Package 'PharmacoGx'

May 18, 2024

Type Package

Title Analysis of Large-Scale Pharmacogenomic Data

Version 3.8.0

Date 2024-04-08

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 GPL (>= 3)

Suggests pander, rmarkdown, knitr, knitcitations, crayon, testthat,
markdown, BiocStyle, R.utils

Encoding UTF-8

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

Depends R (>= 3.6), CoreGx

LinkingTo Rcpp

Roxygen list(markdown = TRUE, r6=FALSE)

RoxygenNote 7.3.1

VignetteBuilder knitr

VignetteEngine knitr::rmarkdown

biocViews GeneExpression, Pharmacogenetics, Pharmacogenomics,
Software, Classification

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

Collate 'GR.R' 'GWC.R' 'PharmacoSet-class.R' 'PharmacoSet-accessors.R'
'PharmacoSet-utils.R' 'RcppExports.R' 'adaptiveMatthewCor.R'
'callingWaterfall.R' 'class-SignatureClass.R' 'computeABC.R'

'computeAUC.R' 'computeAUC_old.R' 'computeAmax.R'
'computeDSS.R' 'computeDrugSensitivity.R' 'computeGR.R'
'computeIC50.R' 'computeICn.R' 'computeSlope.R'
'computeSynergy.R' 'connectivityScore.R' 'cosinePerm.R'
'datasets.R' 'downloadPSet.R' 'downloadSignatures.R'
'drugDoseResponseCurve.R' 'drugPerturbationSig.R'
'filterNoisyCurves.R' 'geneDrugPerturbation.R'
'geneDrugSensitivity.R' 'geneDrugSensitivityPBCorr.R'
'geneDrugSensitivityPCorr.R' 'getRawSensitivityMatrix.R'
'globals.R' 'intersectPSets.R' 'logLogisticRegression.R'
'matthewCor.R' 'mergePSets.R' 'methods-[.R
'methods-drugSensitivitySig.R' 'methods-intersect.R'
'methods-subsetTo.R' 'methods-summarizeMolecularProfiles.R'
'methods-summarizeSensitivityProfiles.R' 'plotPSig.R'
'rankGeneDrugPerturbation.R' 'rankGeneDrugSensitivity.R'
'sanityCheck.R' 'updateObject-methods.R' 'zzz.R'

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

git_branch RELEASE_3_19

git_last_commit 7c2d523

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-17

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

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

Contents

.computeZIPdelta   *Vector-based version of [computeZIPdelta](#)*

Description

Following the calculation of ZIP delta score as in Appendix A3. See reference for details.

Usage

```
.computeZIPdelta(
  treatment1id,
  treatment2id,
  treatment1dose,
  treatment2dose,
  sampleid,
  HS_1,
  HS_2,
  EC50_1,
  EC50_2,
  E_inf_1,
  E_inf_2,
  combo_viability,
  ZIP = NULL,
  residual = "logcosh",
  nthread = 1L,
  show_Rsqr = FALSE
)
```

Arguments

treatment1id	character a vector of identifiers for treatment 1
treatment2id	character a vector of identifiers for treatment 2
treatment1dose	numeric a vector of concentrations for treatment 1
treatment2dose	numeric a vector of concentrations for treatment 2
sampleid	character Cell-line ID of a drug combination screening experiment.
HS_1	numeric Hill coefficient of treatment 1
HS_2	numeric Hill coefficient of treatment 2
EC50_1	numeric relative EC50 of treatment 1.
EC50_2	numeric relative EC50 of treatment 2.
E_inf_1	numeric viability produced by the maximum attainable effect of treatment 1.
E_inf_2	numeric viability produced by the maximum attainable effect of treatment 2.
combo_viability	
	numeric observed viability of the two treatments combined.

ZIP	numeric pre-computed ZIP reference values. If not provided, it will be computed during delta score calculation.
residual	character Method used to minimise residual in fitting curves. 3 methods available: c("logcosh", "normal", "Cauchy"). The default method is logcosh. It minimises the logarithmic hyperbolic cosine loss of the residuals and provides the fastest estimation among the three methods, with fitting quality in between normal and Cauchy; recommanded when fitting large-scale datasets. The other two methods minimise residuals by considering the truncated probability distribution (as in their names) for the residual. Cauchy provides the best fitting quality but also takes the longest to run.
nthread	integer Number of cores used to perform computation. Default 1.
show_Rsqr	logical Whether to show the 2-way curve fitting quality in the result. Default FALSE.

Value

numeric delta scores of every dose combinations for any given treatment combinations.

References

Yadav, B., Wennerberg, K., Aittokallio, T., & Tang, J. (2015). Searching for Drug Synergy in Complex Dose–Response Landscapes Using an Interaction Potency Model. Computational and Structural Biotechnology Journal, 13, 504–513. https://doi.org/10.1016/j.csbj.2015.09.001

Examples

```
## Not run:
## ZIP is optional. Will be recomputed if not provided.
combo_profiles <- CoreGx::buildComboProfiles(
     tre,
     c("HS", "EC50", "E_inf", "ZIP", "combo_viability"))
combo_profiles[,
      .computeZIPdelta(
           treatment1id = treatment1id,
           treatment2id = treatment2id,
           treatment1dose = treatment1dose,
           treatment2dose = treatment2dose,
           sampleid = sampleid,
           HS_1 = HS_1, HS_2 = HS_2,
           EC50_1 = EC50_1, EC50_2 = EC50_2,
           E_inf_1 = E_inf_1, E_inf_2 = E_inf_2,
           combo_viability = combo_viability,
           ZIP = ZIP,
           nthread = 4,
           show_Rsqr = TRUE
      )
    ] -> delta_scores

## End(Not run)
```

.summarizeSensProfiles

Summarize the sensitivity profiles when the sensitivity slot is a LongTable

---

Description

Summarize the sensitivity profiles when the sensitivity slot is a LongTable

Usage

```
.summarizeSensProfiles(
  object,
  sensitivity.measure = "auc_recomputed",
  profiles_assay = "profiles",
  treatment_col = "treatmentid",
  sample_col = "sampleid",
  cell.lines,
  drugs,
  summary.stat,
  fill.missing = TRUE
)
```

Value

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

---

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, 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

```
amcc(0.6^(1:5), 0.5^(1:5))
```

---

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(1) 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

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

1. Samples within a 4-fold IC50 (or within a 1.2-fold ActArea or 20% Amax difference) difference centered around this inflection point are classified as being "intermediate", samples 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".
2. Require at least x sensitive and x insensitive samples after applying these criteria (x=5 in our case).

Value

`factor` Containing the drug sensitivity status of each sample.

