmacoSet-method
(PharmacoSet-class), 34
sensitivityMeasures<-,PharmacoSet,character-method
(PharmacoSet-class), 34
sensitivityProfiles,PharmacoSet-method
(PharmacoSet-class), 34
sensitivityProfiles<-,PharmacoSet,data.frame-method
(PharmacoSet-class), 34
sensitivityProfiles<-,PharmacoSet,matrix-method
(PharmacoSet-class), 34
sensitivityRaw,PharmacoSet-method
(PharmacoSet-class), 34
sensitivityRaw<-,PharmacoSet,array-method
(PharmacoSet-class), 34
sensitivitySlot,PharmacoSet-method
(PharmacoSet-class), 34
sensitivitySlot<-,PharmacoSet,list-method
(PharmacoSet-class), 34
sensitivitySlotToLongTable,PharmacoSet-method,
44
sensNumber,PharmacoSet-method
(PharmacoSet-class), 34
sensNumber<-,PharmacoSet,matrix-method
(PharmacoSet-class), 34
show,PharmacoSet-method, 44
show,PharmacoSig-method, 45
showSigAnnot,PharmacoSig-method, 45
subsetTo,PharmacoSet-method, 46
summarizeSensitivityProfiles,PharmacoSet-method,
47