Package 'curatedBreastData'

October 16, 2025

Type Package

Title Curated breast cancer gene expression data with survival and treatment information

Version 2.36.0

Date 2016-10-26

Author Katie Planey

Maintainer Katie Planey <katie.planey@gmail.com>

Depends R (>= 3.0.0), XML, ggplot2, impute, Biobase, BiocStyle

Imports methods, stats

biocViews ExperimentData, ExpressionData, CancerData, Tissue, BreastCancerData, qPCRData, MicroarrayData, TissueMicroarrayData, GEO

Description Curated human breast cancer tissue S4 ExpresionSet datasets from over 16 clinical trials comprising over 2,000 patients. All datasets contain at least one type of outcomes variable and treatment information (minimum level: whether they had chemotherapy and whether they had hormonal therapy). Includes code to post-process these datasets.

License GPL (>= 2)

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

git_branch RELEASE_3_21

git_last_commit 51e5733

git_last_commit_date 2025-04-15

Repository Bioconductor 3.21

Date/Publication 2025-10-16

Contents

`curatedBreastData-package`

Curated breast gene expression data with survival and treatment information

---

Description

34 manually curated high-quality gene expression microarray datasets with advanced beast cancer samples collected from GEO. All datasets provided have some form of survival and treatment information, and all such clinical variables are semantically normalized across all datasest for easy analyses across datasets. Authors of the Pubmed article linked to each GEO dataset was contacted in an effort to collect as much extra clinical data as possible. See vignette and publication reference from AMIA Translational Science Joint Summits presentation in 2013 for more details on how this data was curated.

Functions are provided to post-process standard S4 ExpressionSet objects to remove samples with high NA rates, impute missing values, collapse duplicated gene symbols or probes, remove duplicated samples that share the same patient ID, and filter genes by variance magnitude or percentile.

Details

Package:	curatedBreastData
Type:	Package
Version:	1.0
Date:	2015-02-25
License:	What license is it under?

Note

Suggestions for new datasets to add are always welcome; the maintainer does aim to only include datasets that have minimal treatment and some form of survival (and/or treatment response) to allow for richer analyses. Raw data is always preferred in order to control normalization schemes. Normalization details for each dataset can be found the the Github repo in the References section.

Author(s)

Katie Planey

Maintainer: Katie Planey <katie.planey@gmail.com>

References

Planey, Butte. Database integration of 4923 publicly-available samples of breast cancer molecular and clinical data. AMIA Joint Summits Translational Science Proceedings. (2003) PMC3814460

Github repo with code, further documentation on datasets and baseline normalization schemes, and database quality checks: https://github.com/kplaney/curatedBreastCancer

Examples

```
#don't run below in examples() because
#somewhat slow, and similar examples are already run
#from individual man function files
## Not run:
#load up master clinical data table
data(clinicalTable)
#Check out the treatment information.
head(clinicalTable)[,c(112:ncol(clinicalTable))]
#how many had chemotherapy?
numChemoPatients <- length(which(clinicalTable$chemotherapyClass==1))
#how many patients have non-NA OS binary data?


#load up datasets that are in S4 expressionSet format.
#clinical data from master clinicalTable already linked to each sample
#in these ExpressionSets in the phenoData slot.
data(curatedBreastDataExprSetList);

#process only the first two datasets to avoid a long-running example:
#take top 5000 genes by variance from each dataset.
proc_curatedBreastDataExprSetList <- processExpressionSetList(exprSetList=
curatedBreastDataExprSetList[1:2],
outputFileDirectory = "./", numTopVarGenes=5000)

#now we have processed expression matrices,
#each with the top 5000 genes by variance

## End(Not run)
```

---

clinicalData *Clinical Data Table & Variable Definitions*

Description

Clinical data for all samples across all studies, and corresponding variable definitions. Rownames are the GEO_GSMID feature, which corresponds to the sample names in the expression object for a certain study. Includes treatment information.

Usage

```
data("clinicalData")
```

Format

A list with the following two items: -clinicalTable:A data frame. Rownames are the GEO_GSMID feature, which corresponds to the sample names in the expression object for a certain study. -clinicalVarDef:Character string descriptions of each variable.