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

---

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	numeric is a vector of drug concentrations.
conc2	numeric is a vector of drug concentrations.
viability1	numeric 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	numeric 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	lis 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

Author(s)

Mark Freeman

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`	numeric is a vector of drug concentrations.
`viability`	numeric 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	numeric is a vector of drug concentrations.
viability	numeric 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)
```

---

computeBliss	*Compute Bliss Null References*

Description

Given two numeric containing viability of two monotherapy respectively, Compute Bliss null reference values for the expected response of the two treatments combined.

Usage

```
computeBliss(viability_1, viability_2)
```

Arguments

`viability_1`	numeric monotherapeutic response of treatment 1.
`viability_2`	numeric monotherapeutic response of treatment 2.

Value

`numeric` expected response of the two treatments combined under Bliss null assumption.

Examples

```
(bliss <- computeBliss(0.75, 0.65))
```

---

computeHSA   *Compute HSA Null References*

Description

Given two `numeric` containing viability of two monotherapy respectively, Compute highest single-agent (HSA) values as the expected response of the two treatments combined.

Usage

```
computeHSA(viability_1, viability_2)
```

Arguments

`viability_1`	numeric monotherapeutic response of treatment 1.
`viability_2`	numeric monotherapeutic response of treatment 2.

Value

`numeric` expected response of the two treatments combined using the highest response of the two (lower viability).

Examples

```
(hsa <- computeHSA(0.75, 0.65))
```

---

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   numeric is a vector of drug concentrations.

viability   numeric 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

numeric(1) The ICn of the Hill curve over the specified dose range.

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)
```

---

computeLoewe *Computes Loewe Null References*

Description

Predict the response of a treatment combination under the Loewe additive null assumption.

Usage

```
computeLoewe(
  treatment1dose,
  HS_1,
  E_inf_1,
  EC50_1,
  treatment2dose,
  HS_2,
  E_inf_2,
  EC50_2,
  tol = 0.1,
```

```
    lower_bound = 0,
    upper_bound = 1,
    verbose = FALSE
)
```

Arguments

treatment1dose	numeric a vector of concentrations for treatment 1
HS_1	numeric Hill coefficient of treatment 1
E_inf_1	numeric viability produced by the maximum attainable effect of treatment 1.
EC50_1	numeric relative EC50 of treatment 1.
treatment2dose	numeric a vector of concentrations for treatment 2
HS_2	numeric Hill coefficient of treatment 2
E_inf_2	numeric viability produced by the maximum attainable effect of treatment 2.
EC50_2	numeric relative EC50 of treatment 2.
tol	numeric Error tolerance for deviations from Loewe assumption. Loewe predictions with error higher than tol will be returned as NA. Deafult 0.1.
lower_bound	numeric Lowest possible value for Loewe expected viability. Default 0.
upper_bound	numeric Highest possible value for Loewe expected viability. Default 1.
verbose	logical whether to display warning messages. Default FALSE.

Value

numeric expected viability under Loewe additive null assumption.

Examples

```
## Not run:
tre |>
    endoaggregate(
        assay="combo_viability",
        Loewe = computeLoewe(
            treatment1dose=treatment1dose,
            treatment2dose=treatment2dose,
            HS_1=HS_1,
            HS_2=HS_2,
            E_inf_1=E_inf_1,
            E_inf_2=E_inf_2,
            EC50_1=EC50_1,
            EC50_2=EC50_2
        ),
        by = assayKeys(tre, "combo_viability")
    ) -> tre

## End(Not run)
```

---

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	numeric 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	numeric 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	logical(1) A flag that identify if the viabality values should be truncated to be in the range of (0,100)
verbose	logical(1) 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)
```

computeZIP   *Computes ZIP Null References*

Description

Predict the additive response of a treatment combination under the ZIP null assumption.

Usage

```
computeZIP(
  treatment1dose,
  HS_1,
  EC50_1,
  E_inf_1,
  treatment2dose,
  HS_2,
  EC50_2,
  E_inf_2
)
```

Arguments

treatment1dose	numeric a vector of concentrations for treatment 1
HS_1	numeric Hill coefficient of treatment 1
EC50_1	numeric relative EC50 of treatment 1.
E_inf_1	numeric viability produced by the maximum attainable effect of treatment 1. Default 0 by the original paper.
treatment2dose	numeric a vector of concentrations for treatment 2
HS_2	numeric Hill coefficient of treatment 2
EC50_2	numeric relative EC50 of treatment 2.
E_inf_2	numeric viability produced by maximum effect of treatment 2. Default 0 by the original paper.

Value

numeric expected viability under ZIP null assumption.

Examples

```
(zip <- computeZIP(
  treatment1dose = c(0.1, 0.01, 0.001),
  treatment2dose = c(1, 0.1, 0.01),
  HS_1 = rep(1, 3), HS_2 = rep(1.2, 3),
  EC50_1 = rep(0.01, 3), EC50_2 = rep(0.1, 3),
  E_inf_1 = rep(0, 3), E_inf_2 = rep(0.1, 3)
))
```

computeZIPdelta   *Generic to compute ZIP delta scores from an S4 object*

Description

Generic to compute ZIP delta scores from an S4 object

Usage

```
computeZIPdelta(object, ...)
```

Arguments

object   S4 An object to compute delta scores from.

...   Allow new arguments to this generic.

Value

Depends on the implemented method.

Examples

```
print("Generics shouldn't need examples?")
```

computeZIPdelta,TreatmentResponseExperiment-method
                    Compute ZIP delta score

Description

Following the calculation of ZIP delta score as in Appendix A3. See reference for details.

Compute ZIP delta score as described in the original paper.

Usage

```
## S4 method for signature 'TreatmentResponseExperiment'
computeZIPdelta(object, residual = "logcosh", nthread = 1L, show_Rsqr = FALSE)
```

Arguments

object	[TreatmentResponseExperiment](#) The TreatmentResponseExperiment from which to extract assays `mono_profile` and `combo_viability` to compute ZIP delta scores.
residual	`character` Method used to minimise residual in fitting curves. 3 methods available: `c("logcosh", "normal", "Cauchy")`. The default method is `logcosh`. It minimises the logarithmic hyperbolic cosine loss of the residuals and provides the fastest estimation among the three methods, with fitting quality in between `normal` and `Cauchy`; recommanded when fitting large-scale datasets. The other two methods minimise residuals by considering the truncated probability distribution (as in their names) for the residual. `Cauchy` provides the best fitting quality but also takes the longest to run.
nthread	`integer` Number of cores used to perform computation. Default 1.
show_Rsqr	`logical` Whether to show the 2-way curve fitting quality in the result. Default FALSE.

Value

[TreatmentResponseExperiment](#) with assay `combo_scores` containing `delta_scores`

References

Yadav, B., Wennerberg, K., Aittokallio, T., & Tang, J. (2015). Searching for Drug Synergy in Complex Dose–Response Landscapes Using an Interaction Potency Model. Computational and Structural Biotechnology Journal, 13, 504–513. https://doi.org/10.1016/j.csbj.2015.09.001

Examples

```
## Not run:
tre <- computeZIPdelta(tre, residual = "Cauchy", nthread = 2L)

## End(Not run)
```

---

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"),
  fileName,
  verbose = TRUE,
  ...,
  myfn
)
```

Arguments

name	A `character(1)` 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(1)` string with the folder path where the PharmacoSet should be saved. Defaults to `"./PSets/Sigs/"`. Will create directory if it does not exist.
fileName	`character(1)` What to name the downloaded file. Defaults to 'name_signature.RData' when excluded.
verbose	`logical(1)` Should downloader show detailed messages?
...	`pairlist` Force subsequent arguments to be named.
myfn	`character(1)` A deprecated version of `fileName`. Still works for now, but will be deprecated in future releases.

