Package 'netDx'

May 9, 2024

Title Network-based patient classifier

Version 1.16.0

Description netDx is a general-purpose algorithm to build a patient classifier from heterogenous patient data. The method converts data into patient similarity networks at the level of features. Feature selection identifies features of predictive value to each class. Methods are provided for versatile predictor design and performance evaluation using standard measures. netDx natively groups molecular data into pathway-level features and connects with Cytoscape for network visualization of pathway themes. For method details see: Pai et al. (2019). netDx: interpretable patient classification using integrated patient similarity networks. Molecular Systems Biology. 15, e8497

Depends R (>= 3.6)

Suggests curatedTCGAData, rmarkdown, testthat, knitr, BiocStyle, RCy3, clusterExperiment, netSmooth, scater

Imports ROCR,pracma,ggplot2,glmnet,igraph,reshape2, parallel,stats,utils,MultiAssayExperiment,graphics,grDevices, methods,BiocFileCache,GenomicRanges, bigmemory,doParallel,foreach, combinat,rappdirs,GenomeInfoDb,S4Vectors, IRanges,RColorBrewer,Rtsne,httr,plotrix

VignetteBuilder knitr

Encoding UTF-8

License MIT + file LICENSE

LazyData false

URL http://netdx.org

biocViews Classification, BiomedicalInformatics, Network, SystemsBiology

RoxygenNote 7.1.2

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

git_branch RELEASE_3_19

git_last_commit 3e00d1b

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-08

Author Shraddha Pai [aut, cre] (<https://orcid.org/0000-0002-1048-581X>),
Philipp Weber [aut],
Ahmad Shah [aut],
Luca Giudice [aut],
Shirley Hui [aut],
Anne Nøhr [ctb],
Indy Ng [ctb],
Ruth Isserlin [aut],
Hussam Kaka [aut],
Gary Bader [aut]

Maintainer Shraddha Pai <shraddha.pai@utoronto.ca>

Contents

`.get_cache` *wrapper function for getting BiocFileCache associated with netDx package*

Description

wrapper function for getting BiocFileCache associated with netDx package

Usage

```
.get_cache()
```

Value

BiocFileCache object associated with netDx

allowedSims *built-in similarity functions*

Description

built-in similarity functions

Usage

```
allowedSims()
```

avgNormDiff *takes average of normdiff of each row in x*

Description

takes average of normdiff of each row in x

Usage

```
avgNormDiff(x)
```

Arguments

x	(numeric) matrix of values, one column per patient (e.g. ages)

Value

symmetric matrix of size ncol(dat) (number of patients) containing pairwise patient similarities

Examples

```
data(xpr)
sim <- avgNormDiff(xpr[,seq_len(2)])
```

`buildPredictor` *Run nested cross-validation on data*

Description

Run nested cross-validation on data

Usage

```
buildPredictor(
  dataList,
  groupList,
  outDir = tempdir(),
  makeNetFunc = NULL,
  sims = NULL,
  featScoreMax = 10L,
  trainProp = 0.8,
  numSplits = 10L,
  numCores,
  JavaMemory = 4L,
  featSelCutoff = 9L,
  keepAllData = FALSE,
  startAt = 1L,
  preFilter = FALSE,
  impute = FALSE,
  preFilterGroups = NULL,
  imputeGroups = NULL,
  logging = "default",
  debugMode = FALSE
)
```

Arguments

`dataList`	(MultiAssayExperiment) sample metadata. Clinical data is in colData() and other input datatypes are in assays() slot. names(groupList) should match names(assays(dataList)). The only exception is clinical data. If a groupList entry is called "clinical", the algorithm will search for corresponding variable names in colData(dataList) (i.e. columns of sample metadata table).
`groupList`	(list of lists) keys are datatypes, and values are lists indicating how units for those datatypes are to be grouped. Keys must match names(assays(dataList)). The only exception is for clinical values. Variables for "clinical" will be extracted from columns of the sample metadata table (i.e. from colData(dataList)). e.g. groupList[["rna"]] could be a list of pathway definitions. So keys(groupList[["rna"]]) would have pathway names, generating one PSN per pathways, and values(groupList[["rna"]]) would be genes that would be grouped for the corresponding pathwayList.
`outDir`	(char) directory where results will be stored. If this directory exists, its contents will be overwritten. Must be absolute path

makeNetFunc	(function) user-defined function for creating the set of input PSN provided to netDx. See createPSN_MultiData()::customFunc.
sims	(list) rules to create similarity networks from input data. Keys are names of data layers and should be identical to names(groupList). Values is either a character for built-in similarity functions; call allowedSims() to see full list; or a custom function.
featScoreMax	(integer) number of CV folds in inner loop
trainProp	(numeric 0 to 1) Percent samples to use for training
numSplits	(integer) number of train/blind test splits (i.e. iterations of outer loop)
numCores	(integer) number of CPU cores for parallel processing
JavaMemory	(integer) memory in (Gb) used for each fold of CV
featSelCutoff	(integer) cutoff for inner-fold CV to call feature-selected in a given split
keepAllData	(logical) if TRUE keeps all intermediate files, even those not needed for assessing the predictor. Use very cautiously as for some designs, each split can result in using 1Gb of data.
startAt	(integer) which of the splits to start at (e.g. if the job aborted part-way through)
preFilter	(logical) if TRUE uses lasso to prefilter dataList within cross-validation loop. Only variables that pass lasso get included. The current option is not recommended for pathway-level features as most genes will be eliminated by lasso. Future variations may allow other prefiltering options that are more lenient.
impute	(logical) if TRUE applies imputation by median within CV
preFilterGroups	(char) vector with subset of names(dataList) to which prefiltering needs to be limited. Allows users to indicate which data layers should be prefiltered using regression and which are to be omitted from this process. Prefiltering uses regression, which omits records with missing values. Structured missingness can result in empty dataframes if missing values are removed from these, which in turn can crash the predictor. To impute missing data, see the 'impute' and 'imputeGroups' parameters.
imputeGroups	(char) If impute set to TRUE, indicate which groups you want imputed.
logging	(char) level of detail with which messages are printed. Options are: 1) none: turn off all messages; 2) all: greatest level of detail (recommended for advanced users, or for debugging); 3) default: print key details (useful setting for most users)
debugMode	(logical) when TRUE runs jobs in serial instead of parallel and prints verbose messages. Also prints system Java calls and prints all standard out and error output associated with these calls.

Details

wrapper function to run netDx with nested cross-validation, with an inner loop of X-fold cross-validation and an outer loop of different random splits of data into train and blind test. The user needs to supply a custom function to create PSN, see createPSN_MultiData(). This wrapper provides flexibility for designs with one or several heterogeneous data types, and one or more ways of

defining patient similarity. For example, designs it handles includes 1) Single datatype, single similarity metric: Expression data -> pathways 2) Single datatype, multiple metrics: Expression data -> pathways (Pearson corr) and single gene networks (normalized difference) 3) Multiple datatypes, multiple metrics: Expression -> Pathways; Clinical -> single or grouped nets

Value

symmetric matrix of size ncol(dat) (number of patients) containing pairwise patient similarities

(list) "inputNets": data.frame of all input network names. Columns are "NetType" (group) and "NetName" (network name). "Split<i>" is the data for train/test split i (i.e. one per train/test split). Each "SplitX" entry contains in turn a list of results for that split. Key-value pairs are: 1) predictions: real and predicted labels for test patients 2) accuracy: percent accuracy of predictions 3) featureScores: list of length g, where g is number of patient classes. scores for all features following feature selection, for corresponding class. 4) featureSelected: list of length g (num patient classes). List of selected features for corresponding patient class, for that train/test split. Side effect of generating predictor-related data in <outDir>.

Examples

```
library(curatedTCGAData)
library(MultiAssayExperiment)
curatedTCGAData(diseaseCode="BRCA", assays="*",dry.run=TRUE,version="1.1.38")

# fetch mrna, mutation data
brca <- curatedTCGAData("BRCA",c("mRNAArray"),FALSE,version="1.1.38")

# get subtype info
pID <- colData(brca)$patientID
pam50 <- colData(brca)$PAM50.mRNA
staget <- colData(brca)$pathology_T_stage
st2 <- rep(NA,length(staget))
st2[which(staget %in% c("t1","t1a","t1b","t1c"))] <- 1
st2[which(staget %in% c("t2","t2a","t2b"))] <- 2
st2[which(staget %in% c("t3","t3a"))] <- 3
st2[which(staget %in% c("t4","t4b","t4d"))] <- 4
pam50[which(!pam50 %in% "Luminal A")] <- "notLumA"
pam50[which(pam50 %in% "Luminal A")] <- "LumA"
colData(brca)$ID <- pID
colData(brca)$STAGE <- st2
colData(brca)$STATUS <- pam50

# keep only tumour samples
idx <- union(which(pam50 == "Normal-like"), which(is.na(st2)))
cat(sprintf("excluding %i samples\n", length(idx)))

tokeep <- setdiff(pID, pID[idx])
brca <- brca[,tokeep,]

pathList <- readPathways(fetchPathwayDefinitions(month=10,year=2020))
brca <- brca[,,1] # keep only clinical and mRNA data
```

```
# remove duplicate arrays
smp <- sampleMap(brca)
samps <- smp[which(smp$assay=="BRCA_mRNAArray-20160128"),]
notdup <- samps[which(!duplicated(samps$primary)),"colname"]
brca[[1]] <- brca[[1]][,notdup]

groupList <- list()
groupList[["BRCA_mRNAArray-20160128"]] <- pathList[seq_len(3)]
groupList[["clinical"]] <- list(
age="patient.age_at_initial_pathologic_diagnosis",
 stage="STAGE")
makeNets <- function(dataList, groupList, netDir,...) {
    netList <- c()
    # make RNA nets: group by pathway
    if (!is.null(groupList[["BRCA_mRNAArray-20160128"]])) {
    netList <- makePSN_NamedMatrix(dataList[["BRCA_mRNAArray-20160128"]],
                rownames(dataList[["BRCA_mRNAArray-20160128"]]),
                groupList[["BRCA_mRNAArray-20160128"]],
                netDir,verbose=FALSE,
                writeProfiles=TRUE,...)
    netList <- unlist(netList)
    cat(sprintf("Made %i RNA pathway nets\n", length(netList)))
    }

    # make clinical nets,one net for each variable
    netList2 <- c()
    if (!is.null(groupList[["clinical"]])) {
    netList2 <- makePSN_NamedMatrix(dataList$clinical,
        rownames(dataList$clinical),
        groupList[["clinical"]],netDir,
        simMetric="custom",customFunc=normDiff, # custom function
        writeProfiles=FALSE,
        sparsify=TRUE,verbose=TRUE,...)
    }
    netList2 <- unlist(netList2)
    cat(sprintf("Made %i clinical nets\n", length(netList2)))
    netList <- c(netList,netList2)
    cat(sprintf("Total of %i nets\n", length(netList)))
    return(netList)
}

# takes 10 minutes to run
#out <- buildPredictor(dataList=brca,groupList=groupList,
#   makeNetFunc=makeNets, ### custom network creation function
#   outDir=paste(tempdir(),"pred_output",sep=getFileSep()), ## absolute path
#   numCores=16L,featScoreMax=2L, featSelCutoff=1L,numSplits=2L)
```

`buildPredictor_sparseGenetic`

Performs feature selection using multiple resamplings of the data

---

Description

Performs feature selection using multiple resamplings of the data

Usage

```
buildPredictor_sparseGenetic(
  phenoDF,
  cnv_GR,
  predClass,
  group_GRList,
  outDir = tempdir(),
  numSplits = 3L,
  featScoreMax = 10L,
  filter_WtSum = 100L,
  enrichLabels = TRUE,
  enrichPthresh = 0.07,
  numPermsEnrich = 2500L,
  minEnr = -1,
  numCores = 1L,
  FS_numCores = NULL,
  ...
)
```

Arguments

phenoDF	(data.frame) sample metadat. patient ID,STATUS
cnv_GR	(GRanges) genetic events. Must contain "ID" column mapping the event to a patient. ID must correspond to the ID column in phenoDF
predClass	(char) patient class to predict
group_GRList	(list) List of GRangesList indicating grouping rules for CNVs. For example, in a pathway-based design, each key value would be a pathway name, and the value would be a RangesList containing coordinates of the member genes
outDir	(char) path to dir where results should be stored. Results for resampling i are under <outDir>/part<i>, while predictor evaluation results are directly in outDir.
numSplits	(integer) number of data resamplings to use
featScoreMax	(integer) max score for features in feature selection
filter_WtSum	(numeric between 5-100) Limit to top-ranked networks such that cumulative weight is less than this parameter. e.g. If filter_WtSum=20, first order networks by decreasing weight; then keep those whose cumulative weight <= 20.
enrichLabels	(logical) if TRUE, applies label enrichment to train networks
enrichPthresh	(numeric between 0 and 1) networks with label enrichment p-value below this threshold pass enrichment
numPermsEnrich	(integer) number of permutations for label enrichment
minEnr	(integer -1 to 1) minEnr param in enrichLabelsNets()

numCores	(integer) num cores for parallel processing
FS_numCores	(integer) num cores for running GM. If NULL, is set to max(1,numCores-1). Set to a lower value if the default setting gives out-of-memory error. This may happen if networks are denser than expected
...	params for runFeatureSelection()

Details

This function is used for feature selection of patient networks, using multiple resamplings of input data. It is intended for use in the scenario where patient networks are sparse and binary. This function should be called after defining all patient networks. It performs the following steps: For i = 1..numSplits randomly split patients into training and test (optional) filter training networks to exclude random-like networks compile features into database for cross-validation score networks out of 10 end using test samples from all resamplings, measure predictor performance.

In short, this function performs all steps involved in building and evaluating the predictor.

Value

(list) Predictor results 1) phenoDF (data.frame): subset of phenoDF provided as input, but limited to patients that have at least one event in the universe of possibilities e.g. if using pathway-level features, then this table excludes patients with zero CNVs in pathways 2) netmat (data.frame): Count of genetic events by patients (rows) in pathways (columns). Used as input to the feature selection algorithm 3) pathwayScores (list): pathway scores for each of the data splits. Each value in the list is a data.frame containing pathway names and scores. 4) enrichedNets (list): This entry is only found if enrichLabels is set to TRUE. It contains the vector of features that passed label enrichment in each split of the data. 5 - 9) Output of RR_featureTally: 5) cumulativeFeatScores: pathway name, cumulative score over N-way data resampling. 6) performance_denAllNets: positive,negative calls at each cutoff: network score cutoff (score); num networks at cutoff (numPathways) ; total +, ground truth (pred_tot); + calls (pred_ol); + calls as pct of total (pred_pct); total -, ground truth (other_tot) ; - calls (other_ol) ; - calls as pct of total (other_pct) ; ratio of pred_pct and other_pct (rr) ; min. pred_pct in all resamplings (pred_pct_min) ; max pred_pct in all resamplings (pred_pct_max) ; min other_pct in all resamplings (other_pct_min); max other_pct in all resamplings (other_pct_max) 7) performance_denEnrichedNets: positive, negative calls at each cutoff label enrichment option: format same as performance_denAllNets. However, the denominator here is limited to patients present in networks that pass label enrichment 8) resamplingPerformance: breakdown of performance for each of the resamplings, at each of the cutoffs. This is a list of length 2, one for allNets and one for enrichedNets. The value is a matrix with (resamp * 7) columns and S rows, one row per score. The columns contain the following information per resampling: 1) pred_total: total num patients of predClass 2) pred_OL: num of pred_total with a CNV in the selected net 3) pred_OL_pct: 2) divided by 1) (percent) 4) other_total: total num patients of other class(non-predClass) 5) other_OL: num of other_total with CNV in selected net 6) other_OL_pct: 5) divided by 4) (percent) 7) relEnr: 6) divided by 3).

