Package 'oncoPredict'

June 29, 2026

Type Package

Title Drug Response Modeling and Biomarker Discovery

Version 1.3.1

URL https://github.com/HuangLabUMN/oncoPredict

BugReports https://github.com/HuangLabUMN/oncoPredict/issues

Maintainer Robert Gruener <rgruener@umn.edu>

Description Allows for building drug response models using screening data between bulk RNA-Seq and a drug response metric and two additional tools for biomarker discovery that have been developed by the Huang Laboratory at University of Minnesota.
There are 3 main functions within this package.
(1) calcPhenotype() is used to build drug response models on RNA-Seq data and impute them on any other RNA-Seq dataset given to the model.
(2) GLDS() is used to calculate the general level of drug sensitivity, which can improve biomarker discovery.
(3) IDWAS() can take the results from calcPhenotype() and link the imputed response back to available genomic (mutation and CNV alterations) to identify biomarkers.
Each of these functions comes from a paper from the Huang research laboratory. Below gives the relevant paper for each function.
The package is described in Maeser et al. (2021) ``oncoPredict: an R package for predicting in vivo or cancer patient drug response and biomarkers from cell line screening data" <doi:10.1093/bib/bbab260>.
calcPhenotype() - Geeleher et al, Clinical drug response can be predicted using baseline gene expression levels and in vitro drug sensitivity in cell lines.
GLDS() - Geeleher et al, Cancer biomarker discovery is improved by accounting for variability in general levels of drug sensitivity in pre-clinical models.
IDWAS() - Geeleher et al, Discovering novel pharmacogenomic biomarkers by imputing drug response in cancer patients from large genomics studies.

License GPL-2

Encoding UTF-8

RoxygenNote 7.3.2

Depends R (>= 4.1.0)

biocViews sva, limma, stringr, biomaRt, genefilter, GenomicFeatures, genefilter, TCGAbiolinks, BiocGenerics, GenomicRanges, IRanges, S4Vectors

Imports parallel, ridge, car, glmnet, pls, sva, limma, GenomicFeatures, BiocGenerics, GenomicRanges, IRanges, S4Vectors

Suggests knitr, rmarkdown, org.Hs.eg.db, TxDb.Hsapiens.UCSC.hg19.knownGene, TCGAbiolinks, testthat (>= 3.0.0)

Additional_repositories https://bioconductor.org/packages/release/data/annotation

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Robert Gruener [aut, cre] (ORCID: <https://orcid.org/0000-0002-8166-2772>), Danielle Maeser [aut] (ORCID: <https://orcid.org/0000-0002-3890-887X>)

Repository CRAN

Date/Publication 2026-06-29 14:50:02 UTC

Contents

`calcPhenotype` *Generate predicted drug sensitivity scores*

Description

This function predicts a phenotype (drug sensitivity score) when provided with microarray or bulk RNAseq gene expression data of different platforms. The imputations are performed using ridge regression, training on a gene expression matrix where phenotype is already known. This function integrates training and testing datasets via a user-defined procedure, and power transforming the known phenotype.

Usage

```
calcPhenotype(
  trainingExprData,
  trainingPtype,
  testExprData,
  batchCorrect,
  powerTransformPhenotype = TRUE,
  removeLowVaryingGenes = 0.2,
  minNumSamples,
  selection = 1,
  printOutput,
  pcr = FALSE,
  removeLowVaringGenesFrom,
  report_pc = FALSE,
  cc = FALSE,
  percent = 80,
  rsq = FALSE,
  folder = FALSE,
  parallel = FALSE,
  cores = 1
)
```

Arguments

`trainingExprData`
: The training data. A matrix of expression levels. rownames() are genes, colnames() are samples (cell line names or cosmic ides, etc.). rownames() must be specified and must contain the same type of gene ids as "testExprData"

`trainingPtype`
: The known phenotype for "trainingExprData". This data must be a matrix with training samples as rows and drugs or phenotypes as columns. This matrix can contain NA values, that is ok (they are removed in the calcPhenotype() function).

`testExprData`
: The test data where the phenotype will be estimated. It is a matrix of expression levels, rows contain genes and columns contain samples, "rownames()" must be specified and must contain the same type of gene ids as "trainingExprData".

`batchCorrect`
: How should training and test data matrices be homogenized. Choices are "eb" (default) for ComBat, "qn" for quantile normalization, "standardize" for within-dataset z-score standardization, "rank", "rank_then_eb", or "none" for no homogenization.

