Package: iNETgrate (via r-universe)

November 14, 2026

Type Package

Title Integrates DNA methylation data with gene expression in a single gene network

Version 1.10.0

Date 2023-03-24

biocViews GeneExpression, RNASeq, DNAMethylation, NetworkInference, Network, GraphAndNetwork, BiomedicalInformatics, SystemsBiology, Transcriptomics, Classification, Clustering, DimensionReduction, PrincipalComponent, mRNAMicroarray, Normalization, GenePrediction, KEGG, Survival

Depends R (>= 4.3.0), BiocStyle (>= 2.18.1)

Description The iNETgrate package provides functions to build a correlation network in which nodes are genes. DNA methylation and gene expression data are integrated to define the connections between genes. This network is used to identify modules (clusters) of genes. The biological information in each of the resulting modules is represented by an eigengene. These biological signatures can be used as features e.g., for classification of patients into risk categories. The resulting biological signatures are very robust and give a holistic view of the underlying molecular changes.

Imports SummarizedExperiment, GenomicRanges (>= 1.24.1), stats, WGCNA, grDevices, graphics, survival, igraph, Pigengene (>= 1.19.26), Homo.sapiens, glmnet, caret, gplots, minfi, matrixStats, Rfast, tidyr, tidyselect, utils

Suggests knitr, org.Hs.eg.db, org.Mm.eg.db, IlluminaHumanMethylation450kanno.ilmn12.hg19, AnnotationDbi, sesameData, TCGAbiolinks (>= 2.29.4)

License GPL-3

BugReports https://github.com/Bioconductor/BiocManager/issues

VignetteBuilder knitr

NeedsCompilation no

Author Isha Mehta [aut] (<https://orcid.org/0000-0002-6009-0787>), Ghazal Ebrahimi [aut], Hanie Samimi [aut], Habil Zare [aut, cre] (<https://orcid.org/0000-0001-5902-6238>)

Maintainer Habil Zare <zare@u.washington.edu>

Config/pak/sysreqs cmake libfontconfig1-dev libfreetype6-dev libglpk-dev make libbz2-dev libicu-dev liblzma-dev libpng-dev libuv1-dev libxml2-dev libssl-dev libx11-dev xz-utils zlib1g-dev

Repository Bioconductor 3.23

Date/Publication 2026-04-28 13:01:02 UTC

RemoteUrl https://github.com/bioc/iNETgrate

RemoteRef RELEASE_3_23

RemoteSha 27d5942df5b6f4d26c7d3793d7e2702baef6350f

Contents

iNETgrate-package . . . 3
accelFailAnalysis . . . 4
analyzeSurvival . . . 6
bestInetgrator . . . 10
cleanAllData . . . 11
computEigengenes . . . 13
computEigenloci . . . 15
computeInetgrator . . . 17
computeUnion . . . 18
coxAnalysis . . . 20
createLocusGene . . . 21
distanceToTss . . . 22
downloaData . . . 24
electGenes . . . 25
filterLowCor . . . 27
findAliveCutoff . . . 28
findCore . . . 29
findTcgaDuplicates . . . 31
iNETgrate . . . 32
inferEigengenes . . . 36
makeNetwork . . . 37
plotKM . . . 39
plotLociNum . . . 40
plotLociTss . . . 41
prepareSurvival . . . 42
preprocessDnam . . . 44
sample2pat . . . 45
sampleData . . . 47
toyCleanedAml . . . 48
toyComputEloci . . . 49
toyEigengenes . . . 51
toyRawAml . . . 52

Index **55**

`iNETgrate-package` *Integrates DNA methylation data with gene expression in a single gene network*

Description

The iNETgrate package can be used to build a correlation network in which nodes are genes. DNA methylation and gene expression data are integrated to define the connections between genes. This network is used to identify modules (clusters) of genes. The biological information in each of the resulting modules is represented by an eigengene. These biological signatures can be used as features e.g., for classification of patients into risk categories. The resulting biological signatures are very robust and give a holistic view of the underlying molecular changes.

Details

Package:	iNETgrate
Type:	Package
Version:	0.5.24
Date:	2021-08-25
License:	GPL (>= 2)

Author(s)

Isha Mehta, Ghazal Ebrahimi, Hanie Samimi, and Habil Zare

Maintainer: Habil Zare <zare@uthscsa.edu>

See Also

`iNETgrate`, `Pigengene-package`, `GDCdownload`, `WGCNA::blockwiseModules`

Examples

```
?iNETgrate
```

`accelFailAnalysis` *Accelerated Failure Analysis*

Description

Performs accelerated failure time model and prediction on each combination of features (e.g., selected modules).

Usage

```
accelFailAnalysis(Data, survival, time2day, eventCol="Dead", riskCol="Risk1",
                weight=NULL, minRecall4L=0.2, minRecall4H=0.1, until=10, xmin1=0,
                   xmax1=max(survival[, "Time"])*(time2day/365), predType="lp",
                   doTitle=TRUE, resultPath,
                   risk2col=c("Low"='green', "Int"='blue', "High"='red'),
                   doAddConfTable=FALSE, favRisk="High",
                   riskLevel, subSet=NULL, pvalDigits=0,
                   ylabKm="Survival probability", verbose=0)
```

Arguments

`Data`	A matrix of features (eigengenes) values, where rownames are patient IDs and column names are features.
`survival`	A matrix or a data frame with minimum of four columns namely: "PatientID", `eventCol`, "Time", and `riskCol`. The rownames are the same as "PatientID".
`time2day`	A numeric value, that helps calculating survival time in days. E.g. if the "Time" column in survival is in days, time2day=1, else set it to 30 if "Time" is in months.
`eventCol`	The name of the event column in the `survival` data. Values in this column must be 0 or 1 (Alternatively, `TRUE` and `FALSE` values are also supported.)
`riskCol`	The name of the risk column in the `survival` data. Values in this column must be "High", "Int", or "Low".
`weight`	A named numeric vector that determines the weight of each eigengene. Here, names correspond to features and values are weights. If `NULL`, all features would have the same weight.
`minRecall4L`	A numeric value of the aimed recall cut–off for low–risk predictions, which may not be reached exactly. The higher, the more low-risk cases would be identified, but the risk may be relatively higher.
`minRecall4H`	A numeric value of the aimed recall cut–off for high–risk predictions, which may not be reached exactly. The higher, the more high-risk cases would be identified, but the risk may be relatively lower.
`until`	A numeric value that sets minimum survival time (in years) for an alive case to be considered as low risk. Set it to `NULL` if you want to create km-plots based on `riskCol`.
`xmax1`	A numeric value giving the maximum time (in years) for plotting the K–M curves.

xmin1	A numeric value giving the minimum time (in years) for plotting the K–M curves.
predType	A character string that takes type input from predict function in the stats package.
doTitle	Accepts a Boolean value, where TRUE (default) indicates title will be added to plots.
resultPath	A character string that gives path to save results.
risk2col	A named character vector, that assigns color to the risk groups in KM-plots. Names must be in the riskCol, and values are colors e.g., risk2col=c(High="red", Low="green", Int="blue").
doAddConfTable	Accepts a Boolean value, where FALSE (default) indicates confidence table for risk information will not be created.
favRisk	A string value that determines a predicted risk group that will be further stratified based on the survival data risk groups. E.g., "High" leads to plotting the survival data risk groups for the subset of data that are identified high risk based group on network analysis.
riskLevel	A named character vector where values are risk group names (a subset of unique(riskCol)), and names are "Low", "Int" and "High". This mapping is necessary for computing the confusion matrix.
subSet	A string that determines a risk group from the survival data that will be further stratified based on the predicted risk groups e.g., "Int" leads to plotting the predicted risk groups for the subset of data that are determined intermediate risk group based on survival data.
pvalDigits	A numeric value determining the number of decimal digits to be printed on the KM plots for p-value.
ylabKm	A character vector determining Y–axis label for KM–plot.
verbose	The integer level of verbosity. 0 means silent, and higher values produce more details of the computation.

Value

A list with following components:

subsets	A list of predicted risk groups information.
kmPVals	A numeric vector of p-values from KM survival plots.

References

R Core Team (2019). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. stats.

Therneau, T. (2020). A Package for Survival Analysis in R. R package version 3.1-12, survival.

Therneau, T.M., Grambsch, P.M., (2000). Modeling Survival Data: Extending the Cox Model. *Springer, New York. ISBN 0-387-98784-3*.

See Also

predict, analyzeSurvival, plotKM

Examples

```
data(toyCleanedAml)
data(toyEigengenes)

s1 <- toyCleanedAml$survival
head(s1) ## print

afOut <- accelFailAnalysis(Data=toyEigengenes, survival=s1,
                           time2day=1, riskCol="Risk1", until=1, xmax1=15,
                           resultPath=tempdir(),
                           riskLevel=c("Low"="Low", "Int"="Int", "High"="High"),
                           subSet="Int", verbose=1)
```

---

analyzeSurvival *Survival analysis*

Description

It first performs Cox analysis to select up to 3 modules that result in the best risk group categorization based on network analysis. Then, an Accelerated Failure Time (AFT) analyzeSurvival is done using the selected modules.

Usage

```
analyzeSurvival(survival, eventCol="Dead", riskCol="Risk1", Data=NULL,
            excludeSamples=rownames(survival)[as.numeric(survival[,"Time"])<=0],
                  mus, netPath, outPath, favRisk="High", subSet=NULL,
                  doCox=TRUE, eOrMs="e", time2day=1, until=1,
                  xmax1=max(survival[, "Time"]) * (time2day/365),
            xmin1=0, minRecall4L=0.2, minRecall4H=0.05, verbose=0, doDebug=FALSE)
```

Arguments

survival	A matrix or a data frame with at least four columns: "PatientID", "Time", event, and riskCol (see the event and riskCol arguments). The row names are the same as "PatientID".
eventCol	The name of the event column in the survival data. Values in this column must be 0 or 1 (Alternatively, TRUE and FALSE values are also supported.)
riskCol	The name of the risk column in the survival data. Values in this column must be "High", "Int", or "Low".
Data	The feature matrix, e.g., patients on rows and eigengenes on columns. Exactly one of netPath or Data must not be NULL.

excludeSamples — A character vector of samples that will be excluded from the analysis. Must be a subsets of row names of the feature matrix Data. With the default value, all the samples with a non-positive survival time will be excluded.

mus — A numeric vector of mu values to be used for analysis.

netPath — A character string describing path to saved output from makeNetwork and computEigengenes functions. Exactly one of netPath or Data must not be NULL.

outPath — A character string describing the path to save plots and output.

favRisk — A risk category predicted by the network classifier that shall be compared to the input Labels risk categories. The value must be one of these three: "High", "Int", or "Low".

subSet — A risk category in Labels that shall be compared to the risk category predicted by network classifier. The value must be one of these three: "High", "Int", or "Low".

doCox — Accepts a Boolean value, where TRUE (default) indicates the Cox analysis will be performed for module selection. Otherwise, all modules will be used for AFT, which can lead to overfitting.

eOrMs — A character vector to determine which eigengenes (already stored at netPath) to be used as input data. E.g., c("m", "e", "em"), where "e" means "only gene expression", "m" means "DNA methylation", and "em" means the combination of both data types is used to compute an eigengene for a module. See the note below.

time2day — A numeric value that aids calculating survival time (in days). E.g., If the "Time" column in survival data frame is in days, then time2day=1, set it to 30 if "Time" is in months, etc.

until — A numeric value describing the minimum survival time (in years) for a case to be considered as low risk, e.g., if an AML patient is known to be alive for at least 2 years after diagnosis, and not dead at the last follow up time, then we can consider this case low–risk.

xmax1 — A numeric value determining the maximum time (in years) for plotting the K–M curves.

xmin1 — A numeric value determining the minimum time (in years) for plotting the K–M curves.

minRecall4L — A numeric value of the aimed recall cut–off for low–risk predictions, which may not be reached exactly. The higher, the more low–risk cases would be identified, but the risk may be relatively higher. Do not change the default if you are not an expert.

minRecall4H — A numeric value of the aimed recall cut–off for high–risk predictions, which may not be reached exactly. The higher, the more high–risk cases would be identified, but the risk may be relatively lower. Do not change the default if you are not an expert.

verbose — An integer level of verbosity. 0 means silent, and higher values produce more details of the computation.

doDebug — A Boolean that aids in code debugging by developers.