Examples

```
suppressMessages(require(GenomicRanges))
suppressMessages(require(BiocFileCache))

# read CNV data
```

```
phenoFile <- system.file("extdata","AGP1_CNV.txt",package="netDx")
pheno   <- read.delim(phenoFile,sep="\t",header=TRUE,as.is=TRUE)
colnames(pheno)[1] <- "ID"
pheno <- pheno[!duplicated(pheno$ID),]

# create GRanges with patient CNVs
cnv_GR    <- GRanges(pheno$seqnames,IRanges(pheno$start,pheno$end),
                        ID=pheno$ID,LOCUS_NAMES=pheno$Gene_symbols)

# get gene coordinates
geneURL <- paste("http://download.baderlab.org/netDx/",
"supporting_data/refGene.hg18.bed",sep="")
cache <- rappdirs::user_cache_dir(appname = "netDx")
bfc <- BiocFileCache::BiocFileCache(cache,ask=FALSE)
geneFile <- bfcrpath(bfc,geneURL)
genes <- read.delim(geneFile,sep="\t",header=FALSE,as.is=TRUE)
genes <- genes[which(genes[,4]!=""),]
gene_GR     <- GRanges(genes[,1],IRanges(genes[,2],genes[,3]),
   name=genes[,4])

# create GRangesList of pathways
pathFile <- fetchPathwayDefinitions("February",2021,verbose=TRUE)
pathwayList <- readPathways(pathFile)
path_GRList <- mapNamedRangesToSets(gene_GR,pathwayList)

#### uncomment to run - takes 5 min
#out <- buildPredictor_sparseGenetic(pheno, cnv_GR, "case",
#                              path_GRList,outDir,
#                              numSplits=3L, featScoreMax=3L,
#                              enrichLabels=TRUE,numPermsEnrich=20L,
#                              numCores=1L)
#summary(out)
#head(out$cumulativeFeatScores)
```

`callFeatSel` *Return feature selected nets based on given criteria*

Description

Return feature selected nets based on given criteria

Usage

```
callFeatSel(netScores, fsCutoff, fsPctPass)
```

Arguments

`netScores`	(matrix) matrix of net scores
`fsCutoff`	(integer) net must score at least this much in a split to 'pass' the threshold

`fsPctPass` (numeric 0 to 1) net must pass at least this percent of splits to be considered feature-selected

Details

given the output of genNetScores.R and criteria for defining feature-selected (FS) nets, returns subset of nets that pass criteria. Net must score <fsCutoff> for at least <fsPctPass> considered feature-selected.

Value

(char) names of nets that pass feature-selection

Examples

```
data(featScores)
passed <- lapply(featScores, function(x) {
   callFeatSel(x,10,0.7) # score 10/10 in >=70% of trials
})
print(passed)
```

---

callOverallSelectedFeatures

Wrapper to call selected features

Description

Wrapper to call selected features

Usage

```
callOverallSelectedFeatures(
  featScores,
  featureSelCutoff,
  featureSelPct,
  cleanNames = TRUE
)
```

Arguments

`featScores` (list of lists): matrix of feature scores across all splits, separated by patient label. First level: patient labels. Second level: matrix of scores for corresponding label.

`featureSelCutoff` (integer) cutoff score for feature selection. A feature must have minimum of this score for specified fraction of splits (see featureSelPct) to pass.

featureSelPct	(numeric between 0 and 1) cutoff percent for feature selection. A feature must have minimum score of featureSelCutoff for featureSelPct of train/test splits to pass.
cleanNames	(logical) remove internal suffixes for human readability

Details

Calls features that are consistently high-scoring for predicting each class. The context for this is as follows: The original model runs feature selection over multiple splits of data into train/test samples, and each such split generates scores for all features. This function identifies features with scores that exceed a threshold for a fraction of train/test splits; the threshold and fraction are both user-specified. This function is called by the wrapper getResults(), which returns both the matrix of feature scores across splits and list of features that pass the user-specified cutoffs.

Value

(list) Feature scores for all splits, plus those passing selection for overall predictor featScores: (matrix) feature scores for each split selectedFeatures: (list) features passing selection for each class; one key per class

Examples

```
pathways <- paste("PATHWAY_",1:100,sep="")
highrisk <- list()
lowrisk <- list()
for (k in 1:10) {
highrisk[[k]] <- data.frame(PATHWAY_NAME=pathways,
        SCORE=floor(runif(length(pathways),min=0,max=10)),
stringsAsFactors=FALSE);
    lowrisk[[k]] <- data.frame(PATHWAY_NAME=pathways,
        SCORE=floor(runif(length(pathways),min=0,max=10)),
stringsAsFactors=FALSE);
}
names(highrisk) <- sprintf("Split%i",1:length(highrisk))
names(lowrisk) <- sprintf("Split%i",1:length(lowrisk))
callOverallSelectedFeatures(list(highrisk=highrisk,lowrisk=lowrisk), 5,0.5)
```

---

checkMakeNetFuncSims *internal test function to check validity of makeNetFunc and sims*

Description

internal test function to check validity of makeNetFunc and sims

Usage

```
checkMakeNetFuncSims(makeNetFunc, sims, groupList)
```

Arguments

makeNetFunc	(function) makeNetFunc from buildPredictor()
sims	(list) sims from buildPredictor()
groupList	(list) groupList from buildPredictor()s

Details

User must provide either makeNetFunc or sims. This function confirms this.

Value

(list) cleaned values for makeNetFunc and Sims

---

checkSimValid *checks if provided similarity functions are valid. Returns error if not*

Description

checks if provided similarity functions are valid. Returns error if not

Usage

```
checkSimValid(sims)
```

Arguments

sims	(list) keys are layer names, values are functions or characters (names of built-in similarity functions)

Value

TRUE if all pass check. Else throws error.

---

cleanPathwayName *Clean pathway name so it can be a filename.*

Description

Clean pathway name so it can be a filename.

Usage

```
cleanPathwayName(curP)
```

Arguments

curP	(char) pathway name

Value

(char) Cleaned pathway name

Examples

```
cleanPathwayName('7-(3-AMINO-3-CARBOXYPROPYL)-WYOSINE BIOSYNTHESIS%HUMANC')
```

---

cnv_GR *CNV locations for breast cancer (subset)*

Description

Subset of CNV locations for TCGA breast tumour. Each range is associated with a patient (ID)

Usage

```
data(cnv_GR)
```

Source

The Cancer Genome Atlas. (2012). Nature 490:61-70.

References

The Cancer Genome Atlas. (2012). Nature 490:61-70.

Examples

```
data(cnv_GR)
head(cnv_GR)
```

cnv_netPass *Vector of pathways that pass class enrichment*

Description

Vector of pathways that pass class enrichment

Usage

```
data(cnv_netPass)
```

Examples

```
data(cnv_netPass)
head(cnv_netPass)
```

cnv_netScores *List of pathway-level feature selection scores*

Description

List of pathway-level feature selection scores

Usage

```
data(cnv_netScores)
```

Examples

```
data(cnv_netScores)
summary(cnv_netScores)
head(cnv_netScores[[1]])
```

cnv_patientNetCount *Binary matrix of patient occurrence in networks*

Description

Binary matrix of patient occurrence in networks

Usage

```
data(cnv_patientNetCount)
```

Examples

```
data(cnv_patientNetCount)
head(cnv_patientNetCount)
```

cnv_pheno *data.frame of patient labels and status for CNV example*

Description

data.frame of patient labels and status for CNV example

Usage

```
data(cnv_pheno)
```

Examples

```
data(cnv_pheno)
head(cnv_pheno)
```

cnv_TTstatus *list of train/test statuses for CNV example*

Description

list of train/test statuses for CNV example

Usage

```
data(cnv_TTstatus)
```

Examples

```
data(cnv_TTstatus)
head(cnv_TTstatus)
```

compareShortestPath *compare intra-cluster shortest distance to overall shortest distance of the network*

Description

compare intra-cluster shortest distance to overall shortest distance of the network

Usage

```
compareShortestPath(net, pheno, plotDist = FALSE, verbose = TRUE)
```

Arguments

net	(data.frame) network on which to compute shortest path. SOURCE, TARGET, WEIGHTS. Column names are ignored but expects a header row. Distances will be computed based on the third column
pheno	(data.frame) Node information. ID (node name) and GROUP (cluster name)
plotDist	(logical) if TRUE, creates a violin plot showing the shortest path distributions for each group.
verbose	(logical) print messages

Details

Uses Dijkstra's algorithm for weighted edges. Pairwise nodes with infinite distances are excluded before computing average shortest path for a network. This function requires the igraph package to be installed.

Value

(list) Two lists, 'avg' and 'all'. keys are cluster names. values for 'avg' are mean shortest path ; for 'all', are all pairwise shortest paths for subnetworks that contain only the edges where source and target both belong to the corresponding cluster. In addition, there is an 'overall' entry for the mean shortest distance for the entire network.

Examples

```
data(silh);
colnames(silh$net)[3] <- 'weight'
compareShortestPath(silh$net, silh$groups)
```

---

compileFeatures *Create GeneMANIA database*

Description

Create GeneMANIA database

Usage

```
compileFeatures(
  netDir,
  outDir = tempdir(),
  simMetric = "pearson",
  netSfx = "txt$",
  verbose = TRUE,
  numCores = 1L,
  P2N_threshType = "off",
  P2N_maxMissing = 100,
```

```
  JavaMemory = 4L,
  altBaseDir = NULL,
  debugMode = FALSE,
  ...
)
```

Arguments

netDir	(char) path to dir with input networks/profiles. All networks in this directory will be added to the GM database. Note: This needs to be an absolute path, not relative.
outDir	(char) path to dir in which GeneMANIA database is created. The database will be under outDir/dataset.
simMetric	(char) similarity measure to use in converting profiles to interaction networks.
netSfx	(char) pattern for finding network files in netDir.
verbose	(logical) print messages
numCores	(integer) num cores for parallel processing
P2N_threshType	(char) Most users shouldn't have to change this. ProfileToNetworkDriver's threshold option. One of 'offlauto'. unit testing
P2N_maxMissing	(integer 5-100)
JavaMemory	(integer) Memory for GeneMANIA (in Gb)
altBaseDir	(char) Only use this if you're developing netDx. Used in unit tests
debugMode	(logical) when TRUE runs jobs in serial instead of parallel and prints verbose messages. Also prints system Java calls and prints all standard out and error output associated with these calls.
...	params for writeQueryBatchFile()

Details

Creates a generic_db for use with GeneMania QueryRunner. The database is in tab-delimited format, and indexes are built using Apache lucene. NOTE: This pipeline expects input in the form of interaction networks and not profiles. Profile tables have patient-by-datapoint format (e.g. patient-by-genotype) Interaction networks have pairwise similarity measures: <PatientA> <PatientB><similarity> Documentation: https://github.com/GeneMANIA/pipeline/wiki/GenericDb

Value

(list). 'dbDir': path to GeneMANIA database 'netDir': path to directory with interaction networks. If profiles are provided, this points to the INTERACTIONS/ subdirectory within the text-based GeneMANIA generic database If the DB creation process results in an erorr, these values return NA

Examples

```
data(xpr,pheno)
pathwayList <- list(pathA=rownames(xpr)[1:10],
pathB=rownames(xpr)[21:50])

dataList <- list(rna=xpr)  #only one layer type
groupList <- list(rna=pathwayList) # group genes by pathways

makeNets <- function(dataList, groupList, netDir,...) {
    netList <- makePSN_NamedMatrix(dataList[['rna']],
rownames(dataList[['rna']]),
         groupList[['rna']],netDir,verbose=FALSE,
writeProfiles=TRUE,...)
    unlist(netList)
}
tmpDir <- tempdir(); netDir <- paste(tmpDir,"nets",
sep=getFileSep())
if (file.exists(netDir)) unlink(netDir,recursive=TRUE)
dir.create(netDir,recursive=TRUE)

pheno_id <- setupFeatureDB(pheno,netDir)
netList <- createPSN_MultiData(dataList=dataList, groupList=groupList,
    pheno=pheno_id,netDir=netDir,makeNetFunc=makeNets,verbose=TRUE)

outDir <- paste(tmpDir,'dbdir',sep=getFileSep());
dir.create(outDir)
dbDir <- compileFeatures(netDir,outDir)
```

---

compileFeatureScores *Tally the score of networks through cross-validation*

Description

Tally the score of networks through cross-validation

Usage

```
compileFeatureScores(fList, filter_WtSum = 100, verbose = FALSE)
```

Arguments

fList	(char) Vector of paths to GeneMANIA NRANK files
filter_WtSum	(numeric between 5-100) Limit to top-ranked networks such that cumulative weight is less than this parameter. e.g. If filter_WtSum=20, first order networks by decreasing weight; then keep those whose cumulative weight <= 20.
verbose	(logical) print messages

Value

(data.frame) Feature name and score; includes features that occur at least once in `fList`.

Examples

```
netDir <- system.file("extdata","GM_NRANK",package="netDx")
netFiles <- sprintf('%s/%s', netDir,dir(netDir,pattern='NRANK$'))
pTally <- compileFeatureScores(netFiles,verbose=TRUE)
print(head(pTally))
```

---

confmat *Confusion matrix example*

Description

Sample table of True/False Positives and Negatives for various feature selection cutoffs tp: true positive rate, fp: false positive rate, tn: true negative rate, fn: false negative rate

Usage

```
data(confmat)
```

Examples

```
data(confmat)
head(confmat)
```

---

confusionMatrix *Make confusion matrix*

Description

Make confusion matrix

Usage

```
confusionMatrix(model)
```

Arguments

`model`	(list) output of buildPredictor()

Details

Creates a confusion matrix, a square matrix which indicates the fraction of times patients in a class are correctly classified, versus misclassified as each of the other classes. Here, the confusion matrix is computed once per train-test split and the average is displayed. For this reason, the fractions may not cleanly add up to 100

Value

(list) confusion matrix for all train/test splits and final averaged matrix Side effect of plotting the averaged matrix.

Examples

```
data(toymodel)
confusionMatrix(toymodel)
```

---

convertProfileToNetworks

Convert profiles to interaction networks before integration

Description

Convert profiles to interaction networks before integration

Usage

```
convertProfileToNetworks(
  netDir,
  outDir = tempdir(),
  simMetric = "pearson",
  numCores = 1L,
  JavaMemory = 4L,
  GM_jar = NULL,
  P2N_threshType = "off",
  P2N_maxMissing = 100,
  netSfx = "txt$",
  debugMode = FALSE
)
```

Arguments

netDir	(char) directory with .profile files
outDir	(char) path to directory where interaction networks are to be printed
simMetric	(char) similarity measure to use in converting profiles to interaction networks.
numCores	(integer) number of cores for parallel processing
JavaMemory	(integer) Memory for GeneMANIA (in Gb)
GM_jar	(char) path to GeneMANIA jar file
P2N_threshType	(char) Most users shouldn't have to change this. ProfileToNetworkDriver's threshold option. One of 'offlauto'. unit testing
P2N_maxMissing	(integer 5-100)
netSfx	(char) pattern for finding network files in netDir.
debugMode	(logical) if TRUE runs profile generation in serial rather than parallel, allowing debugging

Details

In preparation for network integration. When using GeneMANIA's built-in functionality to create PSN using ProfileToNetworkDriver, this step needs to run to process profiles to networks. These are currently used for Pearson correlation-based networks and those using mutual information.