`powerTransformPhenotype`
: Should the phenotype be power transformed before we fit the regression model? Default to TRUE, set to FALSE if the phenotype is already known to be highly normal.

`removeLowVaryingGenes`
: What proportion of low varying genes should be removed? 20 percent be default

`minNumSamples`
: How many training and test samples are required. Print an error if below this threshold

selection	How should duplicate gene ids be handled. Default is -1 which asks the user. 1 to summarize by their or 2 to disguard all duplicates.
printOutput	Set to FALSE to supress output.
pcr	Indicates whether or not you'd like to use pcr for feature (gene) reduction. Options are 'TRUE' and 'FALSE'. If you indicate 'report_pc=TRUE' you need to also indicate 'pcr=TRUE'
removeLowVaringGenesFrom	Determine method to remove low varying genes. Options are 'homogenizeData' and 'rawData'.
report_pc	Indicates whether you want to output the training principal components. Options are 'TRUE' and 'FALSE'.
cc	Indicate if you want correlation coefficients for biomarker discovery.
percent	Indicate percent variability (of the training data) you'd like principal components to reflect if pcr=TRUE. Default is 80 for 80% These are the correlations between a given gene of interest across all samples vs. a given drug response across samples. These correlations can be ranked to obtain a ranked correlation to determine highly correlated drug-gene associations.
rsq	Indicate whether or not you want to output the R^2 values for the data you train on from true and predicted values. These values represent the percentage in which the optimal model accounts for the variance in the training data. Options are 'TRUE' and 'FALSE'.
folder	If TRUE, write calcPhenotype outputs to calcPhenotype_Output in the current working directory. The default is FALSE.
parallel	If TRUE, fit drug models in parallel after the shared homogenization and gene-filtering steps are complete. The default is FALSE.
cores	The number of cores to use when parallel is TRUE. Parallel execution uses forked processes via parallel::mclapply, which is not available for multicore execution on Windows PCs; on Windows, calcPhenotype will warn and run serially.

Value

A matrix of predicted drug response values. If rsq, cc, or report_pc is TRUE, returns a list containing the predictions and requested optional outputs. If folder is TRUE, the same object is returned invisibly after files are written.

Examples

```
set.seed(1)
genes <- paste0("gene", 1:30)
trainingExprData <- matrix(rnorm(30 * 8), nrow=30,
                           dimnames=list(genes, paste0("train", 1:8)))
testExprData <- matrix(rnorm(30 * 3), nrow=30,
                       dimnames=list(genes, paste0("test", 1:3)))
trainingPtype <- matrix(rnorm(8), ncol=1,
                        dimnames=list(colnames(trainingExprData), "drug1"))
predictions <- calcPhenotype(trainingExprData, trainingPtype, testExprData,
                             batchCorrect="none",
```

```
                              powerTransformPhenotype=FALSE,
                              removeLowVaryingGenes=0,
                              minNumSamples=0,
                              selection=1,
                              printOutput=FALSE,
                              pcr=FALSE,
                              removeLowVaringGenesFrom="rawData")
head(predictions)
```

`completeMatrix` *This function performs an iterative matrix completion algorithm to predict drug response for pre-clinical data when there are missing ('NA') values.*

Description

This function performs an iterative matrix completion algorithm to predict drug response for pre-clinical data when there are missing ('NA') values.

Usage

```
completeMatrix(senMat, nPerms = 50, folder = FALSE)
```

Arguments

`senMat`	A matrix of drug sensitivity data with missing ('NA') values. rownames() are samples (e.g. cell lines), and colnames() are drugs.
`nPerms`	The number of iterations that the EM-algorithm (expectation maximization approach) run. The default is 50, as previous findings recommend 50 iterations (https://genomebiology.biomedcentral.com/articles/10.1186/s13059-016-1050-9)
`folder`	If TRUE, write the completed matrix to complete_matrix_output.txt in the current working directory. The default is FALSE.

Value

A matrix of drug sensitivity scores without missing values. rownames() are samples, and colnames are drugs.

Examples

```
set.seed(1)
senMat <- matrix(rnorm(80 * 4), nrow=80,
                 dimnames=list(paste0("sample", 1:80), paste0("drug", 1:4)))
senMat[1, 1] <- NA
completed <- completeMatrix(senMat, nPerms=1)
dim(completed)
```

doVariableSelection *Remove genes with low variation.*

Description

This function performs variable selection by removing genes with the lowest variance in the datasets.