Value

An array type object contaning the signatures

Examples

```
## Not run:
    if (interactive()) downloadPertSig("CMAP_2016")

## End(Not run)
```

---

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

```
## Not run:
    if (interactive()) downloadPSet("CTRPv2_2015")

## End(Not run)
```

---

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,
  sample_col = "sampleid",
  treatment_col = "treatmentid"
)
```

Arguments

drug   character(1) A drug name for which the drug response curve should be plot-
                    ted. If the plot is desirable for more than one pharmaco set, A unique drug id
                    should be provided.

cellline   character(1) 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 per-
                    centage, and that E_inf passed in as decimal. Applies only to the viabilities
                    parameter.

trunc	logical(1) Should the viability values be truncated to lie in [0-100] before doing the fitting
legends.label	numeric 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	numeric 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	numeric 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	numeric 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	logical(1) Should warning messages about the data passed in be printed?
sample_col	character(1) The name of the column in the profiles assay that contains the sample IDs.
treatment_col	character(1) The name of the column in the profiles assay that contains the treatment IDs.

Value

Plots to the active graphics device and returns an 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="22RV1", pSets=GDSCsmall)
}
```

---

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	`logical(1)` 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,
        parallel.on = c("drug", "gene"),
        modeling.method = c("anova", "pearson"),
        inference.method = c("analytic", "resampling"),
        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 sampleNames(pSet)
tissues	character a vector of which tissue types to include in the signature fitting. Should be a subset of sampleInfo(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?

parallel.on   One of "gene" or "drug", chooses which level to parallelize computation (by gene, or by drug).

modeling.method

        One of "anova" or "pearson". If "anova", nested linear models (including and excluding the molecular feature) adjusted for are fit after the data is standardized, and ANOVA is used to estimate significance. If "pearson", partial correlation adjusted for tissue of origin are fit to the data, and a Pearson t-test (or permutation) test are used. Note that the difference is in whether standardization is done across the whole dataset (anova) or within each tissue (pearson), as well as the test applied.

inference.method

        Should "analytic" or "resampling" (permutation testing + bootstrap) inference be used to estimate significance. For permutation testing, QUICK-STOP is used to adaptively stop permutations. Resampling is currently only implemented for "pearson" modelling method.

verbose   logical 'TRUE' if the warnings and other informative message shoud be displayed

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

Value

array 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)
```

---

   effectToDose   *Inverse function of Hill equation*

Description

For the dose-response Hill equation of a drug defined by $E(x) = E_{inf} + \frac{1-E_{inf}}{1+(\frac{x}{EC50})(\frac{1}{HS})}$, that computes the response in viability from a dose in micromole for a drug, this function is the inverse function of the Hill curve that computes the dose required to produce a given response: $f^{-1}(E) = EC50(\frac{1-E}{E-E_{inf}})^{\frac{1}{HS}})$

Usage

```
effectToDose(viability, EC50, HS, E_inf, is_pct = FALSE)
```

Arguments

viability	numeric is a vector whose entries are the viability values in the range [0, 1] if is_pct is FALSE or [0, 100] if it is TRUE.
EC50	numeric is a vector of relative EC50 for drug-response equation.
HS	numeric Hill coefficient of the drug-response equation that represents the sigmoidity of the curve.
E_inf	numeric the maximum attanable effect of a drug when it is administered with a infinitely high concentration.
is_pct	logical whether both the input viabiliy and E_inf are given in percentage ([0, 100]) rather than decimal ([0, 1]). Default FALSE.

Value

numeric concentrations in micromoles required to produce viability in the corresponding entries.

Examples

```
dose <- effectToDose(viability = 80,
                     EC50 = 42,
                     HS = 1,
                     E_inf = 10,
                     is_pct = TRUE)
```

---

estimateProjParams   *Estimate the projected Hill coefficient, efficacy, and potency*

Description

Estimate the projected shape parameter HS, efficacy E_inf and potency EC50 in the new dose-response curve of a drug after adding another drug to it by fitting a 2-parameter dose-response curve.

Usage

```
estimateProjParams(
  dose_to,
  combo_viability,
  dose_add,
  EC50_add,
  HS_add,
  E_inf_add = 0,
  residual = c("logcosh", "normal", "Cauchy"),
  show_Rsqr = TRUE,
  conc_as_log = FALSE,
  optim_only = FALSE,
  loss_args = list()
)
```

Arguments

`dose_to`	numeric a vector of concentrations of the drug being added to
`combo_viability`	
	numeric observed viability of two treatments; target for fitting curve.
`dose_add`	numeric a vector of concentrations of the drug added.
`EC50_add`	numeric relative EC50 of the drug added.
`HS_add`	numeric Hill coefficient of the drug added.
`E_inf_add`	numeric Efficacy of the drug added.
`residual`	character Method used to minimise residual in fitting curves. 3 methods available: `logcosh`, `normal`, `Cauchy`. The default method is `logcosh`. It minimises the logarithmic hyperbolic cosine loss of the residuals and provides the fastest estimation among the three methods, with fitting quality in between `normal` and `Cauchy`; recommanded when fitting large-scale datasets. The other two methods minimise residuals by considering the truncated probability distribution (as in their names) for the residual. `Cauchy` provides the best fitting quality but also takes the longest to run.
`show_Rsqr`	logical whether to show goodness-of-fit value in the result.
`conc_as_log`	logical indicates whether input concentrations are in log10 scale.
`optim_only`	logical(1) Should the fall back methods when optim fails
`loss_args`	list Additional argument to the `loss` function. These get passed to losss via `do.call` analagously to using ....

Value

list * `HS_proj`: Projected Hill coefficient after adding a drug * `E_inf_proj`: Projected efficacy after adding a drug * `EC50_proj`: Projected potency after adding a drug * `E_ninf_proj`: Projected baseline viability by the added drug * `Rsqr`: if `show_Rsqr` is TRUE, it will include the R squared value indicating the quality of the fit in the result.

References

Motulsky, H., & Christopoulos, A. (2004). Fitting dose-response curves. In Fitting models to biological data using linear and nonlinear regression: A practical guide to curve fitting. Oxford University Press.

`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)
```

---

fitTwowayZIP	*Two-way fitting for projected dose-response curve.*

Description

Fit projected dose-response curves with E_min as the viability of the treatment being added to the other treament at a fixed dose.