Value

No value. Side effect of creating interaction networks in outDir.

`convertToMAE` *Wrapper that converts an input list into a MultiAssayExperiment object*

Description

Wrapper that converts an input list into a MultiAssayExperiment object

Usage

```
convertToMAE(dataList)
```

Arguments

`dataList`	(list) input key-value pairs (keys: data types, values: data in the form of matrices/dataframes); must have a key-value pair that corresponds to patient IDs/metadata labelled pheno.

Details

This function takes in a list of key-value pairs (keys: data types, values: matrices/dataframes) and calls the necessary functions from the MultiAssayExperiment package to incorporate the values from the input list into a MultiAssayExperiment object, transforming the values according to the keys. If duplicate sample names are found in the assay data, only the first instance is kept.

Value

MAE (MultiAssayExperiment) data from input list incorporated into a MultiAssayExperiment object, compatible with further analysis using the netDx algorithm.

Examples

```
data(xpr, pheno)

# Generate random proteomic data
prot <- matrix(rnorm(100*20), ncol=20)
colnames(prot) <- sample(pheno$ID, 20)
rownames(prot) <- sprintf("protein%i",1:100)
```

```
myList <- list(rna = xpr, proteomic = prot, pheno = pheno)

MAE <- convertToMAE(myList)
```

---

countIntType *Counts the number of (+,+) and (+,-) interactions in a single network*

Description

Counts the number of (+,+) and (+,-) interactions in a single network

Usage

```
countIntType(inFile, plusID, minusID)
```

Arguments

inFile	(char) path to interaction networks
plusID	(char) vector of + nodes
minusID	(char) vector of - nodes

Value

(numeric of length 2) Number of (+,+) interactions, and non-(+,+) interactions (i.e. (+,-) and (-,-) interactions)

Examples

```
d <- tempdir()
# write PSN
m1 <- matrix(c("P1","P1","P2","P2","P3","P4",1,1,1),byrow=FALSE,ncol=3)
write.table(m1,file=paste(d,"net1.txt",sep=getFileSep()),
sep="\t",
col.names=FALSE,row.names=FALSE,quote=FALSE)

countIntType(paste(d,"net1.txt",sep=getFileSep()),c("P1","P2","P3"),
c("P4","P5"))
```

countIntType_batch *Counts number of (+,+) and (+,-) interactions in a set of networks*

Description

Counts number of (+,+) and (+,-) interactions in a set of networks

Usage

```
countIntType_batch(
  inFiles,
  plusID,
  minusID,
  tmpDir = tempdir(),
  enrType = "binary",
  numCores = 1L
)
```

Arguments

inFiles	(char) path to interaction networks to process
plusID	(char) IDs of + nodes
minusID	(char) IDs of - nodes
tmpDir	(char) path to dir where temporary files can be stored
enrType	(char) see getEnr.R
numCores	(integer) number of cores for parallel processing

Value

(matrix) two columns, one row per network If enrType="binary", number of (+,+) and other interactions Otherwise if enrType="corr" mean edge weight of (+,+) edges and of other edges

Examples

```
d <- tempdir()
# write PSN
m1 <- matrix(c("P1","P1","P2","P2","P3","P4",1,1,1),byrow=FALSE,ncol=3)
write.table(m1,file=paste(d,"net1.txt",sep=getFileSep()),sep="\t",
col.names=FALSE,row.names=FALSE,quote=FALSE)
m2 <- matrix(c("P3","P4",1),nrow=1)
write.table(m2,file=paste(d,"net2.txt",sep=getFileSep()),sep="\t",
col.names=FALSE,row.names=FALSE,quote=FALSE)

countIntType_batch(paste(d,c("net1.txt","net2.txt"),sep=getFileSep()),
c("P1","P2","P3"),c("P4","P5"))
```

countPatientsInNet *Count number of patients in a network*

Description

Count number of patients in a network

Usage

```
countPatientsInNet(netDir, fList, ids)
```

Arguments

netDir	(char) dir with network set
fList	(char) filenames of interaction networks to count in
ids	(char) patient IDs to look for

Details

This functionality is needed to count patient overlap when input data is in a form that results in highly missing data, rather than when the same measures are available for almost all patients. An example application is when patient networks are based on unique genomic events in each patients (e.g. CNVs or indels), rather than 'full-matrix' data (e.g. questionnaires or gene expression matrices). The former scenario requires an update in the list of eligible networks each time some type of patient subsetting is applied (e.g. label enrichment, or train/test split). A matrix with patient/network membership serves as a lookup table to prune networks as feature selection proceeds

Value

(matrix) Size P by N, where P is num patients and N is number of networks networks; a[i,j] =1 if patient i in network j, else 0

Examples

```
d <- tempdir()
pids <- paste("P",1:5,sep="")
m1 <- matrix(c("P1","P1","P2","P2","P3","P4",1,1,1),
byrow=FALSE,ncol=3)
write.table(m1,
file=paste(d,"net1.txt",sep=getFileSep()),sep="\t",
col.names=FALSE,row.names=FALSE,quote=FALSE)
m2 <- matrix(c("P3","P4",1),nrow=1)
write.table(m2,
file=paste(d,"net2.txt",sep=getFileSep()),sep="\t",
col.names=FALSE,row.names=FALSE,quote=FALSE)
x <- countPatientsInNet(d,c("net1.txt","net2.txt"), pids)
```

createNetFuncFromSimList *Create PSN from provided similarities*

Description

Create PSN from provided similarities

Usage

```
createNetFuncFromSimList(
  dataList,
  groupList,
  netDir,
  sims,
  verbose = TRUE,
  ...
)
```

Arguments

dataList	(list) patient data, output of dataList2List()
groupList	(list) measure groupings. Keys match assays(dataList) and are usually different data sources. Values for each are a list of networks with user-provided groupings. See groupList in buildPredictor() for details.
netDir	(char) path to directory where networks are to be created
sims	(list) keys must be identical to those of groupList. Values are either of type character, used for built-in similarity functions, or are functions, when a custom function is provided.
verbose	(logical) print messages
...	values to be passed to PSN creation functions such as makePSN_NamedMatrix().

Details

Called by CreatePSN_MultiData(), this is the function that converts user-provided simlarity metrics to internal netDx function calls to generate nets.

createPSN_MultiData *Wrapper to create custom input features (patient similarity networks)*

Description

Wrapper to create custom input features (patient similarity networks)

Usage

```
createPSN_MultiData(
  dataList,
  groupList,
  pheno,
  netDir = tempdir(),
  filterSet = NULL,
  verbose = TRUE,
  makeNetFunc = NULL,
  sims = NULL,
  ...
)
```

Arguments

dataList	(list) key is datatype (e.g. clinical, rna, etc.,), value is table or RangedData) Note that unit names should be rownames of the data structure. e.g If dataList$rna contains genes, rownames(dataList) = gene names
groupList	(list) key is datatype; value is a list of unit groupings for that datatype. e.g. If rna data will be grouped by pathways, then groupList$rna would have pathway names as keys, and member genes as units. Each entry will be converted into a PSN.
pheno	(data.frame) mapping of user-provided patient identifiers (ID) with internally-generated identifiers.
netDir	(char) path to directory where networks will be stored
filterSet	(char) vector of networks to include
verbose	(logical) print messages
makeNetFunc	(function) custom user-function to create PSN. Must take dataList,groupList,netDir as parameters. Must check if a given groupList is empty (no networks to create) before the makePSN call for it. This is to avoid trying to make nets for datatypes that did not pass feature selection
sims	(list) Similarity metric settings for patient data. Keys must be identical to those of groupList. Values are either of type character, used for built-in similarity functions, or are functions, when a custom function is provided.
...	other parameters to makePSN_NamedMatrix() or makePSN_RangedSets()

Value

(char) vector of network names. Side effect of creating the nets

Examples

```
library(curatedTCGAData)
library(MultiAssayExperiment)
curatedTCGAData(diseaseCode='BRCA', assays='*',dry.run=TRUE,version="1.1.38")

# fetch mrna, mutation data
brca <- curatedTCGAData('BRCA',c('mRNAArray'),FALSE,version="1.1.38")

# get subtype info
pID <- colData(brca)$patientID
pam50 <- colData(brca)$PAM50.mRNA
staget <- colData(brca)$pathology_T_stage
st2 <- rep(NA,length(staget))
st2[which(staget %in% c('t1','t1a','t1b','t1c'))] <- 1
st2[which(staget %in% c('t2','t2a','t2b'))] <- 2
st2[which(staget %in% c('t3','t3a'))] <- 3
st2[which(staget %in% c('t4','t4b','t4d'))] <- 4
pam50[which(!pam50 %in% 'Luminal A')] <- 'notLumA'
pam50[which(pam50 %in% 'Luminal A')] <- 'LumA'
colData(brca)$ID <- pID
colData(brca)$STAGE <- st2
colData(brca)$STATUS <- pam50

# keep only tumour samples
idx <- union(which(pam50 == 'Normal-like'), which(is.na(st2)))
cat(sprintf('excluding %i samples\n', length(idx)))

tokeep <- setdiff(pID, pID[idx])
brca <- brca[,tokeep,]

pathList <- readPathways(fetchPathwayDefinitions("October",2020))

brca <- brca[,,1] # keep only clinical and mRNA data

# remove duplicate arrays
smp <- sampleMap(brca)
samps <- smp[which(smp$assay=='BRCA_mRNAArray-20160128'),]
notdup <- samps[which(!duplicated(samps$primary)),'colname']
brca[[1]] <- brca[[1]][,notdup]

groupList <- list()
groupList[['BRCA_mRNAArray-20160128']] <- pathList[seq_len(3)]
makeNets <- function(dataList, groupList, netDir,...) {
    netList <- c()
    # make RNA nets: group by pathway
    if (!is.null(groupList[['BRCA_mRNAArray-20160128']])) {
    netList <- makePSN_NamedMatrix(dataList[['BRCA_mRNAArray-20160128']],
```

```
                rownames(dataList[['BRCA_mRNAArray-20160128']]),
                groupList[['BRCA_mRNAArray-20160128']],
                netDir,verbose=FALSE,
                writeProfiles=TRUE,...)
    netList <- unlist(netList)
    cat(sprintf('Made %i RNA pathway nets\n', length(netList)))
    }

    cat(sprintf('Total of %i nets\n', length(netList)))
    return(netList)
}

exprs <- experiments(brca)
datList2 <- list()
for (k in seq_len(length(exprs))) {
 tmp <- exprs[[k]]
 df <- sampleMap(brca)[which(sampleMap(brca)$assay==names(exprs)[k]),]
 colnames(tmp) <- df$primary[match(df$colname,colnames(tmp))]
 tmp <- as.matrix(assays(tmp)[[1]]) # convert to matrix
 datList2[[names(exprs)[k]]]<- tmp
}
pheno <- colData(brca)[,c('ID','STATUS')]
netDir <- tempdir()
pheno_id <- setupFeatureDB(colData(brca),netDir)
createPSN_MultiData(dataList=datList2,groupList=groupList,
 pheno=pheno_id,
 netDir=netDir,makeNetFunc=makeNets,numCores=1)
```

`dataList2List` *Convert MultiAssayExperiment object to list and data.frame*

Description

Convert MultiAssayExperiment object to list and data.frame

Usage

```
dataList2List(dat, groupList)
```

Arguments

`dat`	(MultiAssayExperiment) Patient data and metadata
`groupList`	(list) variable groupings used for feature construction. See groupList arg in buildPredictor().

Details

Used by internal routines in netDx

Value

(list) Keys are: 1) assays: list of matrices, each corresponding to data from a particular layer 2) pheno: (data.frame) sample metadata

Examples

```
data(xpr,pheno)
require(MultiAssayExperiment)
objlist <- list("RNA"=SummarizedExperiment(xpr))
mae <- MultiAssayExperiment(objlist,pheno)
groupList <- list(RNA=rownames(xpr))
dl <- dataList2List(mae,groupList)
summary(dl)
```

---

enrichLabelNets *Score networks based on their edge bias towards (+,+) interactions*

Description

Score networks based on their edge bias towards (+,+) interactions

Usage

```
enrichLabelNets(
  netDir,
  pheno_DF,
  outDir,
  numReps = 50L,
  minEnr = -1,
  outPref = "enrichLabelNets",
  verbose = TRUE,
  setSeed = 42L,
  enrType = "binary",
  numCores = 1L,
  predClass,
  tmpDir = tempdir(),
  netGrep = "_cont.txt$",
  getShufResults = FALSE,
  ...
)
```

Arguments

netDir	(char) path to dir containing all networks
pheno_DF	(data.frame) for details see `getEnr()`
outDir	(char) path to dir where output/log files are written

numReps	(integer) Max num reps for shuffling class status. Adaptive permutation is used so in practice, few networks would be evaluated to this extent
minEnr	(numeric from -1 to 1) Only include networks with ENR value greater than this threshold.
outPref	(char) prefix for log file (not counting the dir name)
verbose	(logical) print messages
setSeed	(integer) if not NULL, integer is set as seed to ensure reproducibility in random number generation
enrType	(char) see getEnr()
numCores	(integer) num cores for parallel ENR computation of all networks
predClass	(char) see `getEnr()`
tmpDir	(char) path to dir where temporary work can be stored
netGrep	(char) pattern to grep for network files in netDir
getShufResults	(logical) if TRUE, returns the ENR for each permutation, for all networks. Warning: this is likely to be huge. Use this flag for debugging purposes only.
...	parameters for `countIntType_batch()`.

Details

Determines which networks are statistically enriched for interactions between the class of interest. The resulting `ENR` score and corresponding p-value serve as a filter to exclude random-like interaction networks before using feature selection. This filter is known to be important when patient networks are sparse and binary; e.g. networks based on shared overlap of CNV locations. If the filter is not applied, GeneMANIA WILL promote networks with slight bias towards (+,+) edges , even if these are small and random-like.

The measure of (+,+)-enrichment is defined as: ENR(network N) = ((num (+,+) edges) - (num other edges))/(num edges). A p-value for per-network ENR is obtained non-parametrically by measuring a null distribution for ENR following multiple permutations of case-control labels.

Value

(data.frame) networks stats from clique-filtering, one record per network

Examples

```
data(npheno)
netDir <- system.file("extdata","example_nets",package="netDx")
x <- enrichLabelNets(netDir,npheno,".",predClass="case",netGrep="txt$",
numReps=5)
print(x)
```

featScores — *Demo feature-level scores from running feature selection on two-class problem*

Description

List with one entry per patient label ("SURVIVEYES" and "SURVIVENO"). Each entry contains scores obtained through feature-selection acros 100 train/test splits. Scores range from 0 to 10. Scores in data.frame format, with rows corresponding to features and columns to a particular train/test split.

Usage

```
data(featScores)
```

Examples

```
data(featScores)
head(featScores)
```

fetchPathwayDefinitions — *fetch pathway definitions from downloads.baderlab.org*

Description

fetch pathway definitions from downloads.baderlab.org

Usage

```
fetchPathwayDefinitions(month = NULL, year = NULL, day = 1, verbose = FALSE)
```

Arguments

month	(numeric or char) month of pathway definition file. Can be numeric or text (e.g. "January","April"). If NULL, fails.
year	(numeric) year of pathway definition file. Must be in yyyy format (e.g. 2020). If NULL, fails.
day	(integer)
verbose	(logical) print messages

Details

Fetches genesets compiled from multiple curated pathway databases. Downloaded from: https://download.baderlab.org/EM_
The file contains pathways from HumanCyc, NetPath, Reactome, NCI Curated Pathways and mSigDB.
For details see Merico D, Isserlin R, Stueker O, Emili A and GD Bader. (2010). PLoS One. 5(11):e13984.