Usage

```
doVariableSelection(exprMat, removeLowVaryingGenes = 0.2)
```

Arguments

exprMat	A matrix of gene expression levels. rownames() are genes, and colnames() are samples.
removeLowVaryingGenes	The proportion of low varying genes to be removed.The default is .2

Value

A vector of row/genes to keep.

Examples

```
exprMat <- matrix(seq_len(20), nrow=5,
                  dimnames=list(paste0("gene", 1:5), paste0("sample", 1:4)))
doVariableSelection(exprMat, removeLowVaryingGenes=0.2)
```

glds *This function determines drug-gene associations for pre-clinical data.*

Description

This function determines drug-gene associations for pre-clinical data.

Usage

```
glds(
  drugMat,
  drugRelatedness,
  markerMat,
  minMuts = 5,
  additionalCovariateMatrix = NULL,
  expression = NULL,
  threshold = 0.7,
  folder = FALSE
)
```

Arguments

drugMat	A matrix of drug sensitivity data. rownames() are pre-clinical samples, and colnames() are drug names.
drugRelatedness	A matrix in which column 1 contains a list of compounds, and column 2 contains a list of their corresponding target pathways. Given the subjective nature of drug classification, please ensure these pathways are as specific as possible for accurate results.
markerMat	A matrix containing the data for which you are looking for an association with drug sensitivity (e.g. a matrix of somatic mutation data). rownames() are marker names (e.g. gene names), and colnames() are samples.
minMuts	The minimum number of non-zero entries required so that a p-value can be calculated (e.g. how many somatic mutations must be present). The default is 5.
additionalCovariateMatrix	A matrix containing covariates to be fit in the drug biomarker association models. This could be, for example, tissue of origin or cancer type. Rows are sample names. The default is NULL.
expression	A matrix of expression data. rownames() are genes, and colnames() are the same pre-clinical samples as those in the drugMat (also in the same order). The default is NULL. If expression data is provided, a gene signature will be obtained.
threshold	Determine the correlation coefficient. Drugs with a correlation coefficient greater than or equal to this number with the drug under scrutiny will be removed from the negative control group. The default is 0.7
folder	If TRUE, write GLDS p-value and beta outputs to CSV files in the current working directory. The default is FALSE.

Value

A list containing GLDS-adjusted p-values and beta values, plus naive p-values and beta values.

Examples

```
set.seed(1)
sampleNames <- paste0("cell", 1:14)
drugNames <- paste0("drug", 1:6)
drugMat <- matrix(rnorm(14 * 6), nrow=14,
                  dimnames=list(sampleNames, drugNames))
drugRelatedness <- cbind(drug=drugNames,
                         pathway=paste0("pathway", 1:6))
markerMat <- matrix(c(rep(c(0, 1), 7), rep(c(1, 0), 7)),
                    nrow=2, byrow=TRUE,
                    dimnames=list(c("marker1", "marker2"), sampleNames))
gldsResult <- glds(drugMat, drugRelatedness, markerMat,
                   minMuts=1, threshold=1)
names(gldsResult)
```

homogenizeData *Homogenizes two expression matrices*

Description

This function takes two gene expression matrices (like trainExprMat and testExprMat) and returns homogenized versions of the matrices by employing the homogenization method specified. By default, the Combat method from the sva library is used. In both matrices, genes are row names and samples are column names. It will deal with duplicated gene names, as it subsets and orders the matrices correctly.

Usage

```
homogenizeData(
  testExprMat,
  trainExprMat,
  batchCorrect = "eb",
  selection = -1,
  printOutput = TRUE
)
```

Arguments

testExprMat	A gene expression matrix for samples on which we wish to predict a phenotype.Genes are rows, samples are columns.
trainExprMat	A gene expression matrix for samples for which the phenotype is already known.Genes are rows, samples are columns.
batchCorrect	The type of batch correction to be used. Options are 'eb' for ComBat, 'qn' for quantile normalization, 'standardize' for within-dataset z-score standardization, 'rank', 'rank_then_eb', or 'none'. The 'standardize' option can be useful when using microarray training data to build models on RNA-seq testing data. #The default is 'eb'.
selection	This parameter can be used to specify how duplicates are handled. The default value of -1 means to ask the user. #Other options include '1' to summarize duplicates by their mean, and '2'to discard all duplicated genes.
printOutput	To suppress output, set to false. Default is TRUE.

Value

A list containing two entries $train and $test, which are the homogenized input matrices.