Details

When Data is not NULL, set netPath=NULL. When netPath is NULL, eorMs value is ignored. In the iNETgrate pipeline, we pass a value to netPath for training set analysis and set it to NULL for test set (i.e., validation) data.

Value

A list with following components:

mus	A vector of mu values same as the input.
bestPvalues	A data frame with three columns namely: "p-values", "mu", and "features". The best p–value and module names are identified for each mu value.
bestModules	A matrix of ranked eigengenes from high to low based on Cox analysis for each mu.
kmPVals	A vector of best p–values for each mu.
concordanceV	A vector of best concordance for each mu.
bestPvaluePlot	A plot of best p–values for each mu.
afts	The result of accelFailAnalysis for each mu values.
aftPaths	A vector of characters named with mus.
baseCriteria	A named vector as the same as the input.
timeTaken	The time to complete analyzeSurvival.

Note

If computEigengene is ran with genExpr=NULL, then "e" and "em" cannot be used here. Similarly, if computEigengene(eigenloci=NULL, ...), then "m" and "em" cannot be used here. If in the next step of the analysis the eigengenes will be combined in a linear combination (common), then the "em" eigengene is redundant and not needed.

References

R Core Team (2019). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. stats.

Therneau, T. (2020). A Package for Survival Analysis in R. R package version 3.1-12, survival.

Therneau, T.M., Grambsch, P.M., (2000). Modeling Survival Data: Extending the Cox Model. *Springer, New York. ISBN 0-387-98784-3*.

See Also

makeNetwork, computEigengenes, prepareSurvival, inferEigengenes, accelFailAnalysis, plotKM

Examples

```
## Preparing data:
data(toyCleanedAml)
data(toyComputEloci)
eigenloci <- toyComputEloci$eigenloci

patientLabel <- as.matrix(toyCleanedAml$survival)[,"Risk1"]
dimnames1 <- list(c("High", "Low"), c("Risk1", "Risk2"))
netPath <- file.path(tempdir(), "net")
dir.create(netPath, showWarnings=FALSE)
print(paste("Results will be saved in:", netPath))
print(Sys.time())

## Make the network:
madeNetwork <- makeNetwork(genExpr=toyCleanedAml$genExpr, eigenloci=eigenloci,
                           geNames=colnames(eigenloci), mus=c(0.6),
                           outPath=netPath, verbose=1)

## Compute eigengenes:
## More samples maybe included if eigenloci=NULL below. See the note in ?computEigengenes.
eigengenes <- computEigengenes(genExpr=toyCleanedAml$genExpr,
                                eigenloci=eigenloci, netPath=netPath,
                                geNames=colnames(eigenloci), Labels=patientLabel,
                                Label1="Low", Label2="High",
                                mus=c(0.6), survival=toyCleanedAml$survival,
                                mu2modules=madeNetwork$mu2modules)

## Survival analysis:
survRes <- analyzeSurvival(survival=toyCleanedAml$survival,
                           favRisk="High", subSet="Int", mus=0.6,
                           netPath=netPath, outPath=netPath,
                           eOrMs="e", xmax1=15, xmin1=0, verbose=1)

## Getting the best inetgrator:
bestPval <- survRes$bestPvalues ## Best p-values per mu value
## Alternatively, it can be read from the saved csv file:
bestPval <- read.csv(file=file.path(netPath, "bestPvalues_e.csv"), header=TRUE)

inetgrator <- bestInetgrator(bestPvalues=bestPval,
                             usefuLoci=toyComputEloci$usefuLoci,
                             lociPigen=toyComputEloci$lociPigen, netPath=netPath)

## Infer eigengenes for a validation dataset:
valExpr <- t(toyCleanedAml$genExpr)[1:10, ]
##^ In a real applications, the validation data must be independent from the train data.
inferred <- inferEigengenes(inetgrator=inetgrator, expr=valExpr)

## Plot the inferred eigengenes:
toPlot <- inferred$eigengeneS
plToPlot <- as.data.frame(patientLabel[rownames(toPlot)])
Pigengene::pheatmap.type(Data=scale(toPlot), annRow=plToPlot, cluster_cols=FALSE)
```

```
print(Sys.time())
```

`bestInetgrator` *Identifies the bestInetgrator*

Description

It creates the best integrator object by collecting all the weights corresponding to the best `mu` value from the results of the training dataset. The inetgrator can then be used to infer the eigengenes in validation datasets using the `inferEigengenes` function.

Usage

```
bestInetgrator(bestPvalues, usefuLoci, lociPigen, netPath, verbose=0)
```

Arguments

`bestPvalues`	A data frame with at least three columns named: "pvalue", "mu", and "features". This is similar to the `bestPvalues` output from `analyzeSurvival`.
`usefuLoci`	A list of vectors where the names are genes and the values are the corresponding loci that contribute to computing eigenloci. This is an output of `computEigenloci`.
`lociPigen`	A list of genes (same as `usefuLoci`'s genes and their related Pigengene object (based on their mapped loci). This is an output of `computEigenloci`.
`netPath`	A string determining the path to the network folder, where the pigengene and eigengene objects saved for selected modules of each mu.
`verbose`	An integer level of verbosity. `0` means silent, and higher values produce more details of the computation.

Value

The output of the `computeInetgrator` function corresponding to the best `mu` based on the analyzeSurvival.

See Also

`inferEigengenes`, `analyzeSurvival`, `computEigenloci`, `computeInetgrator`, `pigengene-class`

Examples

```
## See the analyzeSurvival() for an example.
?analyzeSurvival()
```

cleanAllData *Cleans and preprocesses data*

Description

This is a wrapper function that executes preprocessDnam and prepareSurvival functions to preprocess and impute DNA methylation and clinical data respectively. It also creates the sample2pat mappings for gene expression and DNA methylation data.

Usage

```
cleanAllData(genExpr, genExprSampleInfo, rawDnam, rawDnamSampleInfo=NULL,
          sampleCol="barcode", patientCol="patient", sampleTypeCol="shortLetterCode",
               sampleTypesIn=NULL, savePath, isFfpe=c(FALSE), annLib="Auto",
          clinical, patientIDCol="bcr_patient_barcode", eventCol="vital_status",
               event="Dead", timeCol="days_to_last_followup", riskFactorCol,
               riskCatCol=NULL, riskHigh="High", riskLow="Low", verbose=0)
```

Arguments

genExpr	A matrix of gene expression data where rows are genes and columns are samples.
genExprSampleInfo	A data frame that gives information about the gene expression samples. It contains at least two columns determining patient IDs and the corresponding sample IDs. The row names may or may not be patient IDs.
rawDnam	A matrix of beta values or a summarized experiment object similar to the output of GDCprepare. Rows are genes and columns are samples.
rawDnamSampleInfo	A data frame that gives information about the DNA methylation samples. It contains at least two columns determining patient IDs and the corresponding sample IDs. The row names may or may not be patient IDs.
sampleCol	The name of the column in the sample information data frames that contains sample IDs.
patientCol	The name of the column in the sample information data frames that contains patient IDs.
sampleTypeCol	The name of the column in the sample information data frames that contains sample type information. See sampleTypesIn.
sampleTypesIn	A character vector determining the sample types to be included in the iNETgrate analysis e.g., c("TP", "NT") in TCGA data, where TP and NT indicate Primary Tumor and Non-Tumor adjacent samples, respectively.
savePath	A string determining the path to the folder to save plots.
isFfpe	A character vector that takes Boolean values to indicate inclusion or exclusion of FFPE sample. c(FALSE, TRUE) will include all samples, TRUE will include only FFPE samples, and FALSE will exclude FFPE samples. Set it to NULL to disable it.

annLib	A string that determines annotation (reference) library to be used. Default is "IlluminaHumanMethylation450kanno.ilmn12.hg19".
clinical	A matrix or a data frame of clinical data, with minimum four columns that gives information about patient IDs, event occurance, survival or follow up time, and risk factors. The rownames are patient IDs.
patientIDCol	A string determining the column in clinical data that lists patient or case IDs, e.g., "bcr_patient_barcode" in sample AML data
eventCol	A string determining the column in clinical data that lists the event. The event column must be a binary column e.g., "vital_status" column in sample AML data where two possible events are "Dead" and "Alive".
event	A string determining an event of interest from eventCol in clinical data. E.g., "Dead" is the event of interest in the example data.
timeCol	A string determining the column in clinical data storing survival time information, e.g. "days_to_last_followup" in sample AML data
riskFactorCol	A string determining the column in clinical data that describes the risk factors associated to risk levels.
riskCatCol	A string determining the column in clinical data that lists risk levels of the disease.
riskHigh	A character vector that describes high risk group of patients in clinical data. See section "Details".
riskLow	A character vector that describes low risk group of patients in clinical data. See section "Details".
verbose	An integer that sets the level of verbosity. 0 means silent, and higher values produce more details of the computation.

Details

When riskCatCol is NULL, riskHigh and riskLow are expected to have values that belong to riskFactorCol. When riskCatCol is defined, riskHigh and riskLow are expected to have values that belong to riskCatCol. All riskCatCol, riskHigh and riskLow cannot be NULL at the same time.

Value

A list with following components:

survival	A data frame with a minimum of five columns namely: "PatientID", event, "Time", "Risk1", and riskFactorCol. The row names are the same as "PatientID".
genExpr	A data frame of gene expression data where rows are genes and columns are (tagged) patient IDs.
sample2patient	A list of named vectors.
patient2type	A list of named vectors.
dnam	A matrix of dnam beta values that is cleaned and imputed where rownames are probe IDs and column names are (tagged) patient IDs.

locus2gene — A data frame where rownames are probeIDs and at least 4 columns that contain information about probeIDs, geneIDs, genomic coordinates, and chromosome information.

See Also

prepareSurvival, preprocessDnam, sample2pat

Examples

```
data(toyRawAml)
riskCatCol <- "acute_myeloid_leukemia_calgb_cytogenetics_risk_category"
riskFactorCol <- "cytogenetic_abnormalities"
cleaned <- cleanAllData(genExpr=toyRawAml$genExpr,
                        genExprSampleInfo=toyRawAml$genExprSampleInfo,
                        rawDnam=toyRawAml$rawDnam, savePath=tempdir(),
                        clinical=toyRawAml$clinical, riskCatCol=riskCatCol,
                        riskFactorCol=riskFactorCol, riskHigh="Favorable",
                        riskLow="Poor", verbose=1)
```

---

computEigengenes *Compute eigengenes of combined modules.*

Description

Computes eigengenes (features) for network modules obtained from expression and dnam data combined.

Usage

```
computEigengenes(genExpr=NULL, eigenloci=NULL, netPath, geNames,
                 Labels, Label1, Label2, mus, combiningMu=NA,
                 survival, event="Dead", doIgnoreNas=FALSE,
                 mu2modules, doWarn=TRUE, verbose=0)
```

Arguments

genExpr — A matrix of gene expression values where row names are gene IDs, and column names are patient IDs.

eigenloci — A matrix of eigenloci where row names are patient IDs, and column names are gene IDs. Set this to NULL if you do not want to compute eigengenes using DNA methylation data. See Note.

netPath — A string that determines the path to a folder where output object combinedNetwork is saved.

geNames — A character vector of selected genes that become the nodes of the network. Both gene expression and eigenloci data must be available for these genes e.g., unionGenes (output of electGenes).