Usage

```
fitTwowayZIP(
  combo_profiles,
  residual = "logcosh",
  show_Rsqr = TRUE,
  nthread = 1L,
  optim_only = TRUE,
  loss_args = list()
)
```

Arguments

combo_profiles	[data.table](#) contains three parameters of dose-response curves for each single agent in a drug comnbination, and the observed viability of two treatments combined.
residual	`character` Method used to minimise residual in fitting curves. 3 methods available: `c("logcosh", "normal", "Cauchy")`. The default method is `logcosh`. It minimises the logarithmic hyperbolic cosine loss of the residuals and provides the fastest estimation among the three methods, with fitting quality in between `normal` and `Cauchy`; recommanded when fitting large-scale datasets. The other two methods minimise residuals by considering the truncated probability distribution (as in their names) for the residual. `Cauchy` provides the best fitting quality but also takes the longest to run.
show_Rsqr	`logical` whether to show goodness-of-fit value in the result.
nthread	`integer` Number of cores used to perform computation. Default 1.
optim_only	`logical(1)` Should the fall back methods when optim fails
loss_args	`list` Additional argument to the `loss` function. These get passed to `losss` via `do.call` analagously to using `....`.

Value

[data.table](#) contains parameters of projected dose-response curves for adding one treatment to the other.

References

Yadav, B., Wennerberg, K., Aittokallio, T., & Tang, J. (2015). Searching for Drug Synergy in Complex Dose–Response Landscapes Using an Interaction Potency Model. Computational and Structural Biotechnology Journal, 13, 504–513. https://doi.org/10.1016/j.csbj.2015.09.001

Examples

```
## Not run:
combo_profiles <- CoreGx::buildComboProfiles(tre, c("HS", "EC50", "E_inf", "viability"))
combo_twowayFit <- fitTwowayZIP(combo_profiles)

## End(Not run)
```

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

Examples

```
print("TODO::")
```

---

geneDrugSensitivityPBCorr

Calculate The Gene Drug Sensitivity

Description

This version of the function uses a partial correlation instead of standardized linear models, for discrete predictive features Requires at least 3 observations per group.

Usage

```
geneDrugSensitivityPBCorr(
  x,
  type,
  batch,
  drugpheno,
  test = c("resampling", "analytic"),
  req_alpha = 0.05,
  nBoot = 1000,
  conf.level = 0.95,
  max_perm = getOption("PharmacoGx_Max_Perm", ceiling(1/req_alpha * 100)),
  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)
test	A `character` string indicating whether resampling or analytic based tests should be used
req_alpha	numeric, number of permutations for p value calculation
nBoot	numeric, number of bootstrap resamplings for confidence interval estimation
conf.level	numeric, between 0 and 1. Size of the confidence interval required
max_perm	numeric the maximum number of permutations that QUICKSTOP can do before giving up and returning NA. Can be set globally by setting the option "PharmacoGx_Max_Perm", or left at the default of `ceiling(1/req_alpha*100)`.
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.

Examples

```
print("TODO::")
```

---

geneDrugSensitivityPCorr

Calculate The Gene Drug Sensitivity

Description

This version of the function uses a partial correlation instead of standardized linear models.

Usage

```
geneDrugSensitivityPCorr(
  x,
  type,
  batch,
  drugpheno,
  test = c("resampling", "analytic"),
  req_alpha = 0.05,
  nBoot = 1000,
  conf.level = 0.95,
  max_perm = getOption("PharmacoGx_Max_Perm", ceiling(1/req_alpha * 100)),
  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)
test	A `character` string indicating whether resampling or analytic based tests should be used
req_alpha	numeric, number of permutations for p value calculation
nBoot	numeric, number of bootstrap resamplings for confidence interval estimation
conf.level	numeric, between 0 and 1. Size of the confidence interval required
max_perm	numeric the maximum number of permutations that QUICKSTOP can do before giving up and returning NA. Can be set globally by setting the option "PharmacoGx_Max_Perm", or left at the default of `ceiling(1/req_alpha*100)`.
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.

Examples

```
print("TODO::")
```

---

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.

---

hillCurve	*4-Parameter Hill Equation for Stimuli-Response Curves*

Description

Sigmoidal function which fits well to many stimuli-response associations observed in biology and pharmacology. In the context of PharmacoGx we are using it to model treatment-response assocations in cancer cell lines.

Usage

```
hillCurve(dose, HS, EC50, E_inf, E_ninf)
```

Arguments

dose	numeric() A vector of log10(dose) values (or equivalent for the stimuli being modelleled).
HS	numeric(1) Hill coefficient (n) which defines the slope of the dose-response curve at the mid-point. This parameter describes the degree of sigmoidicity of the Hill curve. HS = 1 corresponds to the rectangular hyperbola in dose-response space.
EC50	numeric(1) The dose required to produce 50% of the theoretically maximal response in the system, E_inf. Should be in the same units as dose!
E_inf	numeric(1) Theoretical maximal response (minimal viability) in the system as a proportion in the range \[0, 1\]. Note that since we are predicting viability (percent of cells alive after treatment) instead of response, this value should be low (i.e., more cell killing).
E_ninf	numeric(1) Theoretical minimum response (basal response). Defaults to 1, which should be the case for most viability experiments since we expect no cell killing to occur prior to applying a treatment.

Value

numeric() Vector of predicted viabilities for the Hill curve defined by EC50, E_inf, E_ninf and HS for each supplied value of dose.

Author(s)

Feifei Li Petr Smirnov Christopher Eeles

References

Gesztelyi, R., Zsuga, J., Kemeny-Beke, A., Varga, B., Juhasz, B., & Tosaki, A. (2012). The Hill equation and the origin of quantitative pharmacology. Archive for History of Exact Sciences, 66(4), 427–438. https://doi.org/10.1007/s00407-012-0098-5

Motulsky, H., & Christopoulos, A. (2004). Fitting models to biological data using linear and non-linear regression: A practical guide to curve fitting. Oxford University Press. See Chapter 41.