Examples

```
trainExpr <- matrix(rnorm(20), nrow=5,
                    dimnames=list(paste0("gene", 1:5), paste0("train", 1:4)))
testExpr <- matrix(rnorm(10), nrow=5,
```

```
                    dimnames=list(paste0("gene", 1:5), paste0("test", 1:2)))
homData <- homogenizeData(testExpr, trainExpr, batchCorrect="none",
                          selection=1, printOutput=FALSE)
names(homData)
```

`idwas` *This function will test every drug against CNA amplifications or somatic mutations for your cancer type.*

Description

This function will test every drug against CNA amplifications or somatic mutations for your cancer type.

Usage

```
idwas(drug_prediction, data, n = 10, cnv, folder = FALSE)
```

Arguments

`drug_prediction`	The drug prediction data. Must be a data frame. rownames are samples, colnames are drugs. Make sure sample names are of the same form as the sample names in your cnv or mutation data. e.g. if the rownames() are TCGA barcodes of the form TCGA-##-####-###, make sure your cnv/mutation data also uses samples in the form TCGA-##-####-###
`data`	The CNA amplification or mutation data. Must be a data frame. If you wish to use CNA data, use the output from map_cnv(), where rows are genes and columns are samples, or use data of similar form. If you wish to use mutation data, use the method for downloading mutation data outlined in the vignette, and make sure the TCGA barcodes use '-' instead of '.'; if you use another dataset (and don't download data from TCGA), make sure your data file includes the following columns: 'Variant_Classification', 'Hugo_Symbol', 'Tumor_Sample_Barcode'.
`n`	The minimum number of samples you want CNA amplifications or mutations to occur in. The default is 10 (arbitrarily chosen).
`cnv`	TRUE or FALSE. Indicate whether or not you would like to test CNA amplification data. If TRUE, you will test CNA amplifications. If FALSE, you will test mutation data.
`folder`	If TRUE, write IDWAS results to CSV files in the current working directory. The default is FALSE.

Value

Raw p-value and beta-values for CNA amplifications or somatic mutations.

Examples

```
drugPrediction <- data.frame(drug1=c(1, 2, 3),
                             drug2=c(2, 1, 3),
                             row.names=c("S1", "S2", "S3"))
mutationData <- data.frame(Variant_Classification=c("Missense_Mutation",
                                                    "Missense_Mutation",
                                                    "Missense_Mutation"),
                           Tumor_Sample_Barcode=c("S1", "S2", "S3"),
                           Hugo_Symbol=c("GENE1", "GENE1", "GENE2"))
idwas(drugPrediction, mutationData, n=1, cnv=FALSE)
```

---

map_cnv *This function maps cnv data to genes.*

Description

This function maps cnv data to genes.

Usage

```
map_cnv(Cnvs, folder = FALSE)
```

Arguments

Cnvs	The cnv data. A table with the following colnames: Sample (named using the TCGA patient barcode), Chromosome, Start, End, Num_Probes, and Segment_Mean.
folder	If TRUE, write a map.RData file containing the mapped CNV matrix and tumor sample indices. The default is FALSE.

Value

A list containing theCnvQuantVecList_mat (rows are genes, columns are samples) and tumorSamps (primary tumor/01A sample column indices).

Examples

```
if (requireNamespace("org.Hs.eg.db", quietly=TRUE) &&
    requireNamespace("TxDb.Hsapiens.UCSC.hg19.knownGene", quietly=TRUE)) {
  cnvData <- data.frame(Sample="TCGA-01-0001-01A-01D-0000-01",
                        Chromosome=7,
                        Start=55000000,
                        End=55200000,
                        Num_Probes=10,
                        Segment_Mean=1.5)
  mapped <- map_cnv(cnvData)
  names(mapped)
}
```

predictionAccuracybyCV *Calculate Cross-Validation Scores using OncoPredict*

Description

This function predicts a phenotype (drug sensitivity score) when provided with microarray or bulk RNAseq gene expression data of different platforms. The imputations are performed using ridge regression, training on a gene expression matrix where phenotype is already known. This function integrates training and testing datasets via a user-defined procedure, and power transforming the known phenotype.