Value

(char) Path to local cached copy of GMT file or initial download is required

Examples

```
fetchPathwayDefinitions("October",2021)
fetchPathwayDefinitions("October",2021)
fetchPathwayDefinitions(month=10,year=2021)
```

---

genes *Table of gene definitions (small subsample of human genes)*

Description

data.frame object with columns of (gene) RefSeq ID (name), chromosome (chrom), strand, transcription start site (txStart), transcription end site (txEnd), and gene symbol (name2)

Usage

```
data(genes)
```

Examples

```
data(genes)
head(genes)
```

---

getCorrType *Counts the relative correlation of (+,+) and (+,-)(-,-) interactions*

Description

Counts the relative correlation of (+,+) and (+,-)(-,-) interactions

Usage

```
getCorrType(inFile, plusID, minusID)
```

Arguments

inFile	(character): path to interaction networks
plusID	(character) vector of + nodes
minusID	(character) vector of - nodes

Value

(numeric) mean edge weight for (+,+) and other edges

`getEMapInput` *write enrichment map for consensus nets*

Description

write enrichment map for consensus nets

Usage

```
getEMapInput(
  featScores,
  namedSets,
  netInfo,
  pctPass = 0.7,
  minScore = 1,
  maxScore = 10,
  trimFromName = c(".profile", "_cont"),
  verbose = FALSE
)
```

Arguments

`featScores`	(data.frame) network scores across rounds of cross validation. Rows are networks and columns are network name followed by scores for cross-validation rounds. Output of getFeatureScores()
`namedSets`	(list) list of nets and units (e.g.e pathway names and genes). Should only contain units profiled in this dataset
`netInfo`	(data.frame) Table of network name (netName) and type (netType). Type is used to assign shapes to nodes: clinical clinical rna GUANOSINE_NUCLEOTIDES__I_DE_NOVO__I__BIO rna RETINOL_BIOSYNTHESIS
`pctPass`	(numeric between 0 and 1) fraction of splits for which the highest score for the network is required, for that to be the network's maxScore
`minScore`	(integer) features with score below this cutoff are excluded from downstream analyses
`maxScore`	(integer) maximum possible score in one round of cross- validation. e.g. for 10-fold cross-validation, maxScore=10.
`trimFromName`	(char) strings to trim from name with sub()
`verbose`	(logical) print messages

Value

(list) Length two. 1) nodeAttrs: data.frame of node attributes 2) featureSets: key-value pairs of selected feature sets (e.g. if pathway features are used, keys are pathway names, and values are member genes).

Examples

```
inDir <- system.file("extdata","example_output",package="netDx")
outDir <- paste(tempdir(),'plots',sep='/')
if (!file.exists(outDir)) dir.create(outDir)
featScores <- getFeatureScores(inDir,predClasses=c('LumA','notLumA'))
gp <- names(featScores)[1]
pathwayList <- readPathways(fetchPathwayDefinitions("October",2020))
pathwayList <- pathwayList[seq_len(5)]
netInfoFile <- system.file("extdata","example_output/inputNets.txt",package="netDx")
netInfo <- read.delim(netInfoFile,sep='\t',h=FALSE,as.is=TRUE)
emap_input <- getEMapInput(featScores[[gp]],pathwayList,netInfo)
summary(emap_input)
```

---

getEMapInput_many *Wrapper to generate multiple EnrichmentMaps (perhaps one per class)*

Description

Wrapper to generate multiple EnrichmentMaps (perhaps one per class)

Usage

```
getEMapInput_many(featScores, namedSets_valid, netTypes, outDir, ...)
```

Arguments

`featScores`	(list) keys are classes, and values are data.frames of network scores across cross-validation (output of getFeatScores()).
`namedSets_valid`	(list) Grouped unit variables limited to the units contained in the dataset. e.g. keys are pathways and values are the genes measured in this dataset. e.g.: $‘MISSPLICED_GSK3BETA_MUTANTS_STABILIZE_BETA-CATENIN‘ [1] ’PPP2R5E’ ’PPP2CB’ ’APC’ ’AXIN1’ ’PPP2R1B’ ’PPP2R1A’ ’CSNK1A1’ [8] ’PPP2R5D’ ’PPP2R5C’ ’PPP2R5B’ ’PPP2R5A’ ’PPP2CA’ ’GSK3B’
`netTypes`	(data.frame) ’inputNets.txt’ file generated by NetDx. Dataframe has two columns, network type and network name. I.E: clinical clinical rna GUANOSINE_NUCLEOTIDES__I_DE_NOVO rna RETINOL_BIOSYNTHESIS
`outDir`	(char) path to output directory
`...`	parameters for getEMapInput()

Value

(list) of length g, where g is the number of groups in featScores. Values are lists, corresponding to the output of getEmapInput.R

Examples

```
data(featScores)

pathwayList <- readPathways(fetchPathwayDefinitions("October",2020))
pathwayList <- pathwayList[seq_len(5)]

netInfoFile <- system.file("extdata","example_output/inputNets.txt",package="netDx")
netTypes <- read.delim(netInfoFile,sep='\t',h=FALSE,as.is=TRUE)
outDir <- paste(tempdir(),'plots',sep='/')
if (!file.exists(outDir)) dir.create(outDir)
EMap_input <- getEMapInput_many(featScores,pathwayList,
     netTypes,outDir=outDir)
```

---

getEnr *Get ENR for all networks in a specified directory*

Description

Get ENR for all networks in a specified directory

Usage

```
getEnr(
  netDir,
  pheno_DF,
  predClass,
  netGrep = "_cont.txt$",
  enrType = "binary",
  ...
)
```

Arguments

netDir	(char) directory containing interaction networks
pheno_DF	(data.frame) table with patient ID and status. Must contain columns for Patient ID (named "ID") and class (named "STATUS"). Status should be a char; value of predictor class should be specified in predClass param; all other values are considered non-predictor class Rows with duplicate IDs will be excluded.
predClass	(char) value for patients in predictor class
netGrep	(char) pattern for grep-ing network text files, used in dir(pattern=..) argument
enrType	(char) how enrichment should be computed. Options are: 1) binary: Skew of number of (+,+) interactions relative to other interactions. Used when all edges in network are set to 1 (e.g. shared CNV overlap) 2) corr: 0.5*((mean weight of (+,+) edges)-(mean weight of other edges))
...	arguments for countIntType_batch

Details

For each network, compute the number of (+,+) and other (+,-),(-,+),(-,-) interactions. From this compute network ENR. The measure of (+,+)-enrichment is defined as: ENR(network N) = ((num (+,+) edges) - (num other edges))/(num edges). A network with only (+,+) interactions has an ENR=1 ; a network with no (+,+) interactions has an ENR=-1; a network with a balance of the two has ENR=0.

Value

(list): 1) plusID (char) vector of + nodes 2) minusID (char) vector of - nodes 3) orig_rat (numeric) ENR for data networks 4) fList (char) set of networks processed 5) orig (data.frame) output of countIntType_batch for input networks

Examples

```
d <- tempdir()
options(stringsAsFactors=FALSE)
pids <- paste("P",seq_len(5),sep="")
pheno <- data.frame(ID=pids,STATUS=c(rep("case",3),rep("control",2)))

# write PSN
m1 <- matrix(c("P1","P1","P2","P2","P3","P4",1,1,1),byrow=FALSE,ncol=3)
write.table(m1,file=paste(d,"net1.nettxt",sep=getFileSep()),sep="\t",
col.names=FALSE,row.names=FALSE,quote=FALSE)
m2 <- matrix(c("P3","P4",1),nrow=1)
write.table(m2,file=paste(d,"net2.nettxt",sep=getFileSep()),sep="\t",
col.names=FALSE,row.names=FALSE,quote=FALSE)

# compute enrichment
x <- countPatientsInNet(d,dir(d,pattern=c("net1.nettxt","net2.nettxt")), pids)
getEnr(d,pheno,"case","nettxt$")
```

---

getFeatureScores *Compile network scores into a matrix*

Description

Compile network scores into a matrix

Usage

```
getFeatureScores(inDir, predClasses, getFullCons = TRUE)
```

Arguments

inDir	(char/list) directory containing directories with all split info or list of all CV score files. if inDir is a single directory then the expected format for CV score files is <inDir>/rngX/predClassX/GM_results/predClassX_pathway_CV_score.txt'

	if inDir is a list, it should have one key per class. The value should be the corresponding set of filenames for pathway_CV_score.txt
predClasses	(char) possible STATUS for patients
getFullCons	(logical) if TRUE, does not remove rows with NA. Recommended only when the number of input features is extensively pruned by first-pass feature selection.

Details

Given network scores over a set of train/test splits, compiles these into a matrix for downstream analysis. See the section on 'Output Files'

Value

(list) one key per patient class. Value is matrix of network scores across all train/test splits. Each score is the output of the inner fold of CV.

Examples

```
inDir <- system.file("extdata","example_output",package="netDx")
netScores <- getFeatureScores(inDir, predClasses = c('LumA','notLumA'))
```

---

getFileSep *platform-specific file separator*

Description

Returns OS-specific file separator

Usage

```
getFileSep()
```

Value

(char) "\" if Windows, else "/"

Examples

```
getFileSep()
```

getGMjar_path *download and update GeneMANIA jar file*

Description

download and update GeneMANIA jar file

Usage

```
getGMjar_path(verbose = FALSE)
```

Arguments

verbose	(logical) print messages

Value

(char) Path to local cached copy of GeneMANIA jar file.. or initial download is required

Examples

```
getGMjar_path()
```

getNetConsensus *compile net score across a set of predictor results*

Description

compile net score across a set of predictor results

Usage

```
getNetConsensus(scorelist)
```

Arguments

scorelist	(list) key is dataset name, value is a data.frame containing PATHWAY_NAME and SCORE. This is the output of compileFeatureScores()

Details

used to compare how individual nets score for different predictor configurations

Value

(data.frame) Rownames are union of all nets in the input list. Columns show net scores for each key of the input list. Where a net is not found in a given list, it is assigned the value of NA

Examples

```
pathways <- paste("PATHWAY_",1:100,sep="")
highrisk <- list()
for (k in 1:10) {
highrisk[[k]] <- data.frame(PATHWAY_NAME=pathways,
SCORE=runif(length(pathways),min=0,max=10),
stringsAsFactors=FALSE);
}
names(highrisk) <- sprintf("Split%i",1:length(highrisk))
x <- getNetConsensus(highrisk)
```

`getOR` *Get relative proportion of patient classes that contribute to a set of networks*

Description

Get relative proportion of patient classes that contribute to a set of networks

Usage

```
getOR(pNetworks, pheno_DF, predClass, netFile, verbose = TRUE)
```

Arguments

`pNetworks`	(matrix) rows are patients, columns are network file filenames. a[i,j] = 1 if patient i has a structural variant in network j; else a[i,j] = 0
`pheno_DF`	(data.frame) Column "ID" has unique patient identifiers; column "STATUS" has patient class
`predClass`	(char) Class for which predictor is being built
`netFile`	(char) vector of networks of interest (e.g. those passing feature selection)
`verbose`	(logical) print messages

Details

Feature selected networks should have the property of being enriched in the class of interest; e.g. be enriched in 'case' relative to 'control'. When given a list of networks N, this method computes the number and proportion of patients that overlap N. A high relative fraction of the predicted class indicates successful feature selection. To create a ROC or precision-recall curve, several calls can be made to this function, one per cutoff.

Value

List. 1) stats: statistics on group overlap with , This is a 2xK matrix, where rows are classes (predClass,other), and columns are: total samples, samples overlapping nets, 2) relEnr: relative enrichment of `predClass` over other

Examples

```
d <- tempdir()
options(stringsAsFactors=FALSE)
pids <- paste("P",seq_len(5),sep="")
pheno <- data.frame(ID=pids,STATUS=c(rep("case",3),rep("control",2)))

# write PSN
m1 <- matrix(c("P1","P1","P2","P2","P3","P4",1,1,1),byrow=FALSE,ncol=3)
write.table(m1,file=paste(d,"net1.txt",sep=getFileSep()),sep="\t",
col.names=FALSE,row.names=FALSE,quote=FALSE)
m2 <- matrix(c("P3","P4",1),nrow=1)
write.table(m2,file=paste(d,"net2.txt",sep=getFileSep()),sep="\t",
col.names=FALSE,row.names=FALSE,quote=FALSE)

# compute enrichment
x <- countPatientsInNet(d,dir(d,pattern=c("net1.txt","net2.txt")), pids)
getOR(x,pheno,"case",colnames(x)) # should give large RelEnr
```

---

getPatientPredictions *Calculates patient-level classification accuracy across train/test splits*

Description

Calculates patient-level classification accuracy across train/test splits

Usage

```
getPatientPredictions(predFiles, pheno, plotAccuracy = FALSE)
```

Arguments

predFiles	(char) vector of paths to all test predictions (e.g. 100 files for a 100 train/test split design). Alternately, the user can also provide a single directory name, and allow the script to retrieve prediction files. Format is 'rootDir/rngX/predictionResults.txt'
pheno	(data.frame) ID=patient ID, STATUS=ground truth (known class label). This table is required to get the master list of all patients, as not every patient is classified in every split.
plotAccuracy	(logical) if TRUE, shows fraction of times patient is misclassified, using a dot plot

Details

Takes all the predictions across the different train/test splits, and for each patient, generates a score indicating how many times they were classified by netDx as belonging to each of the classes. The result is that we get a measure of individual classification accuracy across the different train/test splits.

Value

(list) of length 2. 1) (data.frame) rows are patients, (length(predFiles)+2) columns. Columns seq_len(length(predFiles)): Predicted labels for a given split (NA if patient was training sample for the split). Column (length(predFiles)+1): split, value is NA. Columns are : ID, REAL_STATUS, predStatus1,... predStatusN. Side effect of plotting a dot plot of and the value is '

Examples

```
inDir <- system.file("extdata","example_output",package="netDx")
data(pheno)
all_rngs <- list.dirs(inDir, recursive = FALSE)
all_pred_files <- unlist(lapply(all_rngs, function(x) {
    paste(x, 'predictionResults.txt',
sep = getFileSep())}))
pred_mat <- getPatientPredictions(all_pred_files, pheno)
```

---

getPatientRankings *Process GM PRANK files to get the ROC curve for the query*

Description

Process GM PRANK files to get the ROC curve for the query

Usage

```
getPatientRankings(pFile, pheno_DF, predClass, plotIt = FALSE, verbose = FALSE)
```

Arguments

pFile	(char) path to PRANK file
pheno_DF	(data.frame) patient IDs ('ID') and label('STATUS')
predClass	(character) class label for which predictor is built
plotIt	(logical) if TRUE plots ROC curve
verbose	(logical) print messages

Value

(list) 1) predLbl: GeneMANIA scores (predicted labels). Higher score for higher ranked patient. 2) realLbl: binary value indicating if patient label matches predictor label (real labels) 3) fullmat: pheno_DF merged with similarity scores ('similarityScore') and real label ('isPredClass') 4) roc: output of ROCRs performance(,'tpr','fpr') - ROC curve 5) auc: output of ROCRs auc() 6) precall: output of ROCRs performance(, 'prec','rec') 7) f: output of ROCRs performance(,'f') If < 2 patients in PRANK file, roc,auc, precall, f are all returned as NA.