Details

GEO study ID can be found form the study_ID variable. If site_ID is NA, it pertains to the batch ID, which may be due to different platforms being used in the same study or different tissue site collections. Columns 112-151 pertaint to treatment information. radiotherapyClass, chemotherapyClass, and hormone_therapyClass are indicator variables used to signal whether a patient had radiotherapy, chemotherapy, and/or some form of hormone therapy (usually an estrogen or aromatse inhibitor.)

More granular information, when available, is provided: for example, whether the chemotherapy drug was capecitabine is coded as the indicator "capecitabine" variable. A value of 1 = yes, 0 = no, NA = not recorded/could not infer from publically available information. "Other" means that most likely, gleaned from the study's Pubmed publication, that the patient may have had other treatments that were not recorded (oftentimes radiotherapy, as this is not always recorded and up to a clinician's discretion in a clinical trial.)

Survival information, such as DFS, RFS, OS, and treatment response information, such pCR and RCB, is also recorded when available.

Value

No return value as this is not a function but rather a data object.

References

Planey, Butte. Database integration of 4923 publicly-available samples of breast cancer molecular and clinical data. AMIA Joint Summits Translational Science Proceedings. (2003) PMC3814460

Examples

```
data(clinicalData)
#check out some of the variable name/definitions
clinicalData$clinicalVarDef[c(1:2),]
#Check out the treatment information.
#look at first three patients
head(clinicalData$clinicalTable)[c(1:3),c(112:ncol(clinicalData$clinicalTable))]
#how many had chemotherapy?
numChemoPatients <- length(which(
clinicalData$clinicalTable$chemotherapyClass==1))
#how many patients have non-NA OS binary data?
length(which(!is.na(clinicalData$clinicalTable$OS)))
#how many have OS data in the more granular form of months until OS?
#this variable includes studies that had a cieling for tracking OS
length(which(!is.na(clinicalData$clinicalTable$OS_months_or_MIN_months_of_OS)))
```

```
#how many patients have OS information that is definitively
#followed up until their death
#(details on how studies collect OS data can be surprising!)
length(which(!is.na(clinicalData$clinicalTable$OS_up_until_death)))
```

---

collapseDupProbes *Collapse/handle duplicated probes (genes) in a dataset*

Description

Used internally by processExpressionSet. Code to either take the average across a set of duplicated "keys" (can be probes or genes, which correspond to the rows in the expression matrix "expr"), or take the keys that has the highest variance across the set of duplicated keys.

Usage

```
collapseDupProbes(expr, sampleColNames=colnames(expr),
keys, method = c("average", "highestVariance"), debug = TRUE,
removeNA_keys = TRUE,
varMetric = c("everything", "all.obs", "complete.obs", "na.or.complete",
"pairwise.complete.obs"))
```

Arguments

expr	An expression matrix with genes in the rows and samples in the columns.
sampleColNames	Sample column names. Needed for internal debugging; usually the default colnames(expr) is appropriate.
keys	Generally the list of gene symbols, or some molecular key, that needs to be "collapsed" because it contains duplicated names.
method	Method used to collapse probes: take the mean across all duplicated keys, or just pick the key with the highest variance?
debug	Use internal unit tests that will stop the code if it detects a bug?
removeNA_keys	Remove any NA keys?
varMetric	Standard options taken from the base var() function. May be important if you have NA values in your data matrix; otherwise, "everything" is usually fine.

Value

Returns a processed list with the items "expr" and "keys", the expression matrix and final keys list.

Author(s)

Katie Planey <katie.planey@gmail.com>

Examples

```
#load up our datasets
data(curatedBreastDataExprSetList);

#just perform on second dataset, GSE2034, as an example.
#This dataset has no NAs already but does have duplicated genes
#highestVariance calculation make take a minute to run.
collapsedData <- collapseDupProbes(expr=exprs(curatedBreastDataExprSetList[[2]]),
keys=curatedBreastDataExprSetList[[2]]@featureData$gene_symbol,
method = c("highestVariance"), debug = TRUE, removeNA_keys = TRUE,
varMetric = c("everything"))
#look at names of outputs
names(collapsedData)
```

`curatedBreastDataExprSetList`

curatedBreastDataExprSetList

Description

A list of ExpressionSet objects, one for each curated study, containing study-specific gene expression and phenotype data. FeatureNames are gene symbols. Data is already quantile normalized according to standard protocols for 1 and 2-channel arrays, depending on the platform used for this study.