Labels	A named character vector that determines the risk group for each patient. Here, the names are the patient IDs and the values are the risk group labels. It is needed for sample balancing when computing the eigenloci.
Label1	One of the values from Labels vector that defines a patient group to be used to balance data e.g., low–risk group defined as "Favorable" in toyAmlData.
Label2	One of the values from Labels vector that defines a patient group to be used to balance data e.g., high–risk group defined as "Poor" in toyAmlData.
mus	A numeric value or vector. Lambda values that are used for network construction.
combiningMu	A numeric vector determining the mu value(s) used for eigengene computation. If set to NA, the same value that was used for network construction, will be used for eigengene computation.
survival	A data frame where row names are patient IDs and must have minimum of two columns: "Time" (to event) and event.
event	A string determining the name of the event column in the survival data. It stores binary values 1 and 0 that correspond to the occurance or non-occurance of the event respectively.
doIgnoreNas	A Boolean with a default value of False.
mu2modules	A matrix where rows are mus and values are module information. It is an output of makeNetwork function.
doWarn	Logical determining warnings should be issued.
verbose	An integer level of verbosity. 0 means silent, and higher values produce more details of the computation.

Value

A list with following components:

patients	A character vector of patientIDs that are included in eigengene calculations.
lcombine	A numeric vector where names are "e", "m", and "em". The value corresponding to "em" is same as mus value. If genExpr and eigenLoci inputs are available, "e" and "m" are set to 0 and 1 respectively.
eigengeneFiles	A matrix where rows are mus, columns are lcombine, and values are paths to eigengene RData files.

Note

If eigenloci!=NULL, eigengenes will be computed only for those cases that have both gene expression and DNA methylation data, which can lead to fewer samples for the subsequent analysis. Because eigengenes computed using DNA methylation usually highly correlate with eigengenes computed using gene expression, setting eigenloci=NULL can be reasonable in some datasets to include more samples in the subsequent analysis. Either way, the DNA methylation data have been already used to build the network in the previous steps of the iNETgrate pipeline.

References

Foroushani, A. *et al.* (2016) Large-scale gene network analysis reveals the significance of extracellular matrix pathway and homeobox genes in acute myeloid leukemia: an introduction to the Pigengene package and its applications, *BMC MedicalGenomics*.

See Also

makeNetwork compute.pigengene, project.eigen

Examples

```
data(toyCleanedAml)
data(toyComputEloci)
eigenloci <- toyComputEloci$eigenloci
patientLabel <- as.matrix(toyCleanedAml$survival)[,"Risk1"]

# Path to the network
netPath <- file.path(tempdir(), "net")
dir.create(netPath, showWarnings=FALSE)
madeNetwork <- makeNetwork(genExpr=toyCleanedAml$genExpr, eigenloci=eigenloci,
                           geNames=colnames(eigenloci), mus=c(0.6),
                           outPath=netPath, verbose=0)

## Usually eigenloci=NULL, see the note above.
eigengenes <- computEigengenes(genExpr=toyCleanedAml$genExpr,
                                 eigenloci=eigenloci, netPath=netPath,
                                 geNames=colnames(eigenloci), Labels=patientLabel,
                                 Label1="Low", Label2="High",
                                 mus=c(0.6), survival=toyCleanedAml$survival,
                                 mu2modules=madeNetwork$mu2modules)
```

---

computEigenloci *Computes an eigenloci for each gene*

Description

An eigenloci of a gene is a weighted average (PC1) of beta values of the most correlated loci (i.e., core) of that gene.

Usage

```
computEigenloci(dnam, locus2gene, geNames, Labels, Label1,
                  Label2, plotPath=NULL, genesColName="Gene_Symbol",
                coordinatesColName="Genomic_Coordinate", lociColName="probeID",
                  dnamGene=NULL, doDebug=FALSE, verbose=0)
```

Arguments

dnam	A matrix of beta values where row names are probe IDs and column names are patient IDs.
locus2gene	A data frame with loci information where row names are probe IDs and must have at least 4 columns that contain probe IDs, gene IDs, genomic coordinates, and chromosome information.
geNames	A character vector of selected genes for which eigenloci needs to be obtained e.g. "unionGenes", the output of electGenes. If NULL (default), eigenloci for all the genes in the gene IDs column of the locus2gene dataframe will be calculated.
Labels	A named character vector where names are tagged patient IDs and values are group label. It is needed for sample balancing when computing the eigenloci. See Label1 and Label2.
Label1	One of the values from Labels vector that defines a patient group to be used to balance data e.g., low–risk group defined as "Favorable" in toyAmlData.
Label2	One of the values from Labels vector that defines a patient group to be used to balance data e.g., high–risk group defined as "Poor" in toyAmlData.
plotPath	A string that determines the path to save plots.
genesColName	A string determining the name of the column in the locus2gene data frame that stores gene IDs.
coordinatesColName	A string determining the name of the column in the locus2gene data frame that stores genomic co-ordinates.
lociColName	A string determining the name of the column in the locus2gene data frame that stores probeIDs.
dnamGene	A character vector of selected genes based on correlating survival time with dnam data.
doDebug	A Boolean that aids in code debugging by experts only. Otherwise, leave it as FALSE (default).
verbose	The integer level of verbosity. 0 means silent, and higher values produce more details of the computation.

Details

For each genes, first its core (i.e., the most connected cluster of its loci) is identified using the findCore function. Then, an eigenloci for that gene is computed using PCA on the core loci, which optimizes the weights of each locus in the core.

Value

eigenloci	A matrix of eigenloci values where row names are patient IDs and column names are gene IDs.
usefuLoci	A list of character vectors, where a gene is an item of the list that stores the corresponding loci information.

`lociPigen` A list of Pigengene objects. Names are genes.

`distanceToTss` A list of distances of all selected loci to Transcription Start Sites (TSS).

`distanceToTssDnam`
A list of distances of loci selected based on dnam data to TSS.

References

Amir Foroushani *et al.* (2016) Large-scale gene network analysis reveals the significance of extracellular matrix pathway and homeobox genes in acute myeloid leukemia: an introduction to the Pigengene package and its applications, *BMC MedicalGenomics*.

See Also

`compute.pigengene`, `project.eigen`, `distanceToTss`, `electGenes`

Examples

```
data(toyCleanedAml)
plotPath <- file.path(tempdir(), "plots")
print(paste("Results will be saved in:", plotPath))
dir.create(plotPath, showWarnings=FALSE)
unionGenes <- c("BAI1", "CAPNS1", "CHD1", "CYC1", "EXT1", "FYB", "GFRA1", "HLX",
                "IDH3B", "PSMD8", "SGTA", "SLC1A5", "SSFA2", "TFAM", "TGIF1",
                "UBC", "ZNF70", "ST8SIA4", "ZNF485", "FOXN3", "HCG4", "FOXK1",
                "C2orf44")

survivalAml <- toyCleanedAml$survival
patientLabel <- setNames(as.character(survivalAml[,"Risk1"]), rownames(survivalAml))
inBoth <- intersect(colnames(toyCleanedAml$dnam), names(patientLabel))
dnamData <- toyCleanedAml$dnam[ ,inBoth]
eigenLoci <- computEigenloci(dnam=dnamData, locus2gene=toyCleanedAml$locus2gene,
                             geNames=unionGenes,
                           Labels=patientLabel[is.element(names(patientLabel), inBoth)],
                             Label1="Low", Label2="High", plotPath=plotPath,
                             verbose=1)
```

`computeInetgrator` *Computes an inetgrator*

Description

It computes the integrator for an arbitrary `mu` value. For expert users only. Others can use the `bestInetgrator` function.

Usage

```
computeInetgrator(mu, usefuLoci, genePigens, lociPigen, moMe)
```

Arguments

mu	The numeric value(s) used in network combining in the training dataset. It must be between 0 and 1.
usefuLoci	A list of vectors where the names are genes and the values are the corresponding loci that contribute to computing eigenloci. This is an output of computEigenloci.
genePigens	A list of pigengenes objects with names from c("m", "e", "em") as described in eOrMs in analyzeSurvival.
lociPigen	A list of Pigengene object named with genes in the same way as in usefuLoci. This is an output of computEigenloci.
moMe	A character vector of the selected eigengenes to be used e.g., c("ME51.m", "M55.e", "M46.em"). Each eigengene starts with "ME" and module number followed by a dot and then the type of data used to compute the eigengene. See eOrMs in analyzeSurvival.

Value

A list with following components:

mu	A numeric value used in the network construction of the training dataset. It ranges between 0 and 1.
pigengenes	A list of one or more pigengene-class objects, each having the information on genes in the eigengene. The names may include "m", "e", and "em" as described in eOrMs in analyzeSurvival.
modules	A list of selected data for each module including the weights of genes, the loci for each gene, and if the number of loci is more than 1, a Pigengene object for that particular gene.

See Also

computEigenloci, bestInetgrator, pigengene-class

Examples

```
## See the analyzeSurvival() for an example.
?analyzeSurvival()
```

---

computeUnion *Computes union gene set*

Description

Makes a union set of genes that have high correlation with survival time or vital status based on their expression or methylation levels.

Usage

```
computeUnion(Genes, selectedGenes, loci, selectedLoci, locus2gene,
             doAlLoci=FALSE, genesColName="Gene_Symbol", lociColName="probeID",
              verbose=0)
```

Arguments

Genes	A character vector of all gene IDs from the gene expression data.
selectedGenes	A character vector of selected gene IDs that show high correlation with survival time or vital status based on their expression values.
loci	A character vector of all loci or probe IDs obtained after preprocessing methylation as detailed in preprocessDnam.
selectedLoci	A character vector of loci or probe IDs that have high correlation with survival time or vital status based on their beta values.
locus2gene	A data frame or a matrix with at least two columns that contain information about probe IDs and the corresponding gene IDs. The rownames are probe IDs.
doAlLoci	A Boolean value with a default value of FALSE. If TRUE, a gene with any levels of DNA methylation loci will be included in the output union even if they do not correlate with survival data.
genesColName	A string determining the name of the column in locus2gene matrix that carries gene ID information.
lociColName	A string determining the name of the column in locus2gene matrix that carries probe ID information.
verbose	An integer that sets the level of verbosity. 0 means silent, and higher values produce more details of the computation.

Value

A list comprising of following objects:

unionGenes	A character vector of selected genes that show high correlation with survival time or vital status based on expression or methylation data.
unionLoci	A character vector of loci that are associated with unionGenes set and also have appropriate methylation data.
dnamSelectGenes	A character vector of selected genes that have correlation with survival time or vital status based on mehtylation data only.

See Also

createLocusGene, electGenes, preprocessDnam

Examples

```
data(toyCleanedAml)

highCorGenes <- c("TRPV4", "CHN2", "WFDC1", "XPO1", "HTR7", "ABCC1", "TMCO3",
                  "FBXO22", "VWCE", "KLHL36", "APBA2", "IRF2BP2")
highCorLoci <- c("cg00044050", "cg00086266", "cg00112238", "cg00321478",
                 "cg00371239", "cg00375235", "cg00390484", "cg00424690",
                 "cg00448143", "cg00452755")
unionSet <- computeUnion(Genes=rownames(toyCleanedAml$genExpr),
                         loci=rownames(toyCleanedAml$dnam),
                         selectedGenes=highCorGenes, selectedLoci=highCorLoci,
                         locus2gene=toyCleanedAml$locus2gene, verbose=2)
```

---

coxAnalysis *Cox analysis*

Description

Function to perform Cox regression analysis.

Usage

```
coxAnalysis(dataCategory="training", inputTime, inputEvent, inputMatrix,
            outPath, weight=NULL, verbose=0)
```

Arguments

dataCategory	A string describing the type of data analyzed e.g., "Training data", "validation data", or "all data".
inputTime	A named numeric vector where names are patient IDs and values are survival time (in years).
inputEvent	A named binary vector where names are patient IDs and values are `0` and `1`. Here, `1` indicates the occurance of event and vice versa.
inputMatrix	A matrix of projected eigengenes for each sample, where row names are case IDs and column names are modules.
outPath	Output path for coxAnalysis.
weight	Same as `weights` argument in `glmnet`.
verbose	The integer level of verbosity. `0` means silent, and higher values produce more details of the computation.

Value

A character vector of ranked input features (eigengenes) from high to low based on Cox analysis.