Examples

```
(viability <- hillCurve(
  dose=c(0.1, 0.01, 0.001),
  HS=1.1,
  EC50=0.01,
  E_ninf=1,
  E_inf=0
))
```

---

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
```

---

loeweCI   *Loewe Additive Combination Index (CI)*

Description

Computes the Loewe additive combination index (CI) from its definition $CI = \frac{x_1}{f_1^{-1}(E)} + \frac{x_2}{f_2^{-1}(E)}$

Usage

```
loeweCI(
  viability,
  treatment1dose,
  HS_1,
  E_inf_1,
  EC50_1,
  treatment2dose,
  HS_2,
  E_inf_2,
  EC50_2,
  is_pct = FALSE
)
```

Arguments

`viability`	numeric is a vector whose entries are the viability values in the range [0, 1].
`treatment1dose`	numeric a vector of concentrations for treatment 1
`HS_1`	numeric Hill coefficient of treatment 1
`E_inf_1`	numeric the maximum attainable effect of treatment 1.
`EC50_1`	numeric relative EC50 of treatment 1.
`treatment2dose`	numeric a vector of concentrations for treatment 2
`HS_2`	numeric Hill coefficient of treatment 2
`E_inf_2`	numeric the maximum attainable effect of treatment 2.
`EC50_2`	numeric relative EC50 of treatment 2.
`is_pct`	logical whether both the input viabiliy and E_inf are given in percentage ([0, 100]) rather than decimal ([0, 1]). Default FALSE.

Value

CI under Loewe additive definition

Examples

```
## Not run:
tre |>
    endoaggregate(
        assay="combo_viability",
        Loewe = PharmacoGx::computeLoewe(
            treatment1dose = treatment1dose,
            treatment2dose = treatment2dose,
            HS_1 = HS_1,
            HS_2 = HS_2,
            E_inf_1 = E_inf_1,
            E_inf_2 = E_inf_2,
            EC50_1 = EC50_1,
            EC50_2 = EC50_2
        ),
        by = assayKeys(tre, "combo_viability")
    ) -> tre

## End(Not run)
```

`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, 5),
  step = 0.5/density,
  precision = 1e-04,
  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 = TRUE
)
```

Arguments

conc	numeric is a vector of drug concentrations.
viability	numeric 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	numeric 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	numeric 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	numeric 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	numeric 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 list containing estimates for HS, E_inf, and EC50. It is annotated with the attribute Rsquared, which is the R^2 of the fit. Note that this is calculated using the values actually used for the fit, after truncation and any transform applied. With truncation, this will be different from the R^2 compared to the variance of the raw data. This also means that if all points were truncated down or up, there is no variance in the data, and the R^2 may be NaN.

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, 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)
```

---

partialCorQUICKSTOP   *QUICKSTOP significance testing for partial correlation*

Description

This function will test whether the observed partial correlation is significant at a level of req_alpha, doing up to MaxIter permutations. Currently, it supports only grouping by discrete categories when calculating a partial correlation. Currenlty, only does two sided tests.

Usage

```
partialCorQUICKSTOP(
  pin_x,
  pin_y,
  pobsCor,
  pGroupFactor,
  pGroupSize,
  pnumGroup,
  pMaxIter,
  pn,
  preq_alpha,
```

```
    ptolerance_par,
    plog_decision_boundary,
    pseed
)
```

Arguments

`pin_x`	one of the two vectors to correlate.
`pin_y`	the other vector to calculate
`pobsCor`	the observed (partial) correlation between these varaiables
`pGroupFactor`	an integer vector labeling group membership, to correct for in the partial correlation. NEEDS TO BE ZERO BASED!
`pGroupSize`	an integer vector of size length(unique(pGroupFactor)), counting the number of members of each group (basically table(pGroupFactor)) as integer vector
`pnumGroup`	how many groups are there (len(pGroupSize))
`pMaxIter`	maximum number of iterations to do, as a REAL NUMBER
`pn`	length of x and y, as a REAL NUMBER
`preq_alpha`	the required alpha for significance
`ptolerance_par`	the tolerance region for quickstop. Suggested to be 1/100th of req_alpha'
`plog_decision_boundary`	
	log (base e) of 1/probability of incorrectly calling significance, as per quickstop paper (used to determine the log-odds)
`pseed`	A numeric vector of length 2, used to seed the internal xoroshiro128+ 1.0 random number generator. Note that currently, these values get modified per call, so pass in a copy if you wish to keep a seed for running same simulation twice

Value

a double vector of length 4, entry 1 is either 0, 1 (for TRUE/FALSE) or NA_REAL_ for significance determination NA_REAL_ is returned when the MaxIter were reached before a decision is made. Usually, this occurs when the real p value is close to, or falls within the tolerance region of (req_alpha, req_alpha+tolerance_par). Entry 2 is the current p value estimate. entry 3 is the total number of iterations performed. Entry 4 is the number of time a permuted value was larger in absolute value than the observed cor.

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(),
  sample = data.frame(),
  treatment = 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)),
  curationTreatment = data.frame(),
  curationSample = 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.

sample
: A data.frame containing the annotations for all the sample profiled in the data set, across all data types. Must contain the mandatory sampleid column which uniquely identifies each sample in the object.

treatment
: A data.frame containing annotations for all treatments profiled in the dataset. Must contain the mandatory treatmentid column which uniquely identifies each treatment in the object.

sensitivityInfo
: A data.frame containing the information for the sensitivity experiments. Must contain a 'sampleid' column with unique identifiers to each sample, matching the sample object and a 'treatmentid' columns with unique indenifiers for each treatment, matching the treatment object.

sensitivityRaw
: A 3 Dimensional array contaning the raw drug dose response data for the sensitivity experiments

sensitivityProfiles
: data.frame containing drug sensitivity profile statistics such as IC50 and AUC

sensitivityN, perturbationN
: A data.frame summarizing the available sensitivity/perturbation data

curationSample, curationTissue, curationTreatment
: A data.frame mapping the names for samples, tissues and treatments used in the data set to universal identifiers used between different CoreSet objects

datasetType	A character(1) string of 'sensitivity', 'preturbation', or 'both' detailing what type of data can be found in the `CoreSet`, for proper processing of the data
verify	logical(1)Should the function verify the CoreSet and print out any errors it finds after construction?
...	Catch and parse any renamed constructor arguments.

Value

An object of class `PharmacoSet`

Examples

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

PharmacoSet-accessors   *Accessing and modifying information in a* PharmacoSet

---

Description

Documentation for the various setters and getters which allow manipulation of data in the slots of a `PharmacoSet` object.

Usage

```
drugInfo(...)

drugInfo(...) <- value

drugNames(...)

drugNames(...) <- value

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

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

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

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

## S4 method for signature 'PharmacoSet'
```

```
name(object)

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

## S4 method for signature 'PharmacoSet,character'
```

```
fNames(object, mDataType)

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

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

## S4 replacement method for signature 'PharmacoSet'
mDataNames(object) <- value

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

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

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

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

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

## S4 replacement method for signature 'PharmacoSet,list_OR_LongTable'
treatmentResponse(object) <- value

## S4 method for signature 'PharmacoSet'
```

```
sensNumber(object)

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

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

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

Arguments

`...`	See details.
`value`	See details.
`object`	A PharmacoSet object.
`mDataType`	`character(1)` The name of a molecular datatype to access from the `molecularProfiles` of a `PharmacoSet` object.
`assay`	`character(1)` A valid assay name in the `SummarizedExperiment` of `@molecularProfiles` of a PharmacoSet object for data type `mDataType`.
`dimension`	See details.