Usage

```
data("curatedBreastDataExprSetList")
```

Format

A list, with each index containing an ExpressionSet object from a specific study, and potentially a specific batch.

Details

Batches from studies are treated as individual datasets, as the signal can differ between batches. Thus, an expression object named using a GSE study number followed by an underscore means this ExpressionSet contains samples either from a distinct platform (and the study used >1 platforms), or from a distinct batch or tissue site. An "all" tag means that there were no batches for this study. Raw data files downloaded from GEO oftentimes have clear batch/site information appended to sample names; this was often the source of batch identification and how the package developer chose to create the batch name string.

Value

No return value as this is not a function but rather a data object.

References

Planey, Butte. Database integration of 4923 publicly-available samples of breast cancer molecular and clinical data. AMIA Joint Summits Translational Science Proceedings. (2003) PMC3814460

Examples

```
data(curatedBreastDataExprSetList)
#what are all the names of the studies?
names(curatedBreastDataExprSetList)
#what is the dimension of the gene
#expression matrix for study GSE17705 from the JBI
#(as opposed to MDACC) site?
dim(exprs(curatedBreastDataExprSetList$study_17705_GPL96_JBI_Tissue_BC_Tamoxifen))
```

`filterAndImputeSamples`

Filter and Impute Samples

Description

A method that removes samples or genes with high NA rates and then KNN imputes remaining missing values.

Usage

```
filterAndImputeSamples(study, studyName = "study",
outputFile = "createTestTrainSetsOutput.txt",
impute = TRUE, knnFractionSize = 0.01, fractionSampleNAcutoff = 0.005,
fractionGeneNAcutoff = 0.01, exprIndex = "expr", classIndex,
sampleCol = TRUE, returnErrorRate = TRUE)
```

Arguments

`study`	A list, of minimally the gene expression or some molecular data matrix with keys (molecular features, such as genes) in the rows and patient samples in the columns and a keys list. It is assumed that the keys entity is named "keys", but in line with using this function for any type of molecular data, the exprIndex name in the list can be altered.
`studyName`	Character string to name the study. Useful in cases where looping over multiple datasets; output messages printed to the output file can then be identified by each individual study name.
`outputFile`	Output File for printing progress and stats on gene/sample filtering and data imputation. Include full directory if file should not be printed to current working directory.
`impute`	Impute data? A boolean TRUE or FALSE value. If FALSE, only genes and samples with high NA rates are removed, and the rest of the data is not imputed.

`knnFractionSize`
: What is the fraction of neighbors out of the total dataset to be used for knn impute nearest neighbor? This is translated into the "k" numeric magnitude in impute.knn() from the impute package. Default is .01, or 1

`fractionSampleNAcutoff`
: Max fraction of NAs allowed for a certain sample across all genes. Default is .005 (.005, or .5%, still captures a large number of gnees for a sample if there are tens of thousands of genes in the data matrix.)

`fractionGeneNAcutoff`
: Max fraction of NAs allowed for a certain gene across all samples. Default is .01. Thus, a certain gene cannot be missing in greater than 1% of patients. It is recommended that this threshold be increased for smaller datasets unless a user wants a gene to be removed that is missing in only 1 sample.

`exprIndex`
: Character string. List slot name for the data matrix, presumably an expression matrix.

`classIndex`
: Optional character string giving the list slot name for a phenotype vector or matrix if available. If phenotype/class data such as survival is already in the list, filtering out samples with high NA rates will result in the need to remove these samples from the phenotype data matrix; filterAndImputeSamples will appropriately filter out these samples from the phenoteyp data.

`sampleCol`
: Are samples in the columns of the expression matrix? If not, this function will first transpose the matrix to make sure impute.knn is running properly.