References

Friedman, J., Hastie, T., Tibshirani, R. (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent; *Journal of Statistical Software* **33(1), 1-22**; glmnet.

Simon, N., Friedman, J., Hastie, T., Tibshirani, R. (2011). Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent; *Journal of Statistical Software* **39(5), 1-13**; glmnet.

Therneau, T. (2020). A Package for Survival Analysis in R; *R package version 3.1-12*; survival.

Therneau, T.M., Grambsch, P.M., (2000). Modeling Survival Data: Extending the Cox Model; *Springer, New York* **ISBN 0-387-98784-3**.

See Also

glmnet

Examples

```
data(toyCleanedAml)
data(toyEigengenes)
survival <- toyCleanedAml$survival
inputTime <- setNames(survival[,"Time"]/365, nm=survival[,"PatientID"])
inputTime <- inputTime[inputTime > 0]
inBoth <- intersect(names(inputTime), rownames(toyEigengenes))
inputTime <- inputTime[is.element(names(inputTime), inBoth)]
inputEvent <- setNames(survival[is.element(survival$PatientID, inBoth), "Dead"],
                       nm=names(inputTime))

coxOut <- coxAnalysis(dataCategory="allSamples", inputTime=inputTime,
                      inputEvent=inputEvent, inputMatrix=toyEigengenes[inBoth,],
                      outPath=tempdir(), verbose=1)
```

---

createLocusGene *Mapping one locus to one gene.*

Description

Creates a dataframe that maps each loci to its corresponding gene and vice versa.

Usage

```
createLocusGene(locus2gene="Auto", genesColName="Gene_Symbol", lociColName="probeID",
                verbose=0)
```

Arguments

locus2gene	A dataframe with minimum 4 columns including genesColName, lociColName, "Chromosome", and "Genomic_Coodinate". Default "Auto" gets loci annotations from "IlluminaHumanMethylation450kanno.ilmn12.hg19".

genesColName — A string indicating the column name in the locus2gene dataframe that contain genes information.

lociColName — A string indicating the column name in the locus2gene dataframe that contain loci information.

verbose — The integer level of verbosity. 0 means silent, and higher values produce more details of the computation.

Value

A list with following components:

noGenes — A character vector of loci names that lack gene assignment.

locus2oneGene — A dataframe with one locus mapped to one gene.

Examples

```
data(toyCleanedAml)

locus2oneGene <- createLocusGene(locus2gene=toyCleanedAml$locus2gene,
                                 verbose=1)
```

---

distanceToTss *Computes distance of loci to the closest TSS*

Description

Computes distance of the input loci to the closest Transcription Start Site (TSS). Also, creates a plot to identify the fraction of loci that have distance less than 1000 bp to TSS.

Usage

```
distanceToTss(usefuLoci, locus2oneGene, genesColName="Gene_Symbol",
                        coordinatesColName="Genomic_Coordinate",
                        lociColName="probeID", doWarn=FALSE, verbose=0)
```

Arguments

usefuLoci — A list of character vectors. The name of each element of the list is a gene and the value is a vector of the corresponding loci, e.g, c("cg05116342", "cg15516558").

locus2oneGene — A data frame with at least four columns, namely: "probeID", "Gene_Symbol", "Chromosome", "Genomic_Coordinate". The rows are loci.

genesColName — A name of the column in the locus2oneGene dataframe that contains gene names. Default is set to "Gene_Symbol".

coordinatesColName — A name of the column in the locus2oneGene dataframe that contains genomic co-ordinate information. Default is set to "Genomic_Coordinate".

lociColName	A name of the column in the locus2oneGene dataframe that contains loci names. Default set to "probeID".
doWarn	Logical determining warnings should be issued if some gene symbols were not found.
verbose	A numeric value that can set verbosity level. Default = 1 indicates minimal verbosity.

Details

In the iNETgrate package this function is called in the computEigenloci function.

Value

A list of:

distanceToClosesTssPerGene
: An integer vector named by the loci. If a locus is related to multiple genes, there would be multiple entries of the same loci, e.g., cg27665767.4, cg27665767.8, cg27665767.9, etc.

distanceToClosesTss
: An integer vector named by the loci.

transcriptLen
: A numeric vector of length of gene transcripts

notFound
: A character vector of genes for which transcript length could not be calculated.

See Also

computEigenloci

Examples

```
data(toyCleanedAml)
genes1 <- c("FOXK1", "NTM", "MLPH", "SNTB1", "KCNA7", "XYLT1", "NF1")

## Creating locus2oneGene dataframe by calling createLocusGene function
createdLocusGene <- createLocusGene(locus2gene=toyCleanedAml$locus2gene,
                                    genesColName="Gene_Symbol",
                                    lociColName="probeID", verbose=0)
locus2oneGene <- createdLocusGene$locus2oneGene
subLocusGene <- locus2oneGene[is.element(locus2oneGene[ ,"Gene_Symbol"], genes1), ]

## A list of usefuLoci
usefuLoci <- split(subLocusGene$probeID, subLocusGene$Gene_Symbol)

computeDistance <- distanceToTss(usefuLoci=usefuLoci,
                                          locus2oneGene=locus2oneGene,
                                          genesColName="Gene_Symbol",
                                          coordinatesColName="Genomic_Coordinate",
                                          lociColName="probeID", verbose=1)
```

`downloaData` *Download and save TCGA data*

Description

Uses TCGAbiolinks to query, download, and save TCGA open-access data as an R object. THIS FUNCTION IS NOT CURRENTLY EXPORTED because it was based on the TCGA legacy data, which are not available any more.

Usage

```
downloaData(dataProject, savePath, doDnam=TRUE,  doExpr=TRUE,
              doClinical=TRUE, doMirna=FALSE, offset0=NA, verbose=0)
```

Arguments

`dataProject`	A character vector indicating the name for project as listed in TCGA portal.
`savePath`	A character vector that provides path to the folder where data is saved.
`doDnam`	Accepts a Boolean value, where `TRUE` (default) indicates DNA methylation data will be downloaded.
`doClinical`	Accepts Boolean value, where `TRUE` (default) indicates clinical data will be downloaded.
`doExpr`	Accepts Boolean value, where `TRUE` (default) indicates gene expression data will be downloaded.
`doMirna`	Accepts a Boolean value, where `TRUE` indicates micro RNA data will be downloaded. Default is set to `FALSE`.
`offset0`	A small value added to the expression levels before a logarithmic transformation in base 10 is applied. If `NA`, the smallest non-zero value in the data is used. Set to `NULL` to disable logarithmic transformation.
`verbose`	The integer level of verbosity. `0` means silent, and higher values produce more details of the computation.

Value

A list with data saved as R object, paths to saved data, and parameters used. Some of the list elements are:

`clinical`	A data frame of clinical data where rows are patient IDs; columns are clinical parameters.
`genExpr`	A data frame of gene expression data where rows are genes and columns are sample IDs. A logarithmic transformation in base 10 may be applied depending on the value of `offset0`.
`rawDnam`	A SummarizedExperiment object with DNA methylation data.
`genExprSampleInfo`	A data frame where rows are samples and columns are associated sample information.

References

Colaprico, A., Silva T.C., Olsen, C., Garofano, L., Cava, C., Garolini, D., Sabedot, T., Malta, T., Pagnotta, S.M., Castiglioni, I., Ceccarelli, M., Houtan Noushmehr, G.B., (05 May 2016) TCGAbiolinks: An R/Bioconductor package for integrative analysis of TCGA data *Nucleic Acids Research* **44(8): e71**. (doi:10.1093/nar/gkv1507); TCGAbiolinks

Silva, T. C. *et al.* (2016). TCGA Workflow: Analyze cancer genomics and epigenomics data using Bioconductor packages, *F1000Research* **5**.

Mounir, M. *et al.* (2019). New functionalities in the TCGAbiolinks package for the study and integration of cancer data from GDC and GTEx, *PLoS computational biology* **15.3 : e1006701**.

See Also

GDCquery, GDCdownload, GDCprepare, cleanAllData

Examples

```
 ## New TCGAbiolinks is buggy (2.24.3 < Version  <= 2.27.2).
tcgaAml <- downloaData(dataProject="TCGA-LAML", savePath=tempdir(),
                       doDnam=FALSE, doClinical=TRUE,
                       doExpr=FALSE, doMirna = FALSE)
```

---

electGenes *Selects input genes for iNETgrate analysis.*

Description

Selects a set of genes that have moderate or high correlation with survival time or vital status based on their expression or methylation levels. This is a wrapper function that executes filterLowCor for filtering out low correlation genes followed by the computeUnion functions.

Usage

```
electGenes(genExpr, dnam, survival, savePath, event="Dead", locus2gene,
           genesColName="Gene_Symbol", lociColName="probeID",
           ratio=c(genExpr=1/3, dnam=1), minCor=c(genExpr=0.2, dnam=0.2),
           doAddTitle=TRUE, doAlLoci=FALSE, verbose=0)
```

Arguments

genExpr	A matrix of gene expression data where row names are gene IDs and column names are patient IDs or case IDs.
dnam	A matrix of beta values where row names are probe or loci IDs and column names are patient IDs or case IDs.
survival	A matrix or data frame where rownames are patient IDs and must have minimum of two columns namely: "Time" (to event) and event.

savePath — A character vector determining the path to the folder or directory where the plots and output are saved.

event — A string determining the name of the event column in the survival data. It stores binary values 1 and 0 that correspond to the occurance or non-occurance of the event respectively.

locus2gene — A data frame or matrix with at least two columns that have information about probe IDs and the associated gene names. The rows are loci.

genesColName — A string determining the name of the column in `locus2gene` matrix that carries gene ID information.

lociColName — A string determining the name of the column in `locus2gene` matrix that carries probe ID information.

ratio — A named vector with two numeric values that determine the ratio of highly correlated genes and loci to be included in further analysis. The default is set to `1/3` for gene expression data and `1` for DNA methylation data.

minCor — A named vector with two numeric values that determine the minimum absolute correlation cut–off value to filter data. Default set to `0.2` for both gene expression and DNA methylation.

doAddTitle — A Boolean value, where `TRUE` (default) indicates appropriate titles will be added to the plots.

doAlLoci — A Boolean value, where `FALSE` (default) indicates that union gene set is composed of genes corresponding to selected (highly correlated) loci.

verbose — An integer that sets the level of verbosity. `0` means silent, and higher values produce more details of the computation.

Value

A list comprising of following objects:

selectedGenesExpr — A character vector of gene IDs that show high correlation with vital status or survival time based on gene expression data.

selectedLoci — A character vector of loci or probe IDs that show high correlation with vital status, survival time or both based on DNA methylation data.

selectedGenesDnam — A character vector of gene IDs that have high correlation with survival time or vital status based on DNA methylation data.

unionGenes — A character vector of gene IDs that have correlation with survival time or vital status based on expression levels or DNA methylation data.

unionLoci — A character vector of loci or probe IDs that correspond to the `unionGenes` set and also have appropriate methylation data.

See Also

`filterLowCor`, `computeUnion`

Examples

```
data(toyCleanedAml)

electedGenes  <- electGenes(genExpr=toyCleanedAml$genExpr, dnam=toyCleanedAml$dnam,
                            survival=toyCleanedAml$survival, savePath=tempdir(),
                            locus2gene=toyCleanedAml$locus2gene, verbose=1)
```

`filterLowCor` *Filters out genes or loci*

Description

Filters genes or loci that have low correlation with survival time or vital status and selects highly correlated genes or loci for further analysis.

Usage

```
filterLowCor(Data, survival, whichData=c("expression", "dnam"), ratio=1/3,
               minCor=0.2, event="Dead", savePath, verbose=0)
```

Arguments

`Data`	A matrix of gene expression or beta values where row names are gene or probe IDs and column names are patient or case IDs.
`survival`	A matrix or data frame where rownames are patient IDs and must have minimum of two columns namely "Time" (to event) and `event`.
`whichData`	A character value to describe input Data being filtered. E.g. expression for gene expression data.
`minCor`	Minimum absolute correlation cut–off value to filter data. Default set to `0.2`.
`ratio`	A ratio that determines highly correlated genes or loci to be included in further analysis. Default set to `1/3`.
`event`	The name of the event column in the survival data. It stores binary values `1` and `0` that correspond to the occurance or non-occurance of the event, respectively.
`savePath`	Path to the directory where plots and output are to be saved.
`verbose`	The integer level of verbosity. `0` means silent, and higher values produce more details of the computation.