Details

treatmentInfo: `data.frame` Metadata for all treatments in a `PharmacoSet` object. Arguments:

- object: PharmacoSet An object to retrieve treatment metadata from.

treatmentInfo<-: PharmacoSet object with updated treatment metadata. object. Arguments:

- object: PharmacoSet An object to set treatment metadata for.
- value: `data.frame` A new table of treatment metadata for `object`.

treatmentNames: `character` Names for all treatments in a `PharmacoSet` object. Arguments:

- object: PharmacoSet An object to retrieve treatment names from.

treatmentNames<-: PharmacoSet Object with updates treatment names. object. Arguments:

- object: PharmacoSet An object to set treatment names from.
- value: `character` A character vector of updated treatment names.

@annotation:

annotation: A `list` of PharmacoSet annotations with items: 'name', the name of the object; 'dateCreated', date the object was created; 'sessionInfo', the `sessionInfo()` when the object was created; 'call', the R constructor call; and 'version', the object version.

annotation<-: Setter method for the annotation slot. Arguments:

- value: a `list` of annotations to update the PharmacoSet with.

@dateCreated:

dateCreated: `character(1)` The date the `PharmacoSet` object was created, as returned by the `date()` function.

dateCreated<-: Update the 'dateCreated' item in the `annotation` slot of a `PharmacoSet` object. Arguments:

- value: A `character(1)` vector, as returned by the `date()` function.

name: `character(1)` The name of the `PharmacoSet`, retreived from the `@annotation` slot.

name<-: Update the `@annotation$name` value in a `PharmacoSet` object.

- value: `character(1)` The name of the `PharmacoSet` object.

cellInfo: `data.frame` Metadata for all sample in a `PharmacoSet` object.

sampleInfo<-: assign updated sample annotations to the `PharmacoSet` object. Arguments:

- value: a `data.frame` object.

sampleNames: `character` Retrieve the rownames of the `data.frame` in the `sample` slot from a `PharmacoSet` object.

sampleNames<-: assign new rownames to the sampleInfo `data.frame` for a PharmacoSet object. Arguments:

- value: `character` vector of rownames for the `sampleInfo(object)` `data.frame`.

@curation:

curation: A `list` of curated mappings between identifiers in the PharmacoSet object and the original data publication. Contains three `data.frame`s, 'cell' with cell-line ids and 'tissue' with tissue ids and 'drug' with drug ids.

curation<-: Update the `curation` slot of a PharmacoSet object. Arugments:

- value: A `list` of `data.frame`s, one for each type of curated identifier. For a `PharmacoSet` object the slot should contain tissue, cell-line and drug id `data.frame`s.

datasetType slot:

datasetType: `character(1)` The type treatment response in the `sensitivity` slot. Valid values are 'sensitivity', 'perturbation' or 'both'.

datasetType<-: Update the datasetType slot of a PharmacoSet object. Arguments:

- value: A `character(1)` vector with one of 'sensitivity', 'perturbation' or 'both'

@molecularProfiles:

molecularProfiles: `matrix()` Retrieve an assay in a `SummarizedExperiment` from the `molecularProfiles` slot of a `PharmacoSet` object with the specified mDataType. Valid mDataType arguments can be found with `mDataNames(object)`. Exclude mDataType and assay to access the entire slot. Arguments:

- assay: Optional `character(1)` vector specifying an assay in the `SummarizedExperiment` of the `molecularProfiles` slot of the `PharmacoSet` object for the specified `mDataType`. If excluded, defaults to modifying the first assay in the `SummarizedExperiment` for the given `mDataType`.

molecularProfiles<-: Update an assay in a `SummarizedExperiment` from the `molecularProfiles` slot of a PharmacoSet object with the specified `mDataType`. Valid `mDataType` arguments can be found with `mDataNames(object)`. Omit `mDataType` and `assay` to update the slot.

- assay: Optional `character(1)` vector specifying an assay in the `SummarizedExperiment` of the `molecularProfiles` slot of the `PharmacoSet` object for the specified `mDataType`. If excluded, defaults to modifying the first assay in the `SummarizedExperiment` for the given `mDataType`.
- value: A `matrix` of values to assign to the `assay` slot of the `SummarizedExperiment` for the selected `mDataType`. The rownames and column names must match the associated `SummarizedExperiment`.

featureInfo: Retrieve a `DataFrame` of feature metadata for the specified `mDataType` from the `molecularProfiles` slot of a `PharmacoSet` object. More specifically, retrieve the `@rowData` slot from the `SummarizedExperiment` from the `@molecularProfiles` of a `PharmacoSet` object with the name `mDataType`.

featureInfo<-: Update the `featureInfo(object, mDataType)` `DataFrame` with new feature metadata. Arguments:

- value: A `data.frame` or `DataFrame` with updated feature metadata for the specified molecular profile in the `molecularProfiles` slot of a `PharmacoSet` object.

phenoInfo: Return the `@colData` slot from the `SummarizedExperiment` of `mDataType`, containing sample-level metadata, from a `PharmacoSet` object.

phenoInfo<-: Update the `@colData` slot of the `SummarizedExperiment` of `mDataType` in the `@molecularProfiles` slot of a `PharmacoSet` object. This updates the sample-level metadata in-place.

- value: A `data.frame` or `DataFrame` object where rows are samples and columns are sample metadata.

fNames: `character()` The features names from the `rowData` slot of a `SummarizedExperiment` of `mDataType` within a `PharmacoSet` object.

fNames: Updates the rownames of the feature metadata (i.e., `rowData`) for a `SummarizedExperiment` of `mDataType` within a `PharmacoSet` object.

- value: `character()` A character vector of new features names for the `rowData` of the `SummarizedExperiment` of `mDataType` in the `@molecularProfiles` slot of a `PharmacoSet` object. Must be the same length as `nrow(featureInfo(object, mDataType))`, the number of rows in the feature metadata.

mDataNames: `character` Retrieve the names of the molecular data types available in the `molecularProfiles` slot of a `PharmacoSet` object. These are the options which can be used in the `mDataType` parameter of various `molecularProfiles` slot accessors methods.

mDataNames: Update the molecular data type names of the `molecularProfiles` slot of a PharmacoSet object. Arguments:

- value: `character` vector of molecular datatype names, with length equal to `length(molecularProfilesSlot(object`

molecularProfilesSlot: Return the contents of the `@molecularProfiles` slot of a `PharmacoSet` object. This will either be a `list` or `MultiAssayExperiment` of `SummarizedExperiments`.

molecularProfilesSlot<-: Update the contents of the `@molecularProfiles` slot of a `PharmacoSet` object. Arguemnts:

- value: A `list` or `MultiAssayExperiment` of `SummarizedExperiments`. The `list` and `assays` should be named for the molecular datatype in each `SummarizedExperiment`.