`returnErrorRate`
: Boolean TRUE or FALSE. If TRUE, a small amount of real expression data points are held out, and knn.impute is performed. The accuracy rate of the imputed values vs. the real values is returned. THis is helpful in early data analysis stages to determine whether KNN imputation is appropriate for your type of data. Default is FALSE to reduce computation time.

Value

A list containing the following objects:

`expr`
: original expression matrix

`exprFilterImputed`
: final filtered and imputed expression matrix

`keys`
: original keys

`keys`
: final filtered and imputed keys

`classes`
: original classes/phenotype data

`classes`
: final classes/phenotype data, removing any sample rows that were removed from the expression matrix after filtering.

Author(s)

Katie Planey <katie.planey@gmail.com>

Examples

```
#load up our datasets
data(curatedBreastDataExprSetList);

#just perform on one dataset as an example, GSE9893. This dataset does have NA
#values.
#highestVariance calculation make take a minute to run.
#create study list object.
study <- list(expr=exprs(curatedBreastDataExprSetList[[5]]),
keys=curatedBreastDataExprSetList[[2]]@featureData$gene_symbol,
phenoData=pData(curatedBreastDataExprSetList[[5]]))

filteredStudy <- filterAndImputeSamples(study, studyName = "study",
outputFile = "createTestTrainSetsOutput.txt", impute = TRUE,
knnFractionSize = 0.01, fractionSampleNAcutoff = 0.005,
fractionGeneNAcutoff = 0.01, exprIndex = "expr", classIndex="phenoData",
sampleCol = TRUE, returnErrorRate = TRUE)

#see output list names
names(filteredStudy)
#what is the imputation error fraction (rate)?
filteredStudy$errorRate
```

---

filterGenesByVariance *Filter genes by variance*

Description

A function that filters genes by variance; it can simply threshold out genes that are above or below a certain magnitude of variance, filter out genes that fall outside of a minimum and maximum percentile, or simply select the top N varying genes.

Usage

```
filterGenesByVariance(study, plotSaveDir = "~/", minVarPercentile,
maxVarPercentile=1, maxVar, minVar, exprIndex = "expr",
keysIndex = "keys", outputFile = "varCal.txt", plotVarianceHist = FALSE,
varMetric = c("everything", "all.obs", "complete.obs",
"na.or.complete", "pairwise.complete.obs"),
sampleCol = TRUE, numTopVarGenes)
```

Arguments

study	A list, of minimally the gene expression or some molecular data matrix with keys (molecular features, such as genes) in the rows and patient samples in the columns and a keys list. In line with using this function for any type of molecular data, the exprIndex name, and also the keysIndex name, in the list can be altered.

`plotSaveDir` If plotVarianceHist is TRUE, then the plotSaveDir is a character string specifying where this histogram plot should be saved.

`minVarPercentile` Minimum variance percentile. Must be provided in conjunction with maxVarPercentile to use percentiles to threshold genes.

`maxVarPercentile` Maximum variance percentile. Defaul is 1, i.e. 1%. Must be provided in conjunction with minVarPercentile to use percentiles to threshold genes.

`maxVar` If maxVar is provided, as opposed to minVarPercentile and maxVarPercentile, genes are removed that are above a certain variance magnitude. This may be useful if a user suspects very highly varying genes are actually technical noise/outliers. May be used in conjunction with minVar or in isolation.

`minVar` If maxVar is provided, as opposed to minVarPercentile and maxVarPercentile, genes are removed that are below a certain variance magnitude. This is helpful before running certain algorithms, such as the popular Combat batch normalization technique, that can throw errors if genes with extremely low variances are in the data matrix. May be used in conjunction with maxVar or in isolation.

`exprIndex` Character string. List slot name for the data matrix, presumably an expression matrix.

`keysIndex` Character string. List slot name for the feature names, presumably probes or gene names.

`outputFile` Output file for messages that print status of the filtering. Include full directory if file should not be printed to current working directory.

`plotVarianceHist` Plot the histogram of variances overall? Good for exploratory analyses to understand the distribution of variance across all data points. Default is FALSE to avoid saving a ggplot image for every function run.