Examples

```
data(pheno)
prankFile <- system.file("extdata",
paste("GM_PRANK","CV_1.query-results.report.txt.PRANK",sep=getFileSep()),
package="netDx")
x <- getPatientRankings(prankFile, pheno, 'LumA')
```

`getPerformance` *performance metrics for model*

Description

performance metrics for model

Usage

```
getPerformance(res, predClasses)
```

Arguments

`res`	(data.frame) result from predicting labels on held-out test set. output of predict() function. columns include ID, STATUS (ground truth) and PRED_CLASS (predicted label)
`predClasses`	(character) patient labels used by classifier

Value

(list) 1) rocCurve: ROCR performance object for ROC curve 2) prCurve: ROCR performance object for PR curve 3) auroc: Area under ROC curve 4) aupr: Area under PR curve 5) accuracy: Accuracy

---

`getPSN` *get the integrated patient similarity network made of selected features*

Description

get the integrated patient similarity network made of selected features

Usage

```
getPSN(
  dat,
  groupList,
  makeNetFunc = NULL,
  sims = NULL,
  selectedFeatures,
  plotCytoscape = FALSE,
  aggFun = "MEAN",
  prune_pctX = 0.3,
  prune_useTop = TRUE,
  numCores = 1L,
  calcShortestPath = FALSE
)
```

Arguments

dat	(MultiAssayExperiment) input data
groupList	(list) feature groups, identical to groupList provided for buildPredictor()
makeNetFunc	(function) Function used to create patient similarity networks. Identical to makeNets provided to buildPredictor()
sims	(list) rules for creating PSN. Preferred over makeNetFunc. See buildPredictor() for details.
selectedFeatures	(list) selected features for each class (key of list). This object is returned as part of a call to getResults(), after running buildPredictor().
plotCytoscape	(logical) If TRUE, plots network in Cytoscape. Requires Cytoscape software to be installed and running on the computer when the function call is being made.
aggFun	(char) function to aggregate edges from different PSN (e.g. mean)
prune_pctX	(numeric between 0 and 1) fraction of most/least edges to keep when pruning the integrated PSN for visualization. Must be used in conjunction with useTop=TRUE/FALSE e.g. Setting pctX=0.2 and useTop=TRUE will keep 20% top edges
prune_useTop	(logical) when pruning integrated PSN for visualization, determines whether to keep strongest edges (useTop=TRUE) or weakest edges (useTop=FALSE)
numCores	(integer) number of cores for parallel processing
calcShortestPath	(logical) if TRUE, computes weighted shortest path Unless you plan to analyse these separately from looking at the shortest path violin plots or integrated PSN in Cytoscape, probably good to set to FALSE.

Details

An integrated patient similarity network can be built using combined top features for each patient class. Such a network is created by taking the union of selected features for all patient labels, and aggregating pairwise edges for all of them using a user-specified function (aggFun). The network

is then pruned prior to visualization, using a user-specified fraction of strongest edges (prune_pctX, prune_useTop). In addition, the user may quantify the distance between patients of the same class, relative to those of other classes, using Dijkstra distance (calcShortestPath flag).

Value

(list) information about the integrated network similarity network 2) patientDistNetwork_pruned (matrix) the network plotted in Cytoscape. Also note that this is a dissimilarity network, so that more similar nodes have smaller edge weights 3) colLegend (data.frame): legend for the patient network plotted in Cytoscape. Columns are node labels (STATUS) and colours (colour) 6) outDir (char) value of outDir parameter

---

getRegionOL *Returns overlapping named ranges for input ranges*

Description

Returns overlapping named ranges for input ranges

Usage

```
getRegionOL(gr, rngList)
```

Arguments

gr	(GRanges) query ranges
rngList	(list) keys are names, and values are GRanges, each range of which has a name (in 'name' column). Note: It is faster to provide a list of length 1 ; if the list is long, combining into a single GRanges object could prove slow.

Details

Given a set of query GRanges, and a subject list-of-GRanges, updates the query with a column 'LOCUS_NAMES' containing the names of ranges overlapped by the query. One application is to map structural variants, such as CNVs, to genes in pathways of interest. In this scenario gr would contain the patient CNVs, and rngList would be a list of GenomicRanges objects, one per cellular pathway.

Value

(GRanges) query ranges with the added column 'LOCUS_NAMES'. Where a range overlaps with multiple loci, the names are reported as a comma-separated vector

Examples

```
data(cnv_GR,pathway_GR)
x <- getRegionOL(cnv_GR,pathway_GR)
```

getResults *Compiles performance and selected features for a trained model.*

Description

Compiles performance and selected features for a trained model.

Usage

```
getResults(res, status, featureSelCutoff = 1L, featureSelPct = 0)
```

Arguments

res	(list) output of buildPredictor() function
status	(character) unique patient labels used by the classifier, found in colData()$STATUS
featureSelCutoff	(integer) cutoff score for feature selection. A feature must have minimum of this score for specified fraction of splits (see featureSelPct) to pass.
featureSelPct	(numeric between 0 and 1) cutoff percent for feature selection. A feature must have minimum score of featureSelCutoff for featureSelPct of train/test splits to pass.

Details

This function is run after training a model using buildPredictor(). It takes patient input data, model output, and returns performance and selected features.

Value

list of results. - selectedFeatures (list of character vectors): list, one per class - performance (list of mixed datatypes) including mean accuracy (meanAccuracy), split-level accuracy (splitAccuracy), split-level AUROC (auroc), split-level AUPR (splitAUR) Side effect of plotting ROC curve if binary classifier

Examples

```
data(toymodel) # load example results from binary breast classification
patlabels <- names(toymodel$Split1$featureSelected)
getResults(toymodel,patlabels,2,0.5)
```

getSimilarity *Measures of patient similarity*

Description

Measures of patient similarity

Usage

```
getSimilarity(x, type = "pearson", customFunc, ...)
```

Arguments

x	(matrix) matrix for which pairwise patient similarity is to be computed. Expects one column per patient, and one measurement per row.
type	(character) name of similarity measure. Currently supports Pearson correlation ('pearson') or a custom measure ('custom')
customFunc	(function) custom similarity function. Only used when type='custom'. The function takes x as first argument and can take additional argument. It should return a symmetric matrix of pairwise patient similarities.
...	parameter for customFunc

Value

symmetric matrix of size N, where N is number of samples

Examples

```
data(xpr)
x <- getSimilarity(xpr) # similarity by Pearson corr
mySim <- function(x) cor(x,method='kendall')
x <- getSimilarity(xpr,customFunc=mySim) # custom similarity
```

makeInputForEnrichmentMap *Wrapper to create input files for Enrichment Map*

Description

Wrapper to create input files for Enrichment Map

Usage

```
makeInputForEnrichmentMap(
  model,
  results,
  pathwayList,
  EMapMinScore = 0L,
  EMapMaxScore = 1L,
  EMapPctPass = 0.5,
  outDir
)
```

Arguments

model	(list) Output of training model, generated by running buildPredictor()
results	(list) Model results. output of getResults()
pathwayList	(list) output of readPathwayFile() used to make pathway-level feat ures for predictor
EMapMinScore	(integer) minimum score for Enrichment Map
EMapMaxScore	(integer) maximum score for Enrichment Map
EMapPctPass	(numeric between 0 and 1) percent of splits for which feature must have score in range [EMapMinScore,EMapMaxScore] to be included for EnrichmentMap visualization
outDir	(char) directory where files should be written

Details

An Enrichment Map is a network-based visualization of top-scoring pathway features and themes. It is generated in Cytoscape. This script generates the input files needed for Cytoscape to create an Enrichment Map visualization.

Value

(list) 1) GMTfiles (char): GMT files used to create EnrichmentMap in Cytoscape. 2) NodeStyles (char): .txt files used to assign node attributes in Cytoscape. Importantly, attributes include node fill, which indicates the highest consistent score for a given feature.

---

makePSN_NamedMatrix *Create patient networks from full matrix of named measurements*

Description

Create patient networks from full matrix of named measurements

Usage

```
makePSN_NamedMatrix(
  xpr,
  nm,
  namedSets,
  outDir = tempdir(),
  simMetric = "pearson",
  verbose = TRUE,
  numCores = 1L,
  writeProfiles = TRUE,
  sparsify = FALSE,
  useSparsify2 = FALSE,
  cutoff = 0.3,
  sparsify_edgeMax = Inf,
  sparsify_maxInt = 50,
  minMembers = 1L,
  runSerially = FALSE,
  ...
)
```

Arguments

xpr	(matrix) rows are measurements, columns are samples. Columns must be named (patient ID)
nm	(character) names for measurements corresponding to row order of xpr. Must match the names in the named sets specified in nameSets
namedSets	(list) sets of names to be grouped together. keys are set names, and networks will be named as these. values are character vectors corresponding to groups of names (matching those in nm) that are input to network generation
outDir	(char) path to directory where networks are written. If missing, is set to tempdir()
simMetric	(char) measure of similarity. See getSimilarity() for details. If writeProfiles is set to TRUE, must be one of pearson (Pearson correlation) or MI (correlation by mutual information).
verbose	(logical) print detailed messages
numCores	(integer) number of cores for parallel network generation
writeProfiles	(logical) use GeneMANIA's ProfileToNetworkDriver to create interaction networks. If TRUE, this function writes subsets of the original data corresponding to networks to file (profiles). If FALSE, uses getSimilarity() and writes interaction networks.
sparsify	(logical). If TRUE, sparsifies patient similarity network. See useSparsify2, sparsify_edgeMax and sparsify_maxInt
useSparsify2	(logical). Cleaner sparsification routine. If FALSE, uses new matrix-based sparsify3
cutoff	(numeric) patients with similarity smaller than this value are not included in the corresponding interaction network

sparsify_edgeMax
: (numeric) Max number of edges to include in the final network

sparsify_maxInt
: (numeric) Max num edges per node in sparsified network.

minMembers
: (integer) min number of measures in a network for the network to be included. Useful when similarity measures require a minimum number of measures to be meaningful (e.g. minimum of 6 for Pearson correlation)

runSerially
: (logical) set to TRUE to create nets serially, rather than in parallel

...
: passed to getSimilarity()

Details

Creates patient similarity networks when full matrices of data are provided (e.g. gene expression, questionnaire results). To generate networks from sparse data such as CNVs or indels, use makePSN_RangeSets instead. The rows of the data matrix (xpr) must be named (nm); one network is create for each named set (namedSets). There are two options for the way in which networks are created, depending on the value of writeProfiles. 1. writeProfiles=TRUE: GeneMANIA is used to generate interaction networks and sparsify networks. This only works if the desired measure of similarity is network-level Pearson correlation; an example is networks at the level of pathways. In this case, the user does not explicitly specify a similarity measure and simMetric is ignored. 2. writeProfiles=FALSE: GeneMANIA is not used to generate interaction networks. Rather, netDx uses simMetric to create interaction networks. Networks can be sparsified by excluding weak connections (cutoff).

Value

(char) Basename of files to which networks are written. Side effect of writing interaction networks in outDir

Examples

```
data(xpr,pheno,pathwayList);
# you may get a warning message that the output directory already
# exists; ignore it
out <- makePSN_NamedMatrix(xpr,rownames(xpr),pathwayList,
'.',writeProfiles=TRUE)
```

---

makePSN_RangeSets *Create patient similarity interaction networks based on range sets*

Description

Create patient similarity interaction networks based on range sets

Usage

```
makePSN_RangeSets(
  gr,
  rangeSet,
  netDir = tempdir(),
  simMetric = "coincide",
  quorum = 2L,
  verbose = TRUE,
  numCores = 1L
)
```

Arguments

gr	(GRanges) patient ranges. Metadata should contain: ID: (char) unique patient ID LOCUS_NAME: (comma-separated char) named ranges overlapped
rangeSet	(list) list of GRanges, one entry per range set. Key is the name of the range set, and value is a GRanges object with corresponding ranges
netDir	(char) path to directory where networks should be written
simMetric	(char) Similarity metric. Currently only 'coincide' is supported; two patients share an edge if they overlap elements in the the same gene set. E.g. Two patients with CNVs that overlap different genes of the same pathway would be related, but patients overlapping genes that don't share a pathway (or, more accurately, a named-set grouping) would not be related. The edge weight is therefore binary.
quorum	(integer) minimum number of patients in a network for the network to be constructed
verbose	(logical) print detailed messages
numCores	(integer) num cores for parallel processing

Details

Creates patient similarity networks when data consist of genomic events associated with patients. Examples include CNV or indel data for patients. To generate networks from full matrices such gene expression data, use `makePSN_NamedMatrix` instead. Genomic ranges corresponding to events in patients (gr) should be named. One network is created per named range set (rangeSet). Each set reflects a group of related loci ; for example, genomic ranges associated with genes in the same cellular pathway. Currently, the only similarity measure supported is binary; two patients are related in a network N if they both overlap elements of set N.

Value

Vector of network filenames

Examples

```
data(pathway_GR,cnv_GR)
### # example commented out to avoid build errors because of parallel
```

```
### # execution. Uncomment to run.
### netList <- makePSN_RangeSets(cnv_GR,pathway_GR,'.')
```

`makeQueries` *Randomly select patients for queries for feature selection*

Description

Randomly select patients for queries for feature selection

Usage

```
makeQueries(incPat, featScoreMax = 10L, verbose = TRUE)
```

Arguments

`incPat`	(char) vector of patient IDs to be included in query
`featScoreMax`	(integer) Number of times to run query, usually equal to the max score for features in the design (e.g. if featScoreMax=10, then this value is 10).
`verbose`	(logical) print messages

Value

(list) of length `featScoreMax`, containing names of patients in query file for each fold

Examples

```
data(pheno)
x <- makeQueries(pheno$ID)
```

`makeSymmetric` *Convert a network in source-target-weight format to symmetric matrix*

Description

Convert a network in source-target-weight format to symmetric matrix

Usage

```
makeSymmetric(x, verbose = FALSE)
```

Arguments

`x`	(data.frame) three columns, with source node, target node, and edge weight. Entries must include universe of nodes; those with missing edges must be included as having edge weight NA
`verbose`	(logical) print messages

Details

A common format for network representation is to use a three column table listing source node, target node, and weight. This is the format netDx uses for network integration and visualization in Cytoscape. However, some functionality requires a square symmetric adjacency matrix. This function takes as input the three-column format and converts to the adjacency matrix. NOTE: Symmetric attribute is assumed, and the function automatically sets a[i,j] = a[j,i]. Diagonal is assumed to have value of 1.0. Finally missing edges will be assigned NA values.

Value

(matrix) symmetric adjacency matrix

Examples

```
src <- c("A","B"); tgt <- c("C","C")
cur <- data.frame(source=src,target=tgt,weight=c(0.3,0.8))
makeSymmetric(cur)
```

---

mapNamedRangesToSets *Map named ranges to corresponding set of named ranges*

Description

Map named ranges to corresponding set of named ranges

Usage

```
mapNamedRangesToSets(gr, rangeList, verbose = FALSE)
```

Arguments

gr	(GRanges) named ranged to be grouped
rangeList	(list) sets of range names
verbose	(logical) print detailed messages

Details

Example application is when we have named ranges each corresponding to genes or regulatory elements, and we wish to group these ranges based on metabolic pathway.