@treatmentResponse:

Arguments::
- dimension: Optional `character(1)` One of 'treatment', 'sample' or 'assay' to retrieve `rowData`, `colData` or the 'assay_metadata' assay from the `PharmacoSet` `@sensitvity` `LongTable` object, respectively. Ignored with warning if `@treatmentResponse` is not a `LongTable` object.
- ...: Additional arguments to the `rowData` or `colData`. `LongTable` methods. Only used if the sensitivity slot contains a `LongTable` object instead of a `list` and the `dimension` argument is specified.

Methods::
sensitivityInfo: `DataFrame` or `data.frame` of sensitivity treatment combo by sample metadata for the `PharmacoSet` object. When the `dimension` parameter is used, it allows retrieval of the dimension specific metadata from the `LongTable` object in `@treatmentResponse` of a PharmacoSet object.

sensitivityInfo<-: Update the `@treatmentResponse` slot metadata for a `PharmacoSet` object. When used without the `dimension` argument is behaves similar to the old PharmacoSet implementation, where the `@treatmentResponse` slot contained a list with a `$info` data.frame item. When the `dimension` arugment is used, more complicated assignments can occur where 'sample' modifies the `@sensitvity` `LongTable` colData, 'treatment' the rowData and 'assay' the 'assay_metadata' assay. Arguments:

- value: A `data.frame` of treatment response experiment metadata, documenting experiment level metadata (mapping to treatments and samples). If the `@treatmentResponse` slot doesn't contain a `LongTable` and `dimension` is not specified, you can only modify existing columns as returned by `sensitivityInfo(object)`.

sensitivityMeaures: Get the 'sensitivityMeasures' available in a `PharmacoSet` object. Each measure reprents some summary of sample sensitivity to a given treatment, such as ic50, ec50, AUC, AAC, etc. The results are returned as a `character` vector with all available metrics for the PSet object.

sensitivityMeaures: Update the sensitivity meaure in a `PharmacoSet` object. Thesee values are the column names of the 'profiles' assay and represent various compued sensitviity metrics such as ic50, ec50, AUC, AAC, etc.

- value: A `character` vector of new sensitivity measure names, the then length of the character vector must matcht he number of columns of the 'profiles' assay, excluding metadata and key columns.

sensitivityProfiles: Return the sensitivity profile summaries from the sensitivity slot. This data.frame cotanins vaarious sensitivity summary metrics, such as ic50, amax, EC50, aac, HS, etc as columns, with rows as treatment by sample experiments.

sensitivityProfiles<-: Update the sensitivity profile summaries the sensitivity slot. Arguments: - value: A `data.frame` the the same number of rows as as returned by `sensitivityProfiles(object)`, but potentially modified columns, such as the computation of additional summary metrics.

sensitivityRaw: Access the raw sensitiity measurents for a PharmacoSet object. A 3D `array` where rows are experiment_ids, columns are doses and the third dimension is metric, either 'Dose' for the doses used or 'Viability' for the sample viability at that dose.

sensitvityRaw<-: Update the raw dose and viability data in a `PharmacoSet`.

- value: A 3D `array` object where rows are experiment_ids, columns are replicates and pages are c('Dose', 'Viability'), with the corresponding dose or viability measurement for that experiment_id and replicate.

sensNumber: Return a count of viability observations in a `PharmacoSet` object for each treatment-combo by sample combination.

sensNumber<-: Update the 'n' item, which holds a matrix with a count of treatment by sample-line experiment counts, in the `list` in @treatmentResponse slot of a `PharmacoSet` object. Will error when @sensitviity contains a `LongTable` object, since the counts are computed on the fly. Arguments:

- value: A `matrix` where rows are samples and columns are treatments, with a count of the number of experiments for each combination as the values.

pertNumber: `array` Summary of available perturbation experiments from in a `PharmacoSet` object. Returns a 3D `array` with the number of perturbation experiments per treatment and sample, and data type.

pertNumber<-: Update the @perturbation$n value in a `PharmacoSet` object, which stores a summary of the available perturbation experiments. Arguments:

- value: A new 3D `array` with the number of perturbation experiments per treatment and sample, and data type

Value

Accessors: See details.

Setters: An updated `PharmacoSet` object, returned invisibly.

Examples

```
data(CCLEsmall)
treatmentInfo(CCLEsmall)

treatmentInfo(CCLEsmall) <- treatmentInfo(CCLEsmall)

treatmentNames(CCLEsmall)

treatmentNames(CCLEsmall) <- treatmentNames(CCLEsmall)
```

```
## @annotation

annotation(CCLEsmall)

annotation(CCLEsmall) <- annotation(CCLEsmall)

dateCreated(CCLEsmall)

## dateCreated
dateCreated(CCLEsmall) <- date()

name(CCLEsmall)

name(CCLEsmall) <- 'new_name'

sampleInfo(CCLEsmall) <- sampleInfo(CCLEsmall)

sampleNames(CCLEsmall)

sampleNames(CCLEsmall) <- sampleNames(CCLEsmall)

## curation
curation(CCLEsmall)

curation(CCLEsmall) <- curation(CCLEsmall)

datasetType(CCLEsmall)

datasetType(CCLEsmall) <- 'both'

# No assay specified
molecularProfiles(CCLEsmall, 'rna') <- molecularProfiles(CCLEsmall, 'rna')

# Specific assay
molecularProfiles(CCLEsmall, 'rna', 'exprs') <-
    molecularProfiles(CCLEsmall, 'rna', 'exprs')

# Replace the whole slot
molecularProfiles(CCLEsmall) <- molecularProfiles(CCLEsmall)

featureInfo(CCLEsmall, 'rna')

featureInfo(CCLEsmall, 'rna') <- featureInfo(CCLEsmall, 'rna')

phenoInfo(CCLEsmall, 'rna')

phenoInfo(CCLEsmall, 'rna') <- phenoInfo(CCLEsmall, 'rna')

fNames(CCLEsmall, 'rna')

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

```
mDataNames(CCLEsmall)

mDataNames(CCLEsmall) <- mDataNames(CCLEsmall)

molecularProfilesSlot(CCLEsmall)

molecularProfilesSlot(CCLEsmall) <- molecularProfilesSlot(CCLEsmall)

sensitivityInfo(CCLEsmall)

sensitivityInfo(CCLEsmall) <- sensitivityInfo(CCLEsmall)

sensitivityMeasures(CCLEsmall) <- sensitivityMeasures(CCLEsmall)

sensitivityMeasures(CCLEsmall) <- sensitivityMeasures(CCLEsmall)

sensitivityProfiles(CCLEsmall)

sensitivityProfiles(CCLEsmall) <- sensitivityProfiles(CCLEsmall)

head(sensitivityRaw(CCLEsmall))

sensitivityRaw(CCLEsmall) <- sensitivityRaw(CCLEsmall)

treatmentResponse(CCLEsmall)

treatmentResponse(CCLEsmall) <- treatmentResponse(CCLEsmall)

sensNumber(CCLEsmall)

sensNumber(CCLEsmall) <- sensNumber(CCLEsmall)

pertNumber(CCLEsmall)

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

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.