`varMetric` Standard options taken from the base var() function. May be important if you have NA values in your data matrix; otherwise, "everything" is usually fine.

`sampleCol` Are samples in the columns of the expression matrix? If not, this function will first transpose the matrix, as the function assumes samples are in the columns features are in the rows.

`numTopVarGenes` A numeric value indicating the number of genes (features) to select; the function will only take this number of genes that have the highest variance across all genes.

Value

A list: output <- list(study=study,filteredStudy=filteredStudy,p=p);

`study` Original study list object

`filteredStudy` filteredStudy object, i.e. the gene expression and keys only for the desired filtered keys/features.

Note

Filtering by variance is equivalent to filtering on the coefficient of variation if data is logged. Further work includes automatically allowing the user to use the coefficient of variation as opposed to baseline variation for a threshold.

It is highly suggested you use filterAndImputeSamples() beforehand to remove any NA values, to avoid -Inf or NA variance calculations.

Author(s)

Katie Planey <katie.planey@gmail.com>

Examples

```
#load up our datasets
data(curatedBreastDataExprSetList);

#just perform on one dataset as an example, GSE1379.
#This dataset does not have NA values, which makes for a
#good example without extra preprocessing.
#highestVariance calculation make take a minute to run.
#create study list object.
study <- list(expr=exprs(curatedBreastDataExprSetList[[1]]),
keys=curatedBreastDataExprSetList[[1]]@featureData$gene_symbol)
#take top 100 varying genes

filterGeneStudy <- filterGenesByVariance(study, exprIndex = "expr",
keysIndex = "keys", outputFile = "./varCal.txt",
plotVarianceHist = FALSE,
varMetric = c("everything"), sampleCol = TRUE, numTopVarGenes=100)

#names of output
names(filterGeneStudy)
```

---

processExpressionSet *Post-process a normalized assayData in an ExpressionSet object*

Description

This function Post-processes a normalized assayData in an ExpressionSet object. Is is assumed the assay data is already baseline normalized (for example, for microarray data, this could mean quantile normalized and then logged.)

Usage

```
processExpressionSet(exprSet, outputFileDirectory = "./",numTopVarGenes,
minVarPercentile, maxVarPercentile = 1, minVar)
```

Arguments

exprSet	expressionSet S4 object with expression (assay) data, featureData and phenoData.
outputFileDirectory	Output file directory for messages that print status of post-processing the ExpressionSet.
minVarPercentile	Minimum variance percentile. Must be provided in conjunction with maxVarPercentile to use percentiles to threshold genes.
maxVarPercentile	Maximum variance percentile. Defaul is 1, i.e. 1%. Must be provided in conjunction with minVarPercentile to use percentiles to threshold genes.
minVar	If maxVar is provided, as opposed to minVarPercentile and maxVarPercentile, genes are removed that are below a certain variance magnitude. This is helpful before running certain algorithms, such as the popular Combat batch normalization technique, that can throw errors if genes with extremely low variances are in the data matrix. May be used in conjunction with maxVar or in isolation.
numTopVarGenes	A numeric value indicating the number of genes (features) to select; the function will only take this number of genes that have the highest variance across all genes.

Details

This function performs several post-processing tasks: filtering out genes and samples with high NA rates, imputing missing values, collapsing duplicated features/genes to make a unique feature list, removing any samples for which there is already a sample with the sample patient ID (duplicated samples), and filtering genes by variance. This function is a wrapper for the functions: filterAndImputeSamples(), collapseDupProbes(), removeDuplicatedPatients(), and filterGenesByVariance(). It is is run after initial dataset normalization, such as quantile normalization on microarray datasets.

Value

A post-processed S4 expressionSet. Tests are run to confirm the final S4 object is a valid ExpressionObject before it is returned.