Value

RangeList. keys are names of `rangeList`, values are GRanges

Examples

```
data(genes,pathwayList);
gene_GR<-GenomicRanges::GRanges(genes$chrom,
  IRanges::IRanges(genes$txStart,genes$txEnd),
name=genes$name2)
path_GRList <- mapNamedRangesToSets(gene_GR,pathwayList)
```

`matrix_getIJ` *Converts matrix index (1 to m*n) to row (m) and column (n) number*

Description

Converts matrix index (1 to m*n) to row (m) and column (n) number

Usage

```
matrix_getIJ(dimMat, idx)
```

Arguments

`dimMat`	(integer vector of length 2) output of `dim()` for matrix in question
`idx`	(integer vector of length n) matrix indices

Value

(matrix) n-by-2, first column has row indices ; second column has col indices

---

`MB.pheno` *Sample metadata table for medulloblastoma dataset.*

Description

data.frame with patient ID and tumour subtype (STATUS)

Usage

```
data(MB.pheno)
```

Source

Northcott et al. (2011). J Clin Oncol. 29 (11):1408.

References

Northcott et al. (2011). J Clin Oncol. 29 (11):1408.

Examples

```
data(MB.pheno)
head(MB.pheno)
```

`modelres` *Sample output of getResults()*

Description

Output of getResults() generated by running toymodel. toymodel is itself the output of buildPredictor() run on a simple dataset for binary breast tumour classification using two genomic data sources. BRCA data were downloaded using curatedTCGAData for mRNA and miRNA expression. buildPredictor()] was run by scoring features out of 2, with selected features passing 1 out of 2. Tumours were labelled either "Luminal.A" or "other". See details of getResults() for output format.

Usage

```
data(modelres)
```

Examples

```
data(modelres)
head(modelres)
```

`moveInteractionNets` *moves interaction networks when compiling database for sparse genetic workflow*

Description

moves interaction networks when compiling database for sparse genetic workflow

Usage

```
moveInteractionNets(netDir, outDir, pheno, fileSfx = "_cont.txt")
```

Arguments

`netDir`	(char) source directory
`outDir`	(char) target directory
`pheno`	(data.frame) contains patient ID and STATUS
`fileSfx`	(char) suffix to strip from network file names before registering in metadata tables

Value

No value. Side effect of moving interaction nets to target directory and creating network-related metadata files used to compile feature database

normDiff *Similarity metric of normalized difference*

Description

Similarity metric of normalized difference

Usage

```
normDiff(x)
```

Arguments

x	(numeric) vector of values, one per patient (e.g. ages)

Details

Similarity metric used when data for a network consists of exactly 1 continuous variable (e.g. a network based only on 'age'). When number of variables is 2-5, use avgNormDiff() which takes the average of normalized difference for individual variables

Value

symmetric matrix of size ncol(dat) (number of patients) containing pairwise patient similarities

Examples

```
sim <- normDiff(rnorm(10))
```

npheno *Toy sample metadata table*

Description

data.frame with patient ID ("ID") and label ("STATUS"). 100 "cases" and 100 "controls"

Usage

```
data(npheno)
```

Examples

```
data(npheno)
head(npheno)
```

pathwayList *Sample list of pathways*

Description

List where keys are pathway names and values are character vectors comprising of member genes for corresponding pathways

Usage

```
data(pathwayList)
```

Examples

```
data(pathwayList)
head(pathwayList)
```

pathway_GR *List of genomic ranges mapped to pathways*

Description

List object. Keys are pathway names, values are GRanges objects with coordinates of corresponding genes. Small subset of pathways sufficient for package examples.

Usage

```
data(pathway_GR)
```

Examples

```
data(pathway_GR)
head(pathway_GR)
```

perfCalc *Computes variety of predictor evaluation measures based on the confusion matrix*

Description

Computes variety of predictor evaluation measures based on the confusion matrix

Usage

```
perfCalc(dat)
```

Arguments

dat	(data.frame): 5 columns: score, tp, fp, tn, fn. One row per cutoff score for feature selection

Value

(list) stats (data.frame): score, f1, ppv, precision and recall. One row per cutoff for feature selection auc (numeric between 0 and 1): AUC of overall ROC curve prauc (numeric between 0 and 1): AUC of overall precision-recall curve

Examples

```
data(confmat)
x <- perfCalc(confmat)
```

pheno *Sample metadata table*

Description

data.frame with patient ID (ID), sample type (Type), tumour subtype (STATUS). From TCGA 2012 breast cancer paper (see reference).

Usage

```
data(pheno)
```

Source

The Cancer Genome Atlas. (2012). Nature 490:61-70.

References

The Cancer Genome Atlas. (2012). Nature 490:61-70.

Examples

```
data(pheno)
head(pheno)
```

---

pheno_full *Subsample of TCGA breast cancer data used for netDx function examples*

Description

Patient ID and tumour status in "pheno", subsample of gene expression in "xpr" and CNV data in "cnv_GR"

Usage

```
data(pheno_full)
```

Source

The Cancer Genome Atlas. (2012). Nature 490:61-70.

References

The Cancer Genome Atlas. (2012). Nature 490:61-70.

Examples

```
data(pheno_full)
head(pheno_full)
```

---

plotEmap *Create EnrichmentMap in Cytoscape to visualize predictive pathways*

Description

Create a network where nodes are predictive pathways passing certain cutoff and edges indicate similarity in gene-sets. Pathways are then clustered to identify themes of predictive pathways. Generates one such network for each patient label.

Usage

```
plotEmap(
  gmtFile,
  nodeAttrFile,
  netName = "generic",
  scoreCol = "maxScore",
  minScore = 1,
  maxScore = 10,
  nodeFillStops = c(7, 9),
  colorScheme = "cont_heatmap",
  imageFormat = "png",
  verbose = FALSE,
  createStyle = TRUE,
  groupClusters = FALSE,
  hideNodeLabels = FALSE
)
```

Arguments

gmtFile	(character) file path to GMT file (generated by getEMapInput()). NOTE: This needs to be the absolute path name
nodeAttrFile	(list) file path to nodeAttr.txt file (generated by getEMapInput())
netName	(character) name for network in Cytoscape. Using the patient class name is a good idea. (e.g. SURVIVE_YES and SURVIVE_NO).
scoreCol	(character) column of nodeAttrFile with the node score
minScore	(integer) minimum score of node to show
maxScore	(integer) maximum score of node to show
nodeFillStops	(integer) vector of length 2. Contains score values that indicate "good signal" and "best signal". Nodes with values above "good signal" are coloured orange, and those with "best signal" are coloured red.
colorScheme	(character) colour scheme for nodes. 'cont_heatmap' sets a discrete map ranging from yellow to red for increasing scores. 'netDx_ms' is the colour scheme used in the netDx methods paper. This map is (<=6: white; 7-9: orange; 10: red)
imageFormat	(character) one of PNG, PDF, SVG, or JPEG
verbose	(logical) print messages
createStyle	(logical) if generating more than one EMap, set to TRUE for first one and to FALSE for subsequent. Due to limitation in current version of RCy3
groupClusters	(logical) if TRUE, redraws network with thematic clusters lined up in rows. This setting is useful if setting this flag to FALSE results in a cluttered network. However, applying this layout will organize nodes in each cluster into circles, which loses the c topology.
hideNodeLabels	(logical) if TRUE hides the node label in the EnrichmentMap. Cluster labels remain visible.

Value

No value. Side effect of plotting the EnrichmentMap in an open session of Cytoscape.

Examples

```
#refer to getEMapInput_many.R for working getEMapInput_many() example
data(featScores)
pathwayList <- readPathways(fetchPathwayDefinitions("October",2020))
pathwayList <- pathwayList[seq_len(5)]
netInfoFile <- system.file("extdata",
paste("example_output","inputNets.txt",sep=getFileSep()),
package="netDx")
netTypes <- read.delim(netInfoFile,sep='\t',header=FALSE,as.is=TRUE)
outDir <- paste(tempdir(),'plots',sep=getFileSep())
if (!file.exists(outDir)) dir.create(outDir)
EMap_input <- getEMapInput_many(featScores,pathwayList,
     netTypes,outDir=outDir)
outDir <- paste(getwd(),'plots',sep=getFileSep())
if (!file.exists(outDir)) dir.create(outDir)
gmtFile <- EMap_input[[1]][1]
nodeAttrFile <- EMap_input[[1]][2]

# not run because requires Cytoscape to be installed and open
# plotEmap(gmtFile = gmtFile, nodeAttrFile = nodeAttrFile,
#\t\tnetName='HighRisk')
```

---

plotIntegratedPatientNetwork

Visualize integrated patient similarity network based on selected features

---

Description

Visualize integrated patient similarity network based on selected features

Usage

```
plotIntegratedPatientNetwork(
  dataList,
  groupList,
  makeNetFunc = NULL,
  sims = NULL,
  setName = "predictor",
  prune_pctX = 0.05,
  prune_useTop = TRUE,
  aggFun = "MAX",
  calcShortestPath = FALSE,
  showStats = FALSE,
```

```
  outDir = tempdir(),
  numCores = 1L,
  nodeSize = 50L,
  edgeTransparency = 40L,
  nodeTransparency = 155L,
  plotCytoscape = FALSE,
  verbose = FALSE
)
```

Arguments

dataList	(MultiAssayExperiment) patient data & labels used as input
groupList	(list) features to use to create integrated patient network. Identical in structure to groupList in buildPredictor() method. This is a list of lists, where the outer list corresponds to assay (e.g. mRNA, clinical) and inner list to features to generate from that datatype.
makeNetFunc	(function) function to create features
sims	(list) rules for creating PSN. Preferred over makeNetFunc
setName	(char) name to assign the network in Cytoscape
prune_pctX	(numeric between 0 and 1) fraction of most/least edges to keep when pruning the integrated PSN for visualization. Must be used in conjunction with useTop=TRUE/FALSE e.g. Setting pctX=0.2 and useTop=TRUE will keep 20% top edges
prune_useTop	(logical) when pruning integrated PSN for visualization, determines whether to keep strongest edges (useTop=TRUE) or weakest edges (useTop=FALSE)
aggFun	(char) function to aggregate edges from different PSN
calcShortestPath	(logical) if TRUE, computes weighted shortest path Unless you plan to analyse these separately from looking at the shortest path violin plots or integrated PSN in Cytoscape, probably good to set to FALSE.
showStats	(logical) if FALSE, suppresses shortest path-related stats, such as one-sided WMW test for testing shorter intra-class distances
outDir	(char) path to directory for intermediate files. Useful for debugging.
numCores	(integer) number of cores for parallel processing
nodeSize	(integer) size of nodes in Cytoscape
edgeTransparency	(integer) Edge transparency. Value between 0 and 255, with higher numbers leading to more opacity.
nodeTransparency	(integer) Node transparency. Value between 0 and 255, with higher numbers leading to more opacity.
plotCytoscape	(logical) If TRUE, plots network in Cytoscape. Requires Cytoscape software to be installed and running on the computer when the function call is being made.
verbose	(logical) print detailed messages

Details

Generates a Cytoscape network where nodes are patients and edges are weighted by aggregate pairwise patient similarity. Integrated PSN plotting is intended to run after feature selection, which identifies the subset of input networks predictive for each class of interest. The method of generating the network is as follows: All networks feature-selected in either patient ategory are concatenated; where a network is feature-selected in both categories, it is included once. The similarity between two patients in the integrated network is the mean of corresponding pairwise similarities. Dissimilarity is defined as 1-similarity, and Dijkstra distances are computed on this resulting network. For visualization, only edges representing the top fraction of distances (strongest edge weights) are included.

Value

(list) information about the integrated network similarity network 2) patientDistNetwork_pruned (matrix) the network plotted in Cytoscape. Also note that this is a dissimilarity network, so that more similar nodes have smaller edge weights 3) colLegend (data.frame): legend for the patient network plotted in Cytoscape. Columns are node labels (STATUS) and colours (colour) 6) outDir (char) value of outDir parameter

---

plotPerf *Plots various measures of predictor performance for binary classifiers*

Description

Plots various measures of predictor performance for binary classifiers

Usage

```
plotPerf(resList = NULL, inFiles, predClasses, plotSEM = FALSE)
```

Arguments

resList	(list) list of prediction results. If provided, the method will ignore inDir
inFiles	(char) path to predictionResults.txt files. A vector, each with absolute paths to predictionResults.txt
predClasses	(char) vector of class names.
plotSEM	(logical) metric for error bars. If set to TRUE, plots SEM; else plots SD.

Details

Plots individual and average ROC/PR curves. mean+/-SEM performance for a predictor run using nested cross-validation or a similar repeated design. predictionResults.txt contains a (data.frame)

Value

(list) each key corresponds to an input file in inDir. Value is a list with: 1) stats: 'stats' component of perfCalc 2) rocCurve: ROCR performance object for ROC curve 3) prCurve: ROCR performance object for PR curve 4) auroc: Area under ROC curve 5) aupr: Area under PR curve 6) accuracy: Accuracy

Side effect of plotting in a 2x2 format: 1) mean+/-SEM or (mean+/-SD) AUROC 2) mean+/-SEM or (mean+/-SD) AUPR 3) ROC curve for all runs plus average 4) PR curve for all runs plus average

Examples

```
inDir <- system.file("extdata","example_output",package='netDx')
inFiles <- paste(rep(inDir,3), sprintf("rng%i",seq_len(3)),"predictionResults.txt",
 sep=getFileSep())
resList <- list()
for (k in seq_len(length(inFiles))) {
resList[[k]] <- read.delim(inFiles[k],sep="\t",header=TRUE,as.is=TRUE)
}
plotPerf(resList, predClasses = c('LumA','notLumA'))
```

---

plotPerf_multi *Plots a set of ROC/PR curves with average.*

Description

Plots a set of ROC/PR curves with average.

Usage

```
plotPerf_multi(
  inList,
  plotTitle = "performance",
  plotType = "ROC",
  xlab = "TPR",
  ylab = "FPR",
  meanCol = "darkblue",
  xlim = c(0, 1),
  ylim = c(0, 1)
)
```

Arguments

inList	(list or ROCR::performance object) ROCR::performance objects, one per iteration
plotTitle	(numeric) plot title
plotType	(char) one of ROC | PR | custom. Affects x/y labels
xlab	(char) x-axis label

ylab	(char) y-axis label
meanCol	(char) colour for mean trendline
xlim	(numeric) min/max extent for x-axis
ylim	(numeric) min/max extent for y-axis

Details

Plots average curves with individual curves imposed.

Value

No value. Side effect of plotting ROC and PR curves

Examples

```
inDir <- system.file("extdata","example_output",package="netDx")
all_rng <- list.files(path = inDir, pattern = 'rng.')
fList <- paste(inDir,all_rng,'predictionResults.txt',sep=getFileSep())
rocList <- list()
for (k in seq_len(length(fList))) {
  dat <- read.delim(fList[1],sep='\t',header=TRUE,as.is=TRUE)
  predClasses <- c('LumA', 'notLumA')
  pred_col1 <- sprintf('%s_SCORE',predClasses[1])
  pred_col2 <- sprintf('%s_SCORE',predClasses[2])
  idx1 <- which(colnames(dat) == pred_col1)
  idx2 <- which(colnames(dat) == pred_col2)
 pred <- ROCR::prediction(dat[,idx1]-dat[,idx2],
dat$STATUS==predClasses[1])
 rocList[[k]] <- ROCR::performance(pred,'tpr','fpr')
}
plotPerf_multi(rocList,'ROC')
```

---

predict *predict patient labels*

Description

Once a model is trained, this function is used to classify new patients using selected features

Usage

```
predict(
  trainMAE,
  testMAE,
  groupList,
  selectedFeatures,
  makeNetFunc = NULL,
  sims = NULL,
```

```
  outDir,
  verbose = FALSE,
  numCores = 1L,
  JavaMemory = 4L,
  debugMode = FALSE
)
```

Arguments

trainMAE	(MultiAssayExperiment) patient data for training samples. Same as provided to buildPredictor()
testMAE	(MultiAssayExperiment) new patient dataset for testing model. Assays must be the same as for trainMAE.
groupList	(list) list of features used to train the model. Keys are data types, and values are lists for groupings within those datatypes. e.g. keys could include 'clinical','rna','methylation', and values within 'rna' could include pathway names 'cell cycle', 'DNA repair', etc., selectedFeatures will be used to subset
selectedFeatures	(list) selected features to be used in the predictive model. keys are patient labels (e.g. "responder/nonresponder"), and values are feature names identified by running buildPredictor(). Feature names must correspond to names of groupList, from which they will be subset.
makeNetFunc	(function) function to create PSN features from patient data. See makeNetFunc in buildPredictor() for details
sims	(list) rules for creating PSN. Preferred over makeNetFunc.
outDir	(char) directory for results
verbose	(logical) print messages
numCores	(integer) number of CPU cores for parallel processing
JavaMemory	(integer) memory in (Gb) used for each fold of CV
debugMode	(logical) Set to TRUE for detailed messages. Used for debugging.