Value

A list comprising of following objects:

`alList`	A character vector of complete list of gene or probe IDs from the input.
`selectedList`	A character vector of gene or probe IDs that show high correlation with vital status or survival time.
`corValues`	A numeric vector of correlation values where names are selected genes or loci.

See Also

prepareSurvival

Examples

```
data(toyCleanedAml)

## Gene expression
filteredGeneExpr <- filterLowCor(Data=toyCleanedAml$genExpr,
                                 survival=toyCleanedAml$survival,
                                 savePath=tempdir(), whichData="expression")

## DNA methylation
filteredDnam <- filterLowCor(Data=toyCleanedAml$dnam,
                             survival=toyCleanedAml$survival,
                             savePath=tempdir(), whichData="dnam", ratio=1)
```

---

findAliveCutoff *Find Alive Cutoff*

Description

Finds the best (smallest) cutoff on hope where the ratio of alive cases, for which time1 > until, is at least 'ratio'.

Usage

```
findAliveCutoff(hope, time1, until, minRecall=0.2, risk="Low",
                Labels=NULL, doDebug=FALSE, resPath, verbose=0,
                doPlot=FALSE, lowLabel="Low", highLabel="High")
```

Arguments

hope	A named vector where names are patient IDs and the values are the predicted survival time.
time1	A named vector where names are patient IDs and the values are survival time in years.The vector is of the same length as hope.
until	A numeric value e.g. 10 years for breast cancer. It will be ignored if Labels is not NULL.
minRecall	Minimum recall value to classify patients based on their risk level.
risk	A string indicating the risk level for which cut–off is to be found. Set to "Low" or "High".
Labels	A named vector where names are patient IDs same as time1 and hope and values are training risk groups: "High","Int", "Low". Alternatively, set until to "left" to find the high–risk cut–off.
doDebug	A Boolean with a default value of FALSE.

resPath	A string describing the path to the folder to save results (plots).
verbose	The integer level of verbosity. 0 means silent, and higher values produce more details of the computation.
doPlot	A Boolean with a default value of FALSE.
lowLabel	A string describing the low risk–cases, e.g., "Low".
highLabel	A string describing the high–risk cases, e.g., "High".

Value

A list which contains:

num	A numeric value indicating the number of samples analysed.
cutoff	A number indicating the cutoff value for being high–risk or low–risk based on survival time.
highPrecision	Precision for high–risk cut–off.
lowPrecision	Precision for low–risk cut–off.
highRecall	Recall for high–risk cut–off.
lowRecall	Recall for low–risk cut–off.

Examples

```
data(toyCleanedAml)
survival <- toyCleanedAml$survival
## Labels <- setNames(survival[, "Risk1"], nm=survival[, "PatientID"])
time1 <- setNames(survival[, "Time"]/365, nm=survival[, "PatientID"])
dummyTime <- time1
dummyTime[1:100] <- dummyTime[1:100]/2

aliveCutoff <- findAliveCutoff(hope=dummyTime, time1=time1, until=1,
                                resPath=tempdir(), doPlot=TRUE, verbose=2)
```

---

findCore *Finds the most connected cluster of points*

Description

Uses hierarchical clustering and a greedy approach to find "the core" (the most connected cluster of loci) for a gene in the given DNA methylation data.

Usage

```
findCore(Data, Labels, Label1, Label2, verbose=0)
```

Arguments

Data	A matrix of beta values where column names are probe or loci IDs corresponding to the gene of interest and rownames are patient or case IDs.
Labels	A named character vector where names are patient or case IDs and values are the lave of the actual risk of patients. See Label1 and Label2.
Label1	One of the values from Labels vector that determines a patient group to be used to balance the data. E.g. low–risk group defined as "Low" in toyCleanedAml.
Label2	One of the values from Labels vector that determines a patient group to be used to balance the data. E.g. high–risk group defined as "High" in toyCleanedAml.
verbose	The integer level of verbosity. 0 means silent, and higher values produce more details of the computation.

Value

A list with following components:

PC1	The matrix of inferred (projected) eigengenes where rows are samples and columns are modules.
pigenObject	An object of Pigenengene class.
lociNames	A vector of loci that we used for computing methylation level of the related gene (using PCA, a weighted average value).
aveWeight	Average similarity level of the loci group that relates to a gene
communities	Clustered loci
num	Number of unique loci clusters

References

Amir Foroushani *et al.* (2016) Large-scale gene network analysis reveals the significance of extracellular matrix pathway and homeobox genes in acute myeloid leukemia: an introduction to the Pigengene package and its applications, *BMC MedicalGenomics*.

Examples

```
data(toyCleanedAml)

unionGenes <- c("BAI1", "CAPNS1", "CHD1", "CYC1", "EXT1", "FYB", "GFRA1", "HLX",
                "IDH3B", "PSMD8", "SGTA", "SLC1A5", "SSFA2", "TFAM", "TGIF1",
                "UBC", "ZNF70", "ST8SIA4", "ZNF485", "FOXN3", "HCG4", "FOXK1",
                "C2orf44")

l2g <- toyCleanedAml$locus2gene
survival <- toyCleanedAml$survival

loci <- l2g[is.element(l2g[, "Gene_Symbol"], unionGenes), "probeID"]
patientLabel <- setNames(as.character(survival$Risk1), rownames(survival))
inBoth <- intersect(colnames(toyCleanedAml$dnam), names(patientLabel))
LabelsIn <- patientLabel[is.element(names(patientLabel), inBoth)]
```

```
dnamData <- t(toyCleanedAml$dnam[loci, inBoth])

core <- findCore(Data=dnamData, Labels=LabelsIn, Label1="Low", Label2="High",
                 verbose=1)
```

`findTcgaDuplicates` *Identify and remove duplicate samples*

Description

Identifies sample IDs in gene expression and DNA methylation data that belong to same patients or obtained from same cell types or have been obtained or processed using same experimental techniques. The samples are removed using lexicographic sorting of the sample IDs as generally used in sorting TCGA samples.

Usage

```
findTcgaDuplicates(sampleInfo, sampleCol="barcode", patientCol="patient",
                   sampleTypeCol="shortLetterCode", doTag=FALSE, verbose=0)
```

Arguments

`sampleInfo`	A data frame or a matrix that gives information about the samples with at least two columns that contain patient IDs and corresponding sample IDs. The rows are samples.
`sampleCol`	A string that determines the name of the column in `sampleInfo` data frame that contains sample IDs.
`patientCol`	A string that determines the name of the column in `sampleInfo` data frame that contains patient IDs.
`sampleTypeCol`	A string that determines the name of the column in `sampleInfo` data frame that contains sample type information e.g., "shortLetterCode" in toyRawAml data. Set it to `NULL` to disable filtering based on sample types.
`doTag`	A Boolean value that indicates if patient IDs in patientCol will be tagged with sample type from sampleTypeCol e.g., `TRUE` tags patient IDs with sample types that prevents removing duplicate samples from same patients with different sampling source and vice versa.
`verbose`	An integer that sets the level of verbosity. `0` means silent, and higher values produce more details of the computation.

Value

A list with following components:

`sampleInfo`	A data frame or a matrix with the same number of columns as the input `sampleInfo`, but fewer rows. Rows of duplicate samples are removed.
`keptDups`	A character vector of duplicate sample IDs that are kept for further analysis.
`excludeDups`	A character vector of duplicate sample IDs that are excluded from further analysis.

See Also

sample2pat

Examples

```
data(toyRawAml)

removeDups <- findTcgaDuplicates(sampleInfo=toyRawAml$genExprSampleInfo,
                                 sampleCol="barcode", patientCol="patient",
                                 sampleTypeCol="shortLetterCode",
                                 doTag=TRUE, verbose=1)
```

---

iNETgrate *iNETgrate - Runs entire iNETgrate pipeline.*

Description

Runs the entire iNETgrate pipeline; from cleaning the data to finding the best modules and performing analyzeSurvival. Specifically, it computes the eigenloci (weighted average of beta values per gene), identifies the gene modules using coexpression network analysis, computes eigengenes, identifies best modules, reclassifies patients into risk groups based on the eigengene values, and performs analyzeSurvival.

Usage

```
iNETgrate(Data, clinSettings, mus=(0:10)/10, saveDir="iNETgrate",
          annLib="Auto", isFfpe=c(FALSE), doRemoveTOM=TRUE,
          minModuleSize=5, corMethod="pearson", doReturNetworks=FALSE,
          RsquaredCut=0.75, combiningMu=NA, favRisk="High", subSet="Int",
          xmax1=15, xmin1=0, doCox=TRUE, eOrMs="e", time2day=1,
          until=1, minRecall4L=0.2, minRecall4H=0.05, verbose=0)
```

Arguments

Data	A list with minimum of four elements named rawDnam, clinical, genExpr, and genExprSampleInfo. Here, rawDnam is an object of class SummarizedExperiment, genExpr is the matrix of gene expression values with genes on rows and patients on columns, and clinical and genExprSampleInfo are data frames with patients on rows. Example: toyRawAml.
clinSettings	A vector where names are "patientIDCol", "eventCol", "timeCol", "riskCatCol", "riskFactorCol", "event", "riskHigh", and "riskLow", and the values are the corresponding information from the clinical data. In particular, values for a name ending in "Col" must be the corresponding column name in the clinical dataframe e.g., "patientIDCol" in the name of the column that has the patient ID. "event" is the event of interest (e.g., "Dead" may mean dead due to the disease) as indicated in "eventCol". "riskHigh", and "riskLow" are the values of "riskCatCol" corresponding to high– and low–risk, respectively. See the cleanAllData function and the example for further details.

mus	A numeric vector assigning mu value(s) in the range [0,1]. A higher value for mu will result in more emphasize on the DNA methylation data than gene expression data. In particular, mu=0 would lead to ignoring DNA methylation data, with mu=1 gene expression data would have no impact on the network, and with mu=0.5 DNA methylation and gene expression would have equal contribution to the network.
saveDir	A character string describing the path to directory for saving plots and output.
annLib	A character string that determines the annotation (reference) library to be used. The default is "IlluminaHumanMethylation450kanno.ilmn12.hg19".
isFfpe	A character vector that takes Boolean values to indicate inclusion or exclusion of FFPE sample. c(FALSE, TRUE) will include all samples, TRUE will include only FFPE samples, and FALSE will exclude FFPE samples. Set it to NULL to disable it.
doRemoveTOM	A Boolean determining if TOM files must removed. Default is set to TRUE.
minModuleSize	A numeric value that specifies the minimum number of genes per module.
corMethod	A string that specifies the correlation method to be used. Accepted options include "spearman" or "pearson", and default is set to "pearson".
doReturNetworks	A Boolean value determining whether to return exprNetwork and dnamNetwork, which are relatively big objects (typically Gbs). Default: FALSE.
RsquaredCut	A threshold in the range [0,1] used to estimate the power (beta value) for soft thresholding by WGCNA. A higher value can increase the power to which the similarity matrix is raised. For technical use only. See pickSoftThreshold for more details.
combiningMu	A numeric vector determining the mu value(s) in the range [0,1] used for eigengene computation. If set to NA, the same value that was used for network construction will be used for eigengene computation. See mus.
favRisk	The value must be one of these three: "High", "Int" or "Low". Determines the risk category predicted by network classifier that shall be compared to the input Labels risk categories.
subSet	The value must be one of these three: "High", "Int" or "Low". Determines the risk category in Labels that shall be compared to the risk category predicted by network classifier.
xmax1	A numeric value determining the maximum time (in years) for plotting the K–M curves.
xmin1	A numeric value determining the minimum time (in years) for plotting the K–M curves.
doCox	Accepts a Boolean value, where TRUE (default) indicates Cox analysis will be performed for module selection.
eOrMs	A character vector to determine which eigengenes (stored at netPath) to be used as input data for analyzeSurvival, e.g., c("m", "e", "em"), where "e" means "only gene expression", "m" means "DNA methylation", and "em" means the combination of both data types is used to compute an eigengene for a module.

time2day	A numeric value used to calculate survival time (in days). E.g. if the "Time" column in survival data frame is in days, time2day=1, set it to 30 if "Time" is in months, etc.
until	A numeric value describing minimum survival time (in years) for a case to be considered as low risk, e.g., if an AML patient was alive for at least 2 years after diagnosis AND was alive at the last followup time, then we can consider this case low–risk.
minRecall4L	A numeric value of the aimed recall cut–off for low–risk predictions, which may not be reached exactly. The higher, the more low–risk cases would be identified, but the risk in the group may be relatively higher. Do not change the default if you are not an expert.
minRecall4H	A numeric value of the aimed recall cut–off for high–risk predictions, which may not be reached exactly. The higher, the more high-risk cases would be identified, but the risk in the group may be relatively lower. Do not change the default if you are not an expert.
verbose	An integer level of verbosity: 0 means silent, and higher values produce more details of the computation.