Usage

```
predictionAccuracybyCV(
  trainingExprData,
  trainingPtype,
  testExprData,
  batchCorrect,
  powerTransformPhenotype = TRUE,
  removeLowVaryingGenes = 0.2,
  minNumSamples,
  selection = 1,
  printOutput,
  pcr = FALSE,
  removeLowVaringGenesFrom,
  report_pc = FALSE,
  cc = FALSE,
  percent = 80,
  rsq = FALSE,
  folder = FALSE,
  cvFold = -1,
  parallel = FALSE,
  cores = 1
)
```

Arguments

`trainingExprData`
: The training data. A matrix of expression levels. rownames() are genes, colnames() are samples (cell line names or cosmic ides, etc.). rownames() must be specified and must contain the same type of gene ids as "testExprData"

`trainingPtype`
: The known phenotype for "trainingExprData". This can be a one-drug vector with one value per training sample or a matrix with training samples as rows and drugs or phenotypes as columns. This matrix can contain NA values, that is ok (they are removed in the calcPhenotype() function).

`testExprData` The test data where the phenotype will be estimated. It is a matrix of expression levels, rows contain genes and columns contain samples, "rownames()" must be specified and must contain the same type of gene ids as "trainingExprData".

`batchCorrect` How should training and test data matrices be homogenized. Choices are "eb" (default) for ComBat, "qn" for quantile normalization, "standardize" for within-dataset z-score standardization, "rank", "rank_then_eb", or "none" for no homogenization.

`powerTransformPhenotype` Should the phenotype be power transformed before we fit the regression model? Default to TRUE, set to FALSE if the phenotype is already known to be highly normal.

`removeLowVaryingGenes` What proportion of low varying genes should be removed? 20 percent be default

`minNumSamples` How many training and test samples are required. Print an error if below this threshold

`selection` How should duplicate gene ids be handled. Default is -1 which asks the user. 1 to summarize by their or 2 to disguard all duplicates.

`printOutput` Set to FALSE to supress output.

`pcr` Indicates whether or not you'd like to use pcr for feature (gene) reduction. Options are 'TRUE' and 'FALSE'. If you indicate 'report_pc=TRUE' you need to also indicate 'pcr=TRUE'

`removeLowVaringGenesFrom` Determine method to remove low varying genes. Options are 'homogenizeData' and 'rawData'.

`report_pc` Indicates whether you want to output the training principal components. Options are 'TRUE' and 'FALSE'.

`cc` Indicate if you want correlation coefficients for biomarker discovery.

`percent` Indicate percent variability (of the training data) you'd like principal components to reflect if pcr=TRUE. Default is 80 for 80% These are the correlations between a given gene of interest across all samples vs. a given drug response across samples. These correlations can be ranked to obtain a ranked correlation to determine highly correlated drug-gene associations.

`rsq` Indicate whether or not you want to output the R^2 values for the data you train on from true and predicted values. These values represent the percentage in which the optimal model accounts for the variance in the training data. Options are 'TRUE' and 'FALSE'.

`folder` Retained for compatibility with calcPhenotype arguments. Cross-validation results are returned as an object.

`cvFold` Indicate the number of k-folds wanted in the CV calculation. -1 indicates a leave-one-out cross validation

`parallel` If TRUE, run cross-validation folds in parallel. The default is FALSE.

`cores` The number of cores to use when parallel is TRUE. Parallel execution uses forked processes via parallel::mclapply, which is not available for multicore execution on Windows PCs; on Windows, predictionAccuracybyCV will warn and run serially.

Value

A list containing cross-validated predicted phenotype values and real phenotype values.

Examples

```
set.seed(1)
genes <- paste0("gene", 1:30)
trainingExprData <- matrix(rnorm(30 * 8), nrow=30,
                           dimnames=list(genes, paste0("train", 1:8)))
trainingPtype <- matrix(rnorm(8), ncol=1,
                        dimnames=list(colnames(trainingExprData), "drug1"))
cvResult <- predictionAccuracybyCV(trainingExprData, trainingPtype,
                                   testExprData=NULL,
                                   batchCorrect="none",
                                   powerTransformPhenotype=FALSE,
                                   removeLowVaryingGenes=0,
                                   minNumSamples=0,
                                   selection=1,
                                   printOutput=FALSE,
                                   pcr=FALSE,
                                   removeLowVaringGenesFrom="rawData",
                                   cvFold=2)
names(cvResult)
```

---

summarizeGenesByMean *Average over duplicate gene values*

Description

This function takes a gene expression matrix and if duplicate genes are measured, summarizes them by their means.

Usage

```
summarizeGenesByMean(exprMat)
```

Arguments

exprMat	A gene expression matrix with genes as rownames() and samples as colnames().

Value

A gene expression matrix that does not contain duplicate genes.

Examples

```
exprMat <- matrix(seq_len(12), nrow=3,
                  dimnames=list(paste0("gene", 1:3), paste0("sample", 1:4)))
summarizeGenesByMean(exprMat)
```

Index