Arguments

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

Value

An object of the PharmacoSet class

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

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

treatment A `data.frame` containg the annotations for all the drugs profiled in the data set, across all data types

treatmentResponse 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 `$treatment`, `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

---

PharmacoSet-utils   *Utility methods for a* PharmacoSet *object.*

Description

Documentation for utility methods for a `PharmacoSet` object, such as set operations like subset and intersect. See @details for information on different types of methods and their implementations.

Usage

```
## S4 method for signature 'PharmacoSet'
subsetBySample(x, samples)

## S4 method for signature 'PharmacoSet'
subsetByTreatment(x, treatments)

## S4 method for signature 'PharmacoSet'
subsetByFeature(x, features, mDataTypes)
```

Arguments

x	A PharmacoSet object.
samples	character() vector of sample names. Must be valid rownames from `sampleInfo(x)`.
treatments	character() vector of treatment names. Must be valid rownames from `treatmentInfo(x)`. This method does not work with `CoreSet` objects yet.
features	character() vector of feature names. Must be valid feature names for a given `mDataType`
mDataTypes	character() One or more molecular data types to to subset features by. Must be valid rownames for the selected SummarizedExperiment mDataTypes.

Details

subset methods:

subsetBySample: Subset a `PharmacoSet` object by sample identifier.

- value: a PharmacoSet object containing only `samples`.

subset methods:

subsetByTreatment: Subset a `PharmacoSet` object by treatment identifier.

- value: a PharmacoSet object containing only `treatments`.

subset methods:

subsetByFeature: Subset a `PharmacoSet` object by molecular feature identifier.

- value: a PharmacoSet object containing only `features`.

Value

See details.

Examples

```
data(CCLEsmall)

## subset methods

### subsetBySample
samples <- sampleInfo(CCLEsmall)$sampleid[seq_len(10)]
```

```
CCLEsmall_sub <- subsetBySample(CCLEsmall, samples)


## subset methods

### subsetByTreatment
#treatments <- drugInfo(CCLEsmall)$drugid[seq_len(10)]
#CCLEsmall_sub <- subsetByTreatment(CCLEsmall, treatments)


## subset methods

### subsetByFeature
features <- fNames(CCLEsmall, 'rna')[seq_len(5)]
CCLEsmall_sub <- subsetByFeature(CCLEsmall, features, 'rna')
```

PharmacoSet2   *Make a CoreSet with the updated class structure*

Description

New implementation of the CoreSet constructor to support MAE and TRE. This constructor will be swapped with the original `CoreSet` constructor as part of an overhaul of the CoreSet class structure.

Usage

```
PharmacoSet2(
  name = "emptySet",
  treatment = data.frame(),
  sample = data.frame(),
  molecularProfiles = MultiAssayExperiment(),
  treatmentResponse = TreatmentResponseExperiment(),
  perturbation = list(),
 curation = list(sample = data.frame(), treatment = data.frame(), tissue = data.frame()),
  datasetType = "sensitivity"
)
```

Arguments

name	A `character(1)` vector with the `PharmacoSet` objects name.
treatment	A `data.frame` with treatment level metadata. Treatments in a `PharmacoSet` represent pharmaceutical compounds.
sample	A `data.frame` with sample level metadata for the union of samples in `treatmentResponse` and `molecularProfiles`. Samples in a `PharmacoSet` represent cancer cell-lines.

molecularProfiles

> A MultiAssayExperiment containing one SummarizedExperiment object for each molecular data type.

treatmentResponse

> A LongTable or LongTableDataMapper object containing all treatment response data associated with the PharmacoSet object.

perturbation   A deprecated slot in a PharmacoSet object included for backwards compatibility. This may be removed in future releases.

curation   This class requires an additional curation item, tissue, which maps from published to standardized tissue idenifiers.

datasetType   A deprecated slot in a PharmacoSet object included for backwards compatibility. This may be removed in future releases.

Value

A CoreSet object storing standardized and curated treatment response and multiomic profile data associated with a given publication.

Examples

```
data(CCLEsmall)
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`	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

Examples

```
PharmacoSig()
```

---

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(1) or logical(1) 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)
```

---

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  <- treatmentNames(CCLEsmall)
CCLEcells <- sampleNames(CCLEsmall)
pSet <- subsetTo(CCLEsmall, drugs = CCLEdrugs[1], cells = CCLEcells[1])
pSet
```

---

summarizeMolecularProfiles,PharmacoSet-method
                         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

```
## S4 method for signature 'PharmacoSet'
summarizeMolecularProfiles(
  object,
  mDataType,
  cell.lines,
  features,
  summary.stat = c("mean", "median", "first", "last", "and", "or"),
  fill.missing = TRUE,
  summarize = TRUE,
  verbose = TRUE,
  binarize.threshold = NA,
  binarize.direction = c("less", "greater"),
  removeTreated = TRUE
)
```

Arguments

`object`	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 (defaults to TRUE) disables summarizing and returns the data unchanged as a ExpressionSet
`verbose`	boolean should messages be printed
`binarize.threshold`	
	numeric A value on which the molecular data is binarized. If NA, no binarization is done.
`binarize.direction`	
	character One of "less" or "greater", the direction of binarization on binarize.threshold, if it is not NA.
`removeTreated`	logical If treated/perturbation experiments are present, should they be removed? Defaults to yes.

Value

`matrix` An updated PharmacoSet with the molecular data summarized per cell line.

Examples

```
data(GDSCsmall)
GDSCsmall <- summarizeMolecularProfiles(GDSCsmall, mDataType = "rna", cell.lines=sampleNames(GDSCsmall), summary.
GDSCsmall
```

---

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,
  profiles_assay = "profiles",
  treatment_col = "treatmentid",
  sample_col = "sampleid",
  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 The 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 have no data, they will be filled with missing values.
profiles_assay	character The name of the assay in the PharmacoSet object that contains the sensitivity profiles.
treatment_col	character The name of the column in the profiles assay that contains the treatment IDs.
sample_col	character The name of the column in the profiles assay that contains the sample IDs.

drugs	character The drugs to be summarized. If any drugs have no data, they will be filled with missing values.
summary.stat	character The summary method to use if there are repeated cell line-drug experiments. Choices are "mean", "median", "first", "last", "max", or "min".
fill.missing	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')
```

---

updateObject,PharmacoSet-method

Update the PharmacoSet class after changes in it struture or API

Description

Update the PharmacoSet class after changes in it struture or API

Usage

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

Arguments

object	A PharmacoSet object to update the class structure for.

Value

PharmacoSet with update class structure.

Examples

```
data(GDSCsmall)
updateObject(GDSCsmall)
```

[,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