Details

This is a wrapper function to run the iNETgrate pipeline in several steps. First, the input data is cleaned by dropping any missing data and imputing data as needed. Next, we exclude the genes for which there is no or negligible correlation between expression levels nor DNA methylation and survival data. We compute a weighted average of beta values per gene, and build a network using both gene expression as well as DNA methylation data. Gene modules are identified based on co-expression and DNA methylation correlation between genes. Then, the eigengenes for each module and each sample are calculated, where the expression of an eigengene of a module in a sample is the weighted average of the expression of the genes in that module in that sample. Technically, an eigengene is the first principal component of the gene expression levels in a module. Using PCA, we ensures that the maximum variance across all the training samples is explained by the eigengene. Next, we select a combination of 3 modules or less, based on a Cox regression analysis of survival data and the corresponding module eigengenes. Along the way, several self explanatory directories, and plots are created and stored under saveDir.

Value

A list with following components:

call	The call that created the results.
unionGenes	A vector of gene names that are selected to make the network.
Labels	A vector where names are patient IDs and values are the corresponding risk levels.
eigenloci	A matrix of DNA methylation level per genes (columns) with each row corresponding to a patient.
bestInet	A list of outputs from bestInetgrator function.

Note

The individual functions are exported to facilitate running the pipeline step–by–step and in a customized way.

References

R Core Team (2019). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. stats.

Foroushani, Amir, et al. (2017) Large-scale gene network analysis reveals the significance of extracellular matrix pathway and homeobox genes in acute myeloid leukemia: an introduction to the Pigengene package and its applications, *BMC medical genomics*, **10.1**, doi.org/10.1186/s12920-017-0253-6.

Langfelder, P. and Horvath, S., (2008) WGCNA: an R package for weighted correlation network analysis, *BMC Bioinformatics*, **9:559**, doi:10.1186/1471-2105-9-559.

Langfelder P., Horvath S. (2012) Fast R Functions for Robust Correlations and Hierarchical Clustering. *Journal of Statistical Software*, **46(11),1-17**. WGCNA.

See Also

pickSoftThreshold, cleanAllData, electGenes, makeNetwork, computEigengenes, bestInetgrator, analyzeSurvival, toyRawAml

Examples

```
## Preparing data:
data(toyRawAml)

## Set up clinical settings
clinSettings <- c("patientIDCol"="bcr_patient_barcode",
"eventCol"="vital_status",
"timeCol"="days_to_last_followup",
"riskCatCol"="acute_myeloid_leukemia_calgb_cytogenetics_risk_category",
"riskFactorCol"="cytogenetic_abnormalities",
"event"="Dead",
"riskHigh"="Poor",
"riskLow"="Favorable")
print(clinSettings)

## Printing few rows and columns of clinical data
head(toyRawAml$clinical[ , clinSettings[c("patientIDCol","eventCol", "timeCol",
                                          "riskCatCol", "riskFactorCol")]])

inetgrateRes <- iNETgrate(Data=toyRawAml, clinSettings=clinSettings, mus=0.6)
```

`inferEigengenes` *Infers features for the test(validation) dataset*

Description

It computes features (eigengenes) for the test (validation) dataset.

Usage

```
inferEigengenes(inetgrator, dnam=NULL, expr, verbose=0)
```

Arguments

`inetgrator`	A list with at least 3 components namely "mu", "pigengenes", and "modules" similar to the output from `bestInetgrator`.
`dnam`	A matrix of beta values where rows are samples and columns are loci.
`expr`	A matrix of expression values where rows are samples and columns are genes.
`verbose`	An integer level of verbosity. `0` means silent, and higher values produce more details of the computation.

Value

A list with following components:

`inetgrator`	A list same as the input `inetgrator`.
`eigengeneS`	A matrix with samples on rows, the inferred features (eigengenes) on columns.
`missingInExpr`	A character vector of genes missing in the test dataset.

See Also

`bestInetgrator`, `computEigenloci`

Examples

```
## See the analyzeSurvival() for an example.
?analyzeSurvival()
```

makeNetwork *Make network*

Description

Uses the adjacency to create a network from input data. Then, uses the mu value to create a combined network from the expression network and DNAm network.

Usage

```
makeNetwork(genExpr, eigenloci, geNames, mus, doRemoveTOM=TRUE, outPath,
            minModuleSize=5, corMethod="pearson", doReturNetworks=FALSE,
            RsquaredCut=0.75, doSaveCombined=FALSE, verbose=0)
```

Arguments

genExpr	A gene expression matrix where row names are gene IDs and column names are patient IDs.
eigenloci	An eigenloci matrix where row names are patient IDs and column names are gene IDs.
geNames	A character vector of selected genes that become the nodes of the network. Both gene expression and eigenloci data must be available for these genes (e.g., unionGenes in the output of electGenes).
mus	A numeric vector assigning mu value(s).
doRemoveTOM	A Boolean determining if TOM files must removed. Default is set to TRUE. For technical users only.
outPath	A string that determines the path to the folder where plots and output will be saved.
minModuleSize	A numeric value that specifies the minimum number of genes per module. See blockwiseModules.
corMethod	A string that specifies the correlation method to be used. Accepted options include "spearman" or "pearson", and default set to "pearson".
doReturNetworks	A Boolean value determining whether to return exprNetwork and dnamNetwork, which are relatively big objects (typically Gbs). Default FALSE.
RsquaredCut	A threshold in the range [0,1] used to estimate the power. A higher value can increase power for technical use only. See pickSoftThreshold for more details.
doSaveCombined	If TRUE, the results from combine.networks function will be saved on storage. Leave it as FALSE not to waste typically 1-2 GBs disk space.
verbose	An integer level of verbosity. 0 means silent, and higher values produce more details of the computation.

Value

A list with following components:

corMethod	A character value same as the input.
mus	A numeric vector of mu value(s) same as the input.
mu2modules	A matrix where rows are mus and values are module information that is an output of combine.networks.
powerVector	A numeric vector of power values named with the corresponding mus. See pickSoftThreshold documentation.
outliersNumber	A named numeric vector where names are mu value(s) and values are number of outlier genes (genes in module 0).

If doReturNetworks is TRUE the following components are returned:

exprNetwork	An adjacency matrix of gene expression.
dnamNetwork	An adjacency matrix of DNA methylation.

References

Langfelder, P. and Horvath, S., (2008) WGCNA: an R package for weighted correlation network analysis, *BMC Bioinformatics*, **9:559**, doi:10.1186/1471-2105-9-559

Langfelder P., Horvath S. (2012) Fast R Functions for Robust Correlations and Hierarchical Clustering. *Journal of Statistical Software*, **46(11),1-17**. WGCNA

Foroushani, A. *et al.* (2016) Large-scale gene network analysis reveals the significance of extracellular matrix pathway and homeobox genes in acute myeloid leukemia: an introduction to the Pigengene package and its applications, *BMC MedicalGenomics*.

See Also

combine.networks, pickSoftThreshold, blockwiseModules, electGenes

Examples

```
data(toyCleanedAml)
data(toyComputEloci)
eigenloci <- toyComputEloci$eigenloci

netPath <- file.path(tempdir(), "net")
dir.create(netPath, showWarnings=FALSE)
madeNetwork <- makeNetwork(genExpr=toyCleanedAml$genExpr, eigenloci=eigenloci,
                           geNames=colnames(eigenloci), mus=c(0.6),
                           doRemoveTOM=FALSE, outPath=netPath, verbose=1)
```

plotKM *Survival (Kaplan–Meier estimator) plots*

Description

Plots KM (survival) plots, one curve per each risk group cases.

Usage

```
plotKM(inputTime, inputEvent, cond,
       cond2Color=c("Low"='green', "Int"='blue', "High"='red'),
       titleText=NULL, weight=rep(1, times=length(inputTime)),
       doIntLow4pval=FALSE, plotFile=NULL, legendCex=1.7,
       ylab1="Survival Probablity", pvalDigits=0, xmax1=Inf, xmin1=0,
       verbose=0, ...)
```

Arguments

inputTime	A named numeric vector where names are patient IDs and values are survival time (in years).
inputEvent	A named binary vector where names are patient IDs and values are 0 and 1. Here, 1 indicates the occurance of the event and vice versa.
cond	A named character vector, where names are patient IDs and values are their risk category the patientID belongs to.
cond2Color	A named character vector, that assigns color to risk groups in KM–plots. Names must be in the cond vector, and values are colors. e.g, cond2Color=c("Low"='green', "Int"='blue', "High"='red')
titleText	A character string that assigns title of the plot.
weight	A numeric vector, same as length of inputTime vector.
doIntLow4pval	Accepts a Boolean value, where FALSE (default) indicates intermediate cases are excluded when computing the p–value. If TRUE, intermediate cases are considered as low–risk cases for p–value computation.
plotFile	A character string that determines path to save the plots.
legendCex	A numeric value that determines graphical location of legend in the plots.
ylab1	A character string that determines the Y–axis label in the plot.
pvalDigits	A numeric value that gives number of decimal digits to be printed on the K–M plots for p–value.
xmax1	A numeric value giving the maximum time (in years) for plotting the K–M curves.
xmin1	A numeric value giving the minimum time (in years) for plotting the K–M curves.
verbose	The integer level of verbosity. 0 means silent, and higher values produce more details of the computation.
...	User can pass any more arguements for creating better plots.

Value

A list with following components:

pVal	A numeric value that gives calculated p-value from KM survival plot.
yTrain	A Surv object, an output Surv.
survFitOutput	A survfit object, an output of survfit.
doSkiPlot	A Boolean value, where TRUE indicates KM plot is skipped (not plotted) and vice versa.

References

Therneau, T. (2020). A Package for Survival Analysis in R. R package version 3.1-12, survival.

Therneau, T.M., Grambsch, P.M., (2000). Modeling Survival Data: Extending the Cox Model. *Springer, New York. ISBN 0-387-98784-3*.

See Also

Surv, survfit

Examples

```
data(toyCleanedAml)

survival <- toyCleanedAml$survival
inputTime <- setNames(survival[,"Time"]/365, nm=survival[, "PatientID"])
inputEvent <- setNames(survival[,"Dead"], nm=survival[, "PatientID"])
cond <- setNames(survival[,"Risk1"], nm=survival[, "PatientID"])

plottedKM <- plotKM(inputTime=inputTime, inputEvent=inputEvent, cond=cond)
```

---

plotLociNum *Plot number of loci per gene.*

Description

Creates plots of the number of loci per gene.