Author(s)

Katie Planey <katie.planey@gmail.com>

Examples

```
#load up our datasets
data(curatedBreastDataExprSetList);

#just perform on one dataset as an example, GSE9893.
#This dataset does have NA values, so
#you'll see the impute.knn progress printed to the screen.
#also take only genes that fall in
```

```
#the variance percentiles between .75 and 1
#(i.e. top 75th percentile genes by variance.)

post_procExprSet <- processExpressionSet(exprSet=
curatedBreastDataExprSetList[[5]],
outputFileDirectory = "./",
minVarPercentile=.75, maxVarPercentile = 1)
```

`processExpressionSetList` *Process a list of S4 expressionSet objects.*

Description

A wrapper function for the post-processing function processExpressionSet() on a list of S4 expressionSet objects. This function is run after initial dataset normalization, such as quantile normalization on microarray datasets.

Usage

```
processExpressionSetList(exprSetList, outputFileDirectory = "./",
numTopVarGenes, minVarPercentile, maxVarPercentile = 1, minVar)
```

Arguments

`exprSetList`	List of S4 expression sets.
`outputFileDirectory`	Output file directory for messages that print status of post-processing the ExpressionSets.
`minVarPercentile`	Minimum variance percentile. Must be provided in conjunction with maxVarPercentile to use percentiles to threshold genes.
`maxVarPercentile`	Maximum variance percentile. Defaul is 1, i.e. 1%. Must be provided in conjunction with minVarPercentile to use percentiles to threshold genes.
`minVar`	If maxVar is provided, as opposed to minVarPercentile and maxVarPercentile, genes are removed that are below a certain variance magnitude. This is helpful before running certain algorithms, such as the popular Combat batch normalization technique, that can throw errors if genes with extremely low variances are in the data matrix. May be used in conjunction with maxVar or in isolation.
`numTopVarGenes`	A numeric value indicating the number of genes (features) to select; the function will only take this number of genes that have the highest variance across all genes.

Value

A list of processed S4 ExpressionSet objects.

Author(s)

Katie Planey <katie.planey@gmail.com>

See Also

processExpressionSet

Examples

```
## Not run:
#warning: takes a while to run! you're processing all datasets in the package!
#load up our datasets
data(curatedBreastDataExprSetList);

#just take top 5000 genes by variance
#this will post-process every dataset in the package
#to make them ready for downstream analyses.
proc_curatedBreastDataExprSetList <- processExpressionSetList(
exprSetList=curatedBreastDataExprSetList,
outputFileDirectory = "./", numTopVarGenes=5000)

## End(Not run)
```

---

removeDuplicatedPatients

Remove duplicated patient samples (samples from the same patient/column ID)

Description

Function to keep only 1 sample per patient (column ID) in the data matrix. Keeps the sample that has the overall highest variance.

Usage

```
removeDuplicatedPatients(exprMatrix,
outputFile = "duplicatedPatientsOutput.txt",
varMetric = c("everything", "all.obs", "complete.obs", "na.or.complete",
"pairwise.complete.obs"))
```

Arguments

exprMatrix	Expression matrix, with features in the rows and samples in the columns.
outputFile	Output file for messages that print status of removing duplicated samples. Include full directory if file should not be printed to current working directory.
varMetric	Standard options taken from the base var() function. May be important if you have NA values in your data matrix; otherwise, "everything" is usually fine.

Value

exprMatrix: the final data matrix with only 1 sample per patient ID.

Note

Suggestions are welcome for further ways to pick the best sample from samples from the same patient. No curatedBreastData matrices currently have samples that share the same patient ID, but this function is especially useful for say TCGA data, where this is often the case.

It is suggseted one imputes missing values using the filterAndImpute function before running this function to avoid -Inf and NA values in the variance calculations.

Author(s)

Katie Planey <katie.planey@gmail.com>

Examples

```
#No curatedBreastData has duplicated samples,
#but we can still run this function on one of the datasets:
#load up our datasets
data(curatedBreastDataExprSetList);

#This dataset does not have NA values, which makes for a good example without
#extra pre-processing.
outputMatrix <- removeDuplicatedPatients(exprMatrix=
exprs(curatedBreastDataExprSetList[[1]]),
outputFile = "./duplicatedPatientsOutput.txt", varMetric = c("everything"))
#final dimensions - unchanged in this case with
#no samples sharing the same patient ID.
dim(outputMatrix)
```

Index