Value

(data.frame) predicted patient similarities and labels columns are: 1) ID, 2) STATUS (ground truth), 3) <label>_SCORE: similarity score for the corresponding label, 4) PRED_CLASS: predicted class

---

predictPatientLabels *assign patient class when ranked by multiple GM predictors*

Description

assign patient class when ranked by multiple GM predictors

Usage

```
predictPatientLabels(resSet, verbose = TRUE)
```

Arguments

resSet	(list) output of getPatientRankings, each key for a different predictor. names(resSet) contain predictor label
verbose	(logical) print detailed messages

Value

data.frame: ID, similarityScore, PRED_CLASS

Examples

```
data(predRes); predClass <- predictPatientLabels(predRes)
```

---

predRes *Example output of getPatientRankings, used to call labels for test patients.*

Description

List of lists. First level is a list of size 4, with one key entry for each tumour type in example medulloblastoma dataset (WNT,SHH,Group3,Group4). Each list in the second level is of length 8, with structure corresponding to the output of getPatientRankings().

Usage

```
data(predRes)
```

Examples

```
data(predRes)
summary(predRes)
summary(predRes[[1]])
```

pruneNet *Prune network by retaining strongest edges*

Description

Prune network by retaining strongest edges

Usage

```
pruneNet(net, vertices, pctX = 0.1, useTop = TRUE)
```

Arguments

net	(data.frame) Network to prune. Columns are: source,target,weight
vertices	(char) node names. Should match those in net[,1:2]
pctX	(numeric 0 to 1) Fraction of top/bottom edges to retain
useTop	(logical) if TRUE prunes to top pctX edges; else prunes to bottom pctX edges

Value

(data.frame) pruned network. Three columns: AliasA, AliasB, and weight

pruneNets *Prune interaction networks to keep only the networks and patients requested*

Description

Prune interaction networks to keep only the networks and patients requested

Usage

```
pruneNets(
  oldDir,
  newDir = tempdir(),
  filterNets = "*",
  filterIDs = "*",
  netSfx = "_cont.txt$",
  verbose = TRUE
)
```

Arguments

oldDir	(char) path to directory with original networks
newDir	(char) path to output directory for pruned networks
filterNets	(char) vector of networks to include. These should match filenames in netDir. Value of '*' results in pruning all networks
filterIDs	(char) patients to include in pruned networks. These should match nodes in the input interaction networks
netSfx	(char) suffix for network file names. Only used if filterNets='*'.
verbose	(logical) print messages

Details

This function is crucial for patient data that is highly sparse; examples include patient CNVs indels, as opposed to full matrix measures (gene expression, questionnaire data). Each step where the pool of patients is subset - e.g. limiting feature selection only to patients in training set - changes the set of networks that are eligible. Some networks may only contain test patients, while others may contain a single edge between a training and a test patient. Upon subsetting, such networks are no longer eligible for downstream use, such as feature selection. This function rewrites those subnetworks of the original networks that consist of eligible patients.

Value

(no value). Side effect of writing pruned network files to newDir

Examples

```
data(npheno)
netDir <- system.file("extdata","example_nets",package='netDx')
pruneNets(netDir,tempdir(),filterIDs=npheno[seq_len(10),],
 netSfx='txt$')
```

---

pruneNet_pctX *Prune network by retaining strongest edges*

Description

Prune network by retaining strongest edges

Usage

```
pruneNet_pctX(net, vertices, pctX = 0.1, useTop = TRUE)
```

Arguments

net	(data.frame) Network to prune. Columns are: source,target,weight
vertices	(char) node names. Should match those in net[,1:2]
pctX	(numeric 0 to 1) Fraction of top/bottom edges to retain
useTop	(logical) if TRUE prunes to top pctX edges; else prunes to bottom pctX edges

Value

(data.frame) pruned network. Three columns: AliasA, AliasB, and weight

---

psn__builtIn *make PSN for built-in similarity functions*

Description

make PSN for built-in similarity functions

Usage

```
psn__builtIn(settings, verbose, ...)
```

Arguments

settings	(list) from makeNetFunc
verbose	(logical) print messages
...	parameters for makePSN_NamedMatrix()

Value

(char) names of networks created. Side effect of network creation.

---

psn__corr *wrapper for PSNs using Pearson correlation*

Description

wrapper for PSNs using Pearson correlation

Usage

```
psn__corr(settings, verbose, ...)
```

Arguments

settings	(list) from makeNetFunc
verbose	(logical) print messages
...	parameters for makePSN_NamedMatrix()

Value

(char) names of networks created. Side effect of network creation.

---

psn__custom *make PSN for custom similarity functions*

Description

make PSN for custom similarity functions

Usage

```
psn__custom(settings, fn, verbose, ...)
```

Arguments

settings	(list) from makeNetFunc
fn	(function) custom similarity function
verbose	(logical) print messages
...	parameters for makePSN_NamedMatrix()

Value

(char) names of networks created. Side effect of network creation.

---

randAlphanumString *Generate random alphanumerical string of length 10*

Description

Generate random alphanumerical string of length 10

Usage

```
randAlphanumString(numStrings = 1L)
```

Arguments

numStrings	(integer) number of strings to generate

Details

Used to create multiple temporary directories during an R session

Value

vector of length n, each with 10-char alphanumerical strings

Examples

```
randAlphanumString()
```

`readPathways` *Parse GMT file and return pathways as list*

Description

Parse GMT file and return pathways as list

Usage

```
readPathways(
  fname,
  MIN_SIZE = 10L,
  MAX_SIZE = 200L,
  EXCLUDE_KEGG = TRUE,
  IDasName = FALSE,
  verbose = TRUE,
  getOrigNames = FALSE
)
```

Arguments

`fname`	(char) path to pathway file in gmt format pathway score to include pathway in the filter list
`MIN_SIZE`	(integer) min num genes allowed in a pathway. Pathways with fewer number of genes are excluded from the output list
`MAX_SIZE`	(integer) max num genes allowed in a pathway. Pathways with gene counts greater than this are excluded from the output list
`EXCLUDE_KEGG`	(boolean) If TRUE exclude KEGG pathways. Our experience has been that some KEGG gene sets are to broad to be physiologically relevant
`IDasName`	(boolean) Value for key in output list. If TRUE, uses db name and ID as name (e.g. KEGG:hsa04940) If FALSE, pathway name.
`verbose`	(logical) print detailed messages
`getOrigNames`	(logical) when TRUE also returns a mapping of the cleaned pathway names to the original names

Details

The GMT file format currently supported should match the ones found at http://downloads.baderlab.org. The original GMT file format is: <set name><set description><member 1><member 2>...<member N>, one row per set, with values tab-delimited. The version at baderlab.org has additional unique formatting of the <set name> column as follows: <pathway_full_name> This function requires the specific formatting of the first column to assign the key name of the output list (see `useIDasName` argument).

Value

Depends on value of getOrigNames. If FALSE (Default), list with pathway name as key, vector of genes as value. If TRUE, returns list of length two, (1) geneSets: pathway-gene mappings as default, (2) pNames: data.frame with original and cleaned names.

Examples

```
pathFile <- fetchPathwayDefinitions("October",2020)
pathwayList    <- readPathways(pathFile)
```

`replacePattern` *Replace pattern in all files in dir*

Description

find/replace pattern in all files of specified file type in specified directory. Needed to modify number format when intefacing with GeneMANIA, on French locale machines. Without this step, CacheBuilder throws error with commas.

Usage

```
replacePattern(pattern = ",", target = ".", path = getwd(), fileType = "txt$")
```

Arguments

`pattern`	(char) pattern to find
`target`	(char) pattern to replace
`path`	(char) dir to replace pattern in
`fileType`	(char) pattern for files to replace pattern in

Value

No value. Files have patterns replaced in place.

RR_featureTally *Computes positive and negative calls upon changing stringency of feature selected networks (binary networks only)*

Description

Computes positive and negative calls upon changing stringency of feature selected networks (binary networks only)

Usage

```
RR_featureTally(
  netmat,
  phenoDF,
  TT_STATUS,
  predClass,
  pScore,
  outDir = tempdir(),
  enrichLabels = TRUE,
  enrichedNets,
  maxScore = 30L,
  verbose = FALSE
)
```

Arguments

netmat	(matrix) output of countPatientsInNet. Should contain all patients in dataset that overlap 1+ network
phenoDF	(data.frame) patient ID and STATUS
TT_STATUS	(list) output of splitTestTrain_partition; should be same as used for cross validation
predClass	(char) class to be predicted
pScore	(list of data.frames) contains 10-fold CV score, one entry for each resampling of the data. The data.frame has two columns: 1) pathway name, 2) pathway score
outDir	(char) path to dir where results should be written
enrichLabels	(logical) was network label enrichment used?
enrichedNets	(list of chars) networks passing network label enrichment
maxScore	(integer) max achievable score for pathways corresponding to N-way resampling
verbose	(logical) print messages

Details

This function computes predictor performance in the context of binary networks, where + and - calls are based on membership (or lack thereof) in feature selected networks. An example would be networks based on CNV occurrence in cellular pathways; in this use case, a + is based on patient membership in feature-selected networks. This function takes the output data from a feature selection exercise and computes the number and fraction of positive and negative calls at each level of feature selection stringency. The output of this function can then be used to compute performance measures such as the ROC or precision-recall curve.

Value

(list) 1) cumulativeFeatScores: pathway name, cumulative score over N-way data resampling. 2) performance_denAllNets: positive,negative calls at each cutoff: network score cutoff (score); num networks at cutoff (numPathways) ; total +, ground truth (pred_tot); + calls (pred_ol); + calls as pct of total (pred_pct); total -, ground truth (other_tot) ; - calls (other_ol) ; - calls as pct of total (other_pct) ; ratio of pred_pct and other_pct (rr) ; min. pred_pct in all resamplings (pred_pct_min) ; max pred_pct in all resamplings (pred_pct_max) ; min other_pct in all resamplings (other_pct_min); max other_pct in all resamplings (other_pct_max) 3) performance_denEnrichedNets: positive, negative calls at each cutoff label enrichment option: format same as performance_denAllNets. However, the denominator here is limited to patients present in networks that pass label enrichment 4) resamplingPerformance: breakdown of performance for each of the resamplings, at each of the cutoffs. This is a list of length 2, one for allNets and one for enrichedNets. The value is a matrix with (resamp * 7) columns and S rows, one row per score. The columns contain the following information per resampling: 1) pred_total: total num patients of predClass 2) pred_OL: num of pred_total with a CNV in the selected net 3) pred_OL_pct: 2) divided by 1) (percent) 4) other_total: total num patients of other class(non-predClass) 5) other_OL: num of other_total with CNV in selected net 6) other_OL_pct: 5) divided by 4) (percent) 7) relEnr: 6) divided by 3).

Examples

```
data(cnv_patientNetCount) # patient presence/absence in nets
data(cnv_pheno) # patient ID, label
data(cnv_netScores) # network scores for resampling
data(cnv_TTstatus) # train/test status
data(cnv_netPass)  # nets passing label enrichment

d <- tempdir()
out <- RR_featureTally(cnv_patientNetCount,
cnv_pheno,cnv_TTstatus,"case",cnv_netScores,
outDir=d,enrichLabels=TRUE,enrichedNets=cnv_netPass,
maxScore=30L)
print(summary(out))
```

`runFeatureSelection` *Run GeneMANIA cross-validation with a provided subset of networks*

Description

Run GeneMANIA cross-validation with a provided subset of networks

Usage

```
runFeatureSelection(
  trainID_pred,
  outDir,
  dbPath,
  numTrainSamps = NULL,
  incNets = "all",
  orgName = "predictor",
  fileSfx = "CV",
  verbose = FALSE,
  numCores = 2L,
  JavaMemory = 6L,
  verbose_runQuery = FALSE,
  debugMode = FALSE,
  ...
)
```

Arguments

trainID_pred	(char) vector with universe of predictor class patients (ie all that can possibly be included in the query file
outDir	(char) directory to store query file and GM results
dbPath	(char) path to GeneMANIA generic database with training population
numTrainSamps	(integer) number of training samples in total leave blank to use 5 training samples in order to save memory
incNets	(char) vector of networks to include in this analysis (features/pathway names). Useful for subset-based feature selection
orgName	(char) organism name for GeneMANIA generic database. The default value will likely never need to be changed.
fileSfx	(char) file suffix
verbose	(logical) print messages
numCores	(logical) num parallel threads for cross-validation
JavaMemory	(integer) memory for GeneMANIA run, in Gb.
verbose_runQuery	(logical) print messages for runQuery()
debugMode	(logical) when TRUE runs jobs in serial instead of parallel and prints verbose messages. Also prints system Java calls and prints all standard out and error output associated with these calls.
...	args for makeQueries()

Details

Creates query files, runs GM for 10-fold cross validation.

Value

No value. Side effect of generating feature scores.

Examples

```
data(MB.pheno)
dbPath <- system.file("extdata","dbPath",package="netDx")
runFeatureSelection(MB.pheno$ID[which(MB.pheno$STATUS%in% 'WNT')],
tempdir(),dbPath,103L)
```

---

runQuery *Run a query*

Description

Run a query

Usage

```
runQuery(
  dbPath,
  queryFiles,
  resDir,
  verbose = TRUE,
  JavaMemory = 6L,
  numCores = 1L,
  debugMode = FALSE
)
```

Arguments

dbPath	(char) path to directory with GeneMANIA generic database
queryFiles	(list(char)) paths to query files
resDir	(char) path to output directory
verbose	(logical) print messages
JavaMemory	(integer) Memory for GeneMANIA (in Gb) - a total of numCores*GMmemory will be used and distributed for all GM threads
numCores	(integer) number of CPU cores for parallel processing
debugMode	(logical) when TRUE runs jobs in serial instead of parallel and prints verbose messages. Also prints system Java calls.

Value

(char) path to GeneMANIA query result files with patient similarity rankings (*PRANK) and feature weights (*NRANK) of results file

Examples

```
dbPath <- system.file("extdata","dbPath",package="netDx")
queryFile <- system.file("extdata","GM_query.txt",package="netDx")
runQuery(dbPath, queryFile,tempdir())
```

---

setupFeatureDB *setup database of features for feature selection*

Description

Creates all the input files for the collection of features used in feature selection.