Usage

```
plotLociNum(locus2gene, genesColName="Gene_Symbol", lociColName="probeID",
            selectedLoci="Auto", plotFile, doAddTitle=TRUE,
            xlab1="Number of loci per gene", ylab1="Cumulative probability",
            verbose=0)
```

Arguments

locus2gene	A matrix or a dataframe with at least two columns that contain information about probe ID (e.g., cg00000029), and corresponding gene symbol (e.g., TSPY4). The rows are loci.
genesColName	A string that determines the gene IDs column in locus2gene.
lociColName	A string that determines the probe IDs column in locus2gene.
selectedLoci	A character vector of probe IDs that are selected to be plotted.
plotFile	A string that determines the file path to save plots.
doAddTitle	Accepts a Boolean, where TRUE (default) indicates appropriate titles will be added to the plots.
xlab1	A string describing the X–axis label for the plots.
ylab1	A string describing the Y–axis label for the plots.
verbose	An integer that sets the level of verbosity. 0 means silent, and higher values produce more details of the computation.

Value

It is a list with following components:

noGenes	A character vector of probe IDs that have no associated gene assignement.
l2g	A sorted table of number of loci associated with each gene.

Examples

```
data(toyCleanedAml)

plotFile <- file.path(tempdir(), "plotLociPerGene.png")

plottedLoci <- plotLociNum(locus2gene=toyCleanedAml$locus2gene, genesColName="Gene_Symbol",
                           lociColName="probeID", selectedLoci="Auto", plotFile=plotFile,
                             doAddTitle=TRUE, xlab1="Number of loci per gene",
                             ylab1="Cumulative probability", verbose=1)
```

---

plotLociTss *Plots distance of loci to genes*

Description

Generates plots describing distance of a loci from a Transcription Start Site (TSS).

Usage

```
plotLociTss(distanceToTss, cutoff=1000, plotFile, verbose=0)
```

Arguments

distanceToTss	A numeric vector, where each element is the distance between a locus (probe) and the closest Transcription Start Site (TSS) of the nearby genes. I.e., from the full output of the distanceToTss function, only distanceToClosesTss was selected.
cutoff	A numeric value that determines the maximum distance (in basepairs) between probe and TSS.
plotFile	A character string that gives path to the folder where plot will be saved.
verbose	The integer level of verbosity. 0 means silent, and higher values produce more details of the computation.

Value

The distance matrix. Also, a plot is saved at plotFile.

Examples

```
distanceToTss <- c(2460, 1498, 17182, 7807, 237, 56815, 368948)
names(distanceToTss) <- c("cg25539759", "cg25408497", "cg20716080", "cg07917842",
                          "cg07312445", "cg04774620", "cg06129210")
savePath <- file.path(tempdir(), "plotDistanceToTss.png")

plotted <- plotLociTss(distanceToTss=distanceToTss, cutoff=1000,
                       plotFile=savePath, verbose=0)
```

---

prepareSurvival *Prepare survival data*

Description

Processes and cleans clinical data to gather the required survival information.

Usage

```
prepareSurvival(clinical, patientIDCol="bcr_patient_barcode", eventCol="vital_status",
                event="Dead", timeCol="days_to_last_followup",
            riskCatCol=NULL, riskFactorCol=NULL, riskHigh="High", riskLow="Low",
                verbose=0)
```

Arguments

clinical	A matrix or a data frame of clinical data with a minimum four columns that provide information about patient IDs, event occurrence, survival or follow up time, and risk factors. The rows are patients.
patientIDCol	A string determining the column in clinical data that lists patient IDs or case IDs, e.g., "bcr_patient_barcode" in the sample AML data.

eventCol	A string determining the column in clinical data that lists the event. The Event column must be a binary column E.g. "vital_status" column in sample AML data where two possible events are "Dead" and "Alive".
event	A string determining the event of interest from eventCol in clinical data e.g., "Dead" is the event of interest in the sample data.
timeCol	A string determining the column in clinical data storing survival time information e.g., "days_to_last_followup" in sample AML data
riskCatCol	The name of the column in clinical data that includes risk levels of the disease for all patients. Example values in this column: "Low", "Hight", etc.
riskFactorCol	The name of the column in clinical data that describes the risk factors determining risk levels, which are given in the riskCatCol column. Example values in this column: "benign", "StageII", "metastasized", etc.
riskHigh	A character vector that describes high-risk group of patients in clinical data. See section "Details".
riskLow	A character vector that describes low-risk group of patients in clinical data. See section "Details".
verbose	An integer that sets the level of verbosity. 0 means silent, and higher values produce more details of the computation.

Details

This function removes any cases in clinical data for which vital status or survival time is missing. Risk levels can be determined in two ways: (a) riskCatCol, or alternatively, (b) riskFactorCol. When riskCatCol is defined, riskHigh and riskLow are expected to have values that belong to riskCatCol column of clinical. When riskCatCol is NULL and riskFactorCol is defined, riskHigh and riskLow are expected to have values that belong to the riskFactorCol column of clinical. riskCatCol and riskFactorCol cannot be NULL together. Patients that do not belong to nigher low- nor high-risk groups are considered intermediates-risk (i.e, "Int"). riskHigh and riskLow cannot be NULL.

Value

A data frame with a minimum of five columns namely: "PatientID", event, "Time", "Risk1", and riskFactorCol. The row names are the same as "PatientID".

See Also

downloaData, cleanAllData, analyzeSurvival

Examples

```
data(toyRawAml)

riskCatCol = "acute_myeloid_leukemia_calgb_cytogenetics_risk_category"
riskFactorCol = "cytogenetic_abnormalities"
survival <- prepareSurvival(clinical=toyRawAml$clinical,
                            patientIDCol="bcr_patient_barcode", eventCol="vital_status",
                             event="Dead", timeCol="days_to_last_followup",
```

```
                    riskCatCol=riskCatCol, riskFactorCol=riskFactorCol,
                    riskHigh="Favorable", riskLow="Poor")
```

`preprocessDnam` *Cleans DNA methylation data*

Description

Removes any loci with more than 50% missing beta values and imputes data for any locus that has less than 50 percent missing beta values (i.e., `NA`s are replaced with the mean of the corresponding locus). Also, it identifies and removes loci that are identified as non-CpG, have Single Neucleotide Polymorphism (SNPs), or have missing chromosome information.

Usage

```
preprocessDnam(rawDnam, rawDnamSampleInfo=NULL, savePath, annLib="Auto", verbose=0)
```

Arguments

`rawDnam`	A matrix of beta values or a summarized experiment object similar to the output of `GDCprepare`. See 'Details' section for more.
`rawDnamSampleInfo`	A data frame that gives information about the DNA methylation samples with at least two columns that contain patient IDs and corresponding sample IDs. See 'Details' section for more.
`savePath`	A character vector that determines the path to the folder where plots can be saved.
`annLib`	A string that gives name of annotation (reference) library to be used. Default is "IlluminaHumanMethylation450kanno.ilmn12.hg19".
`verbose`	An integer that sets the level of verbosity. `0` means silent, and higher values produce more details of the computation.

Details

If `rawDnam` is a matrix, the row names are probe IDs, column names are patient IDs, and values are dnam beta values. In case `rawDnam` is of class `SummarizedExperiment`, it is assumed that sample information is a part of `rawDnam` and hence, `rawDnamSampleInfo` can be set to `NULL`. Otherwise `rawDnamSampleInfo` input is needed to get an appropriate sampleInfo output.

Value

A list with following components:

`dnam`	A matrix of dnam beta values that is cleaned and imputed where rownames are probe IDs and column names are patient IDs.

locus2gene	A dataframe of loci information where rownames are probeIDs and at least 4 columns that contain information about probeIDs, geneIDs, genomic coordinates, and chromosome information.
sampleInfo	A dataframe of sample information where rownames are sample IDs and at least two columns that contains patient information and sample type. The rows are samples.

See Also

GDCprepare, getAnnotation

Examples

```
data(toyRawAml)
processeDnam <- preprocessDnam(rawDnam=toyRawAml$rawDnam,
                               rawDnamSampleInfo=NULL, savePath=tempdir(),
                               annLib="Auto", verbose=1)
```

---

sample2pat *Maps sample IDs to corresponding patient IDs*

Description

Maps sample IDs in gene expression and DNA methylation data to corresponding patient IDs in clinical data and renames columns of the matrix with patient IDs. The function also provides options to exclude or include FFPE samples, undesired sample types, or duplicate samples.

Usage

```
sample2pat(sampleInfo, sampleCol="barcode", patientCol="patient",
              sampleTypeCol="shortLetterCode", sampleTypesIn=NULL, isFfpe=NULL,
                 doRemoveDup=TRUE, verbose=0)
```

Arguments

sampleInfo	A data frame or matrix that where rows are the samples with at least two columns that determines patient IDs and corresponding sample IDs.
sampleCol	A string that determines the name of the column that contains sample IDs in sampleInfo data frame.
patientCol	A string that determines the name of the column that contains patient IDs in sampleInfo data frame.
sampleTypeCol	A string that determines the name of the column that contains sample type information in sampleInfo data frame e.g., "shortLetterCode" in the toyRawAml data. Set it to NULL to disable filtering based on sample types. See sampleTypesIn.
sampleTypesIn	A character vector that identifies the sample types to be included for iNETgrate analysis e.g., c("TP", "NT") are sample types in TCGA data where TP and NT indicate Primary Tumor and Non-Tumor adjacent samples respectively.

isFfpe — A character vector that takes Boolean values to indicate inclusion or exclusion of FFPE sample e.g., c(FALSE, TRUE) will include all samples, TRUE will include only FFPE samples, and FALSE will exclude FFPE samples. Set it to NULL to disable FFPE sample filtering.

doRemoveDup — A Boolean value that indicates inclusion or exclusion of duplicate samples. E.g., TRUE (default) removes duplicate samples using lexicographic sorting of the sample IDs as described in findTcgaDuplicates

.

verbose — An integer that sets the level of verbosity. 0 means silent, and higher values produce more details of the computation.

Details

Frozen samples are preferred over FFPE samples for sequencing, and hence, removal of FFPE data is preferred for any expression or methylation analysis.

Value

A list with following components:

sample2patient — A named vector where names are sample IDs and values are patient IDs.

patient2type — A named vector where names are tagged patient IDs and values are the sample type e.g., "TB" is the sample type in the toyRawAml data

patient2type — A list of excluded sample IDs

See Also

cleanAllData, findTcgaDuplicates

Examples

```
data(toyRawAml)

s2pExpr <- sample2pat(sampleInfo=toyRawAml$genExprSampleInfo,
                              sampleCol="barcode", patientCol="patient",
                              sampleTypeCol="shortLetterCode",
                              sampleTypesIn=NULL, isFfpe=NULL,
                              doRemoveDup=TRUE, verbose=1)
```

sampleData *Sample Data*

Description

Samples data for quick running of the pipeline.

Usage

```
sampleData(Data, cleanData, numbeRow=300)
```

Arguments

Data	A list of input data same as the output of downloaData function.
cleanData	A list of cleaned data same as the output of cleanAllData function.
numbeRow	An integer that indicates the number of rows to be sampled.

Value

A list with following elements:

rawData	A list of sampled gene expression matrix, gene expression sample information, rawDnam data, and clinical data.
cleaned	A list of sampled cleaned gene expression matrix, locus2gene dataframe, cleaned Dnam data, and survival data.

Note

Use set.seed before calling this function so that you will be able to reproduce the same sample in the future if needed.

See Also

downloaData, cleanAllData

Examples

```
data(toyRawAml)
data(toyCleanedAml)
sampleData(Data=toyRawAml, cleanData=toyCleanedAml, numbeRow=100)
```

`toyCleanedAml` *A subset of cleaned TCGA-LAML data*

Description

Survival data of 186 acute myeloid leukemia (AML) patients from The Cancer Genome Atlas (TCGA) managed by the NCI and NHGRI (Accession: phs000178.v11.p8) is included here. Additionally, DNA methylation profile of 600 loci for 194 AML cases and gene expression profile of 396 genes (associated with loci) for 173 AML cases are included here.

Usage

```
data("toyCleanedAml")
```

Format

A list

Details

The original gene expression and DNA methylation data was produced using Affymetrix U133 Plus 2 platform and Illumina Infinium HumanMethylation450 BeadChip platform, respectively. These TCGA–LAML data are aimed to serve as a foundation for further investigations of AML pathogenesis, classification, and risk stratification.

Value

A list of 4 components.

`genExpr`	A matrix of gene expression profile where rows and columns are genes Symbol, and sample IDs, respectively.
`locus2gene`	A dataframe where rows are probe IDs.
`dnam`	A matrix of DNA methylation profile where rows and columns are probe IDs, and sample IDs, respectively.
`survival`	A dataframe where rows are patient IDs, and columns provide survival information and vital status.

Source

https://portal.gdc.cancer.gov/projects/TCGA-LAML

References

Genomic and Epigenomic Landscapes of Adult De Novo Acute Myeloid Leukemia (2013). New England Journal of Medicine 368.22: 2059-2074.

See Also

iNETgrate-package, toyRawAml, toyComputEloci

Examples

```
## This dataset can be regenerated using the following commands,
##which can take some time to run.

dataProject <- "TCGA-LAML"
dataPath <- file.path(tempdir(), "data", dataProject)
dir.create(dataPath, recursive=TRUE)
print(paste("Data will be saved in:", dataPath))

## Downloading data
rawData <- downloaData(dataProject=dataProject, savePath=dataPath)

## Cleaning data
print("Cleaning and preparing all data...")
riskCatCol <- "acute_myeloid_leukemia_calgb_cytogenetics_risk_category"
riskFactorCol <- "cytogenetic_abnormalities"

cleaned <- cleanAllData(genExpr=rawData$genExpr,
                        genExprSampleInfo=rawData$genExprSampleInfo,
                        rawDnam=rawData$rawDnam, savePath=dataPath,
                        annLib="Auto", clinical=rawData$clinical,
                        riskCatCol=riskCatCol, riskFactorCol=riskFactorCol,
                        riskHigh="Poor", riskLow="Favorable",
                        verbose=1)

## Prepare toy data
print("Creating a sample of data...")
toyData <- sampleData(Data=rawData, cleanData=cleaned, seed=1)

toyCleanedAml <- toyData$cleaned


data(toyCleanedAml)
class(toyCleanedAml)
```

---

toyComputEloci *A subset of weighted DNA methylation profile of TCGA-LAML data*

Description

Weighted average of DNA methylation profile of 180 acute myeloid leukemia (AML) cases from TCGA-LAML data.

Usage

```
data("toyComputEloci")
```

Format

A list

Value

A list of 4 components.

eigenloci	A matrix of eigenloci values where row names are patient IDs, and column names are gene IDs.
usefuLoci	A list of character vectors, where a gene is an item of the list that stores the corresponding loci information.
lociPigen	A list of Pigengene objects. Names are genes.
distanceToTss	A list of distances of all selected loci to Transcription Start Site(TSS).

Source

https://portal.gdc.cancer.gov/projects/TCGA-LAML

References

Genomic and Epigenomic Landscapes of Adult De Novo Acute Myeloid Leukemia (2013). New England Journal of Medicine 368.22: 2059-2074.

See Also

iNETgrate-package, toyCleanedAml, toyEigengenes

Examples

```
data(toyComputEloci)

## How to reproduce the data:
data(toyCleanedAml)
survivalAml <- toyCleanedAml$survival
plotPath <- file.path(tempdir(), "plots")
dir.create(plotPath, showWarnings=FALSE)

print("Electing genes...")
elected <- electGenes(genExpr=toyCleanedAml$genExpr, dnam=toyCleanedAml$dnam,
                      survival=survivalAml, savePath=tempdir(),
                      locus2gene=toyCleanedAml$locus2gene, verbose=1)

print("Computing eigenloci...")
patientLabel <- setNames(as.character(survivalAml[,"Risk1"]), nm=rownames(survivalAml))
inBoth <- intersect(colnames(toyCleanedAml$dnam), names(patientLabel))
computedEloci <- computEigenloci(dnam=toyCleanedAml$dnam[ ,inBoth],
```

```
                                locus2gene=toyCleanedAml$locus2gene,
                                geNames=elected$unionGenes,
                        Labels=patientLabel[is.element(names(patientLabel), inBoth)],
                                plotPath=plotPath, Label1="High", Label2="Low",
                                verbose=1)

class(computedEloci)
```

---

toyEigengenes *A subset of weighted average of gene expression present in each module*

Description

Weighted average of gene expression of 158 acute myeloid leukemia (AML) cases from TCGA-LAML data that is distributed into 4 feature sets (modules).

Usage

```
data("toyEigengenes")
```

Format

A matrix

Details

The columns and rows are named according to the feature names (modules), and patient IDs, respectively.

Value

It is a 158*4 numeric matrix.

Source

https://portal.gdc.cancer.gov/projects/TCGA-LAML

References

Genomic and Epigenomic Landscapes of Adult De Novo Acute Myeloid Leukemia (2013). New England Journal of Medicine 368.22: 2059-2074.

See Also

iNETgrate-package, computEigengenes, toyCleanedAml, toyComputEloci

Examples

```
data(toyEigengenes)

## How to reproduce data:
data(toyCleanedAml)
data(toyComputEloci)
eigenloci <- toyComputEloci$eigenloci
patientLabel <- as.matrix(toyCleanedAml$survival)[,"Risk1"]

# Path to the network
netPath <- file.path(tempdir(), "net")
dir.create(netPath, showWarnings=FALSE)
madeNetwork <- makeNetwork(genExpr=toyCleanedAml$genExpr, eigenloci=eigenloci,
                           geNames=colnames(eigenloci), mus=c(0.6),
                           outPath=netPath, verbose=0)

eigengenes <- computEigengenes(genExpr=toyCleanedAml$genExpr,
                                 eigenloci=eigenloci, netPath=netPath,
                                 geNames=colnames(eigenloci), Labels=patientLabel,
                                 Label1="Low", Label2="High",
                                 mus=c(0.6), survival=toyCleanedAml$survival,
                                 mu2modules=madeNetwork$mu2modules)

class(eigengenes)
```

---

toyRawAml *A subset of TCGA-LAML data*

Description

Clinical data of 200 acute myeloid leukemia (AML) patients from The Cancer Genome Atlas (TCGA) managed by the NCI and NHGRI (dbGaP Accession: phs000178.v11.p8) is included here. Additionally, DNA methylation profile of 1800 loci for 194 AML cases and gene expression profile of 1077 genes (associated with loci) for 173 AML cases are included here.

Usage

```
data("toyRawAml")
```

Format

A list

Details

The original gene expression and DNA methylation data was produced using Affymetrix U133 Plus 2 platform and Illumina Infinium HumanMethylation450 BeadChip platform, respectively. These TCGA–LAML data are aimed to serve as a foundation for further investigations of AML pathogenesis, classification, and risk stratification.

Value

It is a list of 4 components.

genExpr	A matrix of gene expression profile where rows and columns are genes Symbol, and sample IDs, respectively.
genExprSampleInfo	A dataframe where rows are sample IDs, and columns provide sample information.
rawDnam	A matrix of DNA methylation profile where rows and columns are probe IDs, and sample IDs, respectively.
clinical	A dataframe where rows are patient IDs, and columns provide clinical information.

Source

https://portal.gdc.cancer.gov/projects/TCGA-LAML

References

Genomic and Epigenomic Landscapes of Adult De Novo Acute Myeloid Leukemia (2013). New England Journal of Medicine 368.22: 2059-2074.

See Also

iNETgrate-package, toyCleanedAml, toyComputEloci

Examples

```
## This dataset can be regenerated using the following commands,
##which can take some time to run.

dataProject <- "TCGA-LAML"
dataPath <- file.path(tempdir(), "data", dataProject)
dir.create(savePath)

## Downloading data
rawData <- downloaData(dataProject=dataProject, savePath=dataPath)

## Cleaning data
print("Cleaning and preparing all data...")
riskCatCol <- "acute_myeloid_leukemia_calgb_cytogenetics_risk_category"
riskFactorCol <- "cytogenetic_abnormalities"

cleaned <- cleanAllData(genExpr=rawData$genExpr,
                        genExprSampleInfo=rawData$genExprSampleInfo,
                        rawDnam=rawData$rawDnam, savePath=dataPath,
                        annLib="Auto", clinical=rawData$clinical,
                        riskCatCol=riskCatCol, riskFactorCol=riskFactorCol,
                        riskHigh="Poor", riskLow="Favorable",
                        verbose=1)
```

```
## Prepare toy data
print("Creating sample of data...")
set.seed(seed=1)
toyData <- sampleData(Data=rawData, cleanData=cleaned, seed=1)
toyRawAml <- toyData$rawData


data(toyRawAml)
class(toyRawAml)
```

Index

∗ **Surv**
coxAnalysis, 20
∗ **classif**
iNETgrate, 32
∗ **cluster**
findCore, 29
iNETgrate, 32
makeNetwork, 37
∗ **datasets**
createLocusGene, 21
iNETgrate-package, 3
toyCleanedAml, 48
toyComputEloci, 49
toyEigengenes, 51
toyRawAml, 52
∗ **data**
downloaData, 24
∗ **documentation**
iNETgrate-package, 3
∗ **dplot**
plotKM, 39
plotLociNum, 40
plotLociTss, 41
∗ **glmnet**
coxAnalysis, 20
∗ **graphs**
findCore, 29
∗ **graph**
makeNetwork, 37
∗ **manip**
bestInetgrator, 10
cleanAllData, 11
computeInetgrator, 17
computeUnion, 18
createLocusGene, 21
distanceToTss, 22
electGenes, 25
filterLowCor, 27
findTcgaDuplicates, 31
inferEigengenes, 36
prepareSurvival, 42
preprocessDnam, 44
sample2pat, 45
sampleData, 47
∗ **models**
iNETgrate, 32
iNETgrate-package, 3
∗ **multivariate**
computEigengenes, 13
computEigenloci, 15
∗ **optimize**
iNETgrate, 32
∗ **package**
iNETgrate-package, 3
∗ **survival**
accelFailAnalysis, 4
analyzeSurvival, 6
findAliveCutoff, 28
iNETgrate, 32
plotKM, 39

accelFailAnalysis, 4, *8*
analyzeSurvival, 6, *6*, *10*, *18*, *35*, *43*

bestInetgrator, 10, *18*, *34–36*
blockwiseModules, *3*, *37, 38*

cleanAllData, 11, *25*, *32*, *35*, *43*, *46, 47*
combine.networks, *37, 38*
compute.pigengene, *15*, *17*
computEigengenes, *7, 8*, 13, *35*, *51*
computEigengloci rename *(*computEigengenes*)*, 13
computEigenloci, *10*, 15, *18*, *23*, *36*
computeInetgrator, *10*, 17
computeUnion, 18, *25, 26*
coxAnalysis, 20
createLocusGene, *19*, 21

distanceToTss, *17*, 22, *42*

downloaData, 24

electGenes, *13*, *16, 17*, *19*, 25, *35*, *37, 38*

filterLowCor, *25, 26*, 27
findAliveCutoff, 28
findCore, *16*, 29
findTcgaDuplicates, 31, *46*

GDCdownload, *3*, *25*
GDCprepare, *11*, *25*, *44, 45*
GDCquery, *25*
getAnnotation, *45*
glmnet, *20, 21*

iNETgrate, *3*, 32
iNETgrate-package, 3
inferEigengenes, *8*, *10*, 36

makeNetwork, *7, 8*, *14, 15*, *35*, 37

pickSoftThreshold, *33*, *35*, *37, 38*
plotKM, *6*, *8*, 39
plotLociNum, 40
plotLociTss, 41
predict, *5, 6*
prepareSurvival, *8*, *11*, *13*, *28*, 42
preprocessDnam, *11*, *13*, *19*, 44
project.eigen, *15*, *17*

sample2pat, *11*, *13*, *32*, 45
sampleData, 47
set.seed, *47*
Surv, *40*
survfit, *40*

toyCleanedAml, 48, *50, 51*, *53*
toyComputEloci, 49, *49*, *51*, *53*
toyEigengenes, *50*, 51
toyRawAml, *32*, *35*, *49*, 52