Usage

```
setupFeatureDB(pheno, prepDir = tempdir())
```

Arguments

pheno	(data.frame) patient metadata. Must contain ID column
prepDir	(char) directory in which to setup database

Value

(data.frame) internal numerical id for patients (INTERNAL_ID) and user-provided ID (ID)

Examples

```
data(xpr,pheno)
pathwayList <- list(pathA=rownames(xpr)[1:10],pathB=rownames(xpr)[21:50])

dataList <- list(rna=xpr)  #only one layer type
groupList <- list(rna=pathwayList) # group genes by pathways

makeNets <- function(dataList, groupList, netDir,...) {
    netList <- makePSN_NamedMatrix(dataList[['rna']],
rownames(dataList[['rna']]),
     groupList[['rna']],netDir,verbose=FALSE,
writeProfiles=TRUE,...)
    unlist(netList)
}
tmpDir <- tempdir(); netDir <- paste(tmpDir,"nets",sep=getFileSep())
dir.create(netDir,recursive=TRUE)

pheno_id <- setupFeatureDB(pheno,netDir)
```

silh *Toy network.*

Description

List with two entries. net: Network specification. "X" and "Y" are source and target columns respectively. "DIST" specifies weights. groups: Node labsls. A data.frame with columns "ID" and "GROUP"

Usage

```
data(silh)
```

Examples

```
data(silh)
summary(silh)
silh$net
silh$groups
```

sim.eucscale *Similarity method. Euclidean distance followed by exponential scaling*

Description

Computes Euclidean distance between patients. A scaled exponential similarity kernel is used to determine edge weight. The exponential scaling considers the K nearest neighbours, so that similarities between non-neighbours is set to zero. Alpha is a hyperparameterthat determines decay rate of the exponential. For details, see Wang et al. (2014). Nature Methods 11:333.

Usage

```
sim.eucscale(dat, K = 20, alpha = 0.5)
```

Arguments

dat	(data.frame) Patient data; rows are measures, columns are patients.
K	(integer) Number of nearest neighbours to consider (K of KNN)
alpha	(numeric) Scaling factor for exponential similarity kernel. Recommended range between 0.3 and 0.8.

Value

symmetric matrix of size ncol(dat) (number of patients) containing pairwise patient similarities

Examples

```
data(xpr)
sim <- sim.eucscale(xpr)
```

`sim.pearscale` *various similarity functions Similarity function: Pearson correlation followed by exponential scaling*

Description

Computes Pearson correlation between patients. A scaled exponential similarity kernel is used to determine edge weight. The exponential scaling considers the K nearest neighbours, so that similarities between non-neighbours is set to zero. Alpha is a hyperparameter that determines decay rate of the exponential. For details see Wang et al. (2014). Nature Methods 11:333.

Usage

```
sim.pearscale(dat, K = 20, alpha = 0.5)
```

Arguments

`dat`	(data.frame) Patient data; rows are measures, columns are patients.
`K`	(integer) Number of nearest neighbours to consider (K of KNN)
`alpha`	(numeric) Scaling factor for exponential similarity kernel. Recommended range between 0.3 and 0.8.

Value

symmetric matrix of size ncol(dat) (number of patients) containing pairwise patient similarities

Examples

```
data(xpr)
sim <- sim.pearscale(xpr)
```

simpleCap *simple capitalization*

Description

simple capitalization

Usage

```
simpleCap(x)
```

Arguments

x	(char) name

Details

used to format feature names so they are not in all-caps

Value

(char) Changes case so start of each word is in upper-case, and the rest is in lowercase

Examples

```
simpleCap('this IS a TEST sEnTenCe')
```

smoothMutations_LabelProp *This function applies the random walk with restart propagation algorithm to a matrix of patients profiles*

Description

This function applies the random walk with restart propagation algorithm to a matrix of patients profiles

Usage

```
smoothMutations_LabelProp(mat, net, numCores = 1L)
```

Arguments

mat	(data.frame) Sparse matrix of binarized patient profiles, with rownames being unique patients and columns, unique genes. Entry [i,j] is set to 1 if patient j has a mutation in gene i.
net	(data.frame) Interaction network provided as an adjacency matrix (i.e. symmetric)
numCores	(integer) Number of cores for parallel processing

Details

A network is an undirected graph G defined by a set of nodes corresponding to genes, and edges connecting nodes with an experimental evidence of interaction. A priori nodes are genes for which an information is known. A novel node is a candidate for being associated to the nodes above based on their information. A node prediction task leads to detect novel nodes and propagation techniques are largely applied for the purpose. Network-based propagation algorithms for node prediction transfer the information from a priori nodes to any other node in a network. Each node gets an imputation value which assesses how much information got. The prediction is based on the guilty-by-association principle. A node with a high imputation value has a high probability to be associated to a priori nodes. E.g. in a house where room A has one heater, if room B is the second hottest room it means that B is close to A and that there is a high probability that they share a door or wall. These algorithms exploit the global topology of the network. However, when they are applied to detect if unknown nodes are functionally associated to known ones, they may suffer of a drawback depending by the context. In biology, two functionally related fragments interact physically (direct interaction) or interact indirectly thanks to one or very few mediators. Therefore, exploring too far similarities between nodes can introduce noise in the prediction. We apply a random walk with restart propagation algorithm which resolution is set to 0.2 for giving high values only to the close neighbours of the a priori nodes.

Value

(data.frame) Continuous matrix of patient profiles in which each gene has the final propagation score

Examples

```
suppressWarnings(suppressMessages(require(MultiAssayExperiment)))
require(doParallel)

# load mutation and phenotype data
genoFile <- system.file("extdata","TGCT_mutSmooth_geno.txt",package="netDx")
geno <- read.delim(genoFile,sep="\t",header=TRUE,as.is=TRUE)
phenoFile <- system.file("extdata", "TGCT_mutSmooth_pheno.txt",
package="netDx")
pheno <- read.delim(phenoFile,sep="\t",header=TRUE,as.is=TRUE)
rownames(pheno) <- pheno$ID

# load interaction nets to smooth over
require(BiocFileCache)
netFileURL <- paste("https://download.baderlab.org/netDx/",
```

```
"supporting_data/CancerNets.txt",sep="")
cache <- rappdirs::user_cache_dir(appname = "netDx")
bfc <- BiocFileCache::BiocFileCache(cache,ask=FALSE)
netFile <- bfcrpath(bfc,netFileURL)
cancerNets <- read.delim(netFile,sep="\t",header=TRUE,as.is=TRUE)
# smooth mutations
prop_net <- smoothMutations_LabelProp(geno,cancerNets,numCores=1L)
```

`sparsify2` *cleaner sparsification routine*

Description

cleaner sparsification routine

Usage

```
sparsify2(
  W,
  outFile = paste(tempdir(), "tmp.txt", sep = getFileSep()),
  cutoff = 0.3,
  maxInt = 50,
  EDGE_MAX = 1000,
  includeAllNodes = TRUE,
  verbose = TRUE
)
```

Arguments

`W`	(matrix) similarity matrix
`outFile`	(char) path to file to write sparsified network
`cutoff`	(numeric) edges with weight smaller than this are set to NA
`maxInt`	(numeric) max num edges per node.
`EDGE_MAX`	(numeric) max num edges in network
`includeAllNodes`	(logical) if TRUE, ensures at least one edge is present for each patient. This feature is required when sparsification excludes test patients that are required to be classified. If the sparsification rules exclude all edges for a patient and this flag is set, then the strongest edge for each missing patient is added to the net. Note that this condition results in the total number of edges potentially exceeding EDGE_MAX
`verbose`	(logical) print detailed messages, useful for debugging

Details

Sparsifies similarity matrix to keep strongest edges. Sets diagonal and edges < cutoff to NA. Keeps strongest maxInt edges per node. Ties are ignored. Keeps a max of EDGE_MAX edges in the network.

Value

writes SIF content to text file (node1,node2,edge weight)

Examples

```
data(xpr);
sparsify2(cor(xpr))
```

`sparsify3` *cleaner sparsification routine - faster, matrix-based version*

Description

cleaner sparsification routine - faster, matrix-based version

Usage

```
sparsify3(
  W,
  outFile = sprintf("%s/tmp.txt", tempdir()),
  cutoff = 0.3,
  maxInt = 50,
  EDGE_MAX = Inf,
  includeAllNodes = TRUE,
  verbose = TRUE
)
```

Arguments

`W`	(matrix) similarity matrix
`outFile`	(char) path to file to write sparsified network
`cutoff`	(numeric) edges with weight smaller than this are set to NA
`maxInt`	(numeric) max num edges per node.
`EDGE_MAX`	(numeric) max num edges in network
`includeAllNodes`	(logical) if TRUE, ensures at least one edge is present for each patient. This feature is required when sparsification excludes test patients that are required to be classified. If the sparsification rules exclude all edges for a patient and this flag is set, then the strongest edge for each missing patient is added to the net. Note that this condition results in the total number of edges potentially exceeding EDGE_MAX
`verbose`	(logical) print detailed messages, useful for debugging

Details

Sparsifies similarity matrix to keep strongest edges. Sets diagonal and edges < cutoff to NA. Keeps strongest maxInt edges per node. Ties are ignored. Keeps a max of EDGE_MAX edges in the network.

Value

writes SIF content to text file (node1,node2,edge weight)

Examples

```
m <- matrix(runif(500*500),nrow=500)
y <- sparsify2(m)
m <- matrix(runif(500*500),nrow=500)
y <- sparsify2(m)
```

---

splitTestTrain *Split samples into train/test*

Description

Split samples into train/test

Usage

```
splitTestTrain(pheno_DF, pctT = 0.7, verbose = FALSE)
```

Arguments

pheno_DF	(data.frame) patient information Must contain the following columns: 1. ID: (char) patient IDs 2. STATUS: (char) patient classes. Values not equal to predClass will be considered as 'other' Expects rows with unique IDs
pctT	(numeric between 0 and 1) Fraction of patients to randomly assign to the training set. The remainder will be used for blind test set
verbose	(logical) print messages

Value

(char) vector of length nrow(pheno_DF), with values of 'TRAIN' or 'TEST'. The order corresponds to pheno_DF; a patient labelled 'TRAIN' has been assigned to the training set, and one labelled 'TEST' as been assigned to the test set.

Examples

```
data(pheno)
x <- splitTestTrain(pheno)
```

splitTestTrain_resampling *Assign train/test labels over several resamplings of the data.*

Description

Assign train/test labels over several resamplings of the data.

Usage

```
splitTestTrain_resampling(pheno_DF, nFold = 3L, predClass, verbose = FALSE)
```

Arguments

pheno_DF	(data.frame) table with patient ID and status. Must contain columns for Patient ID (named 'ID') and class (named 'STATUS'). Status should be a char; value of predictor class should be specified in predClass param; all other values are considered non-predictor class Expects rows with unique IDs Rows with duplicate IDs will be excluded.
nFold	(integer) number of resamplings. Each sample will be a test sample exactly once.
predClass	(char) name of predictor class
verbose	(logical) print messages

Details

This function is useful when feature selection needs to occur over multiple resamplings of the data, as a strategy to reduce overfitting. Each sample serves as a test for exactly one resampilng, and as a training sample for the others. The method is provided with the positive label and splits the samples so that an even number of positive and negative classes are represented in all the resamplings (i.e. it avoids the situation where one resampling has too many positives and another has too few).

Value

(list) of length nFold, each with char vector of length nrow(pheno_DF). Values of 'TRAIN' or 'TEST'

Examples

```
data(pheno)
x <- splitTestTrain_resampling(pheno,predClass='LumA')
```

`subsampleValidationData` *Subsample a hold-out set from a larger patient dataset*

Description

Subsample a hold-out set from a larger patient dataset

Usage

```
subsampleValidationData(dataMAE, pctValidation = 0.2, verbose = TRUE)
```

Arguments

`dataMAE`	(MultiAssayExperiment) patient data to be subsampled. Must have columns ID (patient ID) and STATUS
`pctValidation`	(numeric) Fraction of dataset to include in the validation set. Value from 0.05 to 0.95.
`verbose`	(logical)

Details

Creates a partition of data to be used for model validation after initial model building. In netDx, buildPredictor() is used for model training, and selected features from this exercise are used to validate a held-out dataset with the predict() function. Note that this function identifies a random subsample, which may result in a validation sample that is not representative of your training bias. If this method is used, please use data exploration techniques (e.g. UMAP) to ensure that validation accuracy is not confounded by stratification.

Value

(list) Keys are trainMAE and validationMAE. These contain corresponding MultiAssayExperiments for training and test data

`thresholdSmoothedMutations` *Apply discretization to the matrix resulted from the propagation on the sparse patient matrix*

Description

Apply discretization to the matrix resulted from the propagation on the sparse patient matrix

Usage

```
thresholdSmoothedMutations(
  smoothedMutProfile,
  unsmoothedMutProfile,
  nameDataset,
  n_topXmuts = c(10)
)
```

Arguments

smoothedMutProfile	(data.frame) continous matrix of patient profiles resulting from applying :.,$ s/network-based propagation algorithm (smoothMutations_LabelProp()) on a binary somatic mutation sparse matrix.
unsmoothedMutProfile	(data.frame) binary somatic mutation sparse matrix. Rownames are unique genes. Colnames are unique patients. A cell contains a zero or a one.
nameDataset	(char) for titles on plot
n_topXmuts	(numeric between 0 and 1) percent of top mutations to keep. This function converts these to 1.0 when binarizing, so they remain in the thresholded output matrix; other mutations are set to zero.

Details

This function is included in the netDx use case which involves propagating the sparse matrix of patient's profiles to reduce its sparsity. This function applies discretization on the propagated matrix of patient profiles. It sets to 1 the genes which got the highest propagation value. While, the remaining genes are set to 0. This discretization is driven by the fact that higher is the propagation value and higher is the chance that the gene is involved in the patient condition and expression/mutation profile. On the contrary, genes which got either a medium or a low value are not trustable.

Value

(data.frame) binary somatic mutation matrix which sparsity has been decreased

Examples

```
suppressWarnings(suppressMessages(require(MultiAssayExperiment)))
require(doParallel)

# load mutation and phenotype data
genoFile <- system.file("extdata","TGCT_mutSmooth_geno.txt",package="netDx")
geno <- read.delim(genoFile,sep="\t",header=TRUE,as.is=TRUE)
phenoFile <- system.file("extdata", "TGCT_mutSmooth_pheno.txt",
package="netDx")
pheno <- read.delim(phenoFile,sep="\t",header=TRUE,as.is=TRUE)
rownames(pheno) <- pheno$ID

# load interaction nets to smooth over
```

```
require(BiocFileCache)
netFileURL <- paste("https://download.baderlab.org/netDx/",
"supporting_data/CancerNets.txt",sep="")
cache <- rappdirs::user_cache_dir(appname = "netDx")
bfc <- BiocFileCache::BiocFileCache(cache,ask=FALSE)
netFile <- bfcrpath(bfc,netFileURL)
cancerNets <- read.delim(netFile,sep="\t",header=TRUE,as.is=TRUE)
# smooth mutations
prop_net <- smoothMutations_LabelProp(geno,cancerNets,numCores=1L)
genoP <- thresholdSmoothedMutations(
   prop_net,geno,"TGCT_CancerNets",c(20)
  )
```

`toymodel` *Example model returned by a buildPredictor() call.*

Description

Output of buildPredictor() generated by a simple use-case of binary breast tumour classification using two genomic data sources. BRCA data were downloaded using curatedTCGAData for mRNA and miRNA expression. buildPredictor()] was run by scoring features out of 2, with selected features passing 1 out of 2. Tumours were labelled either "Luminal.A" or "other".

Usage

```
data(toymodel)
```

Examples

```
data(toymodel)
head(toymodel)
```

`tSNEPlotter` *Plot tSNE*

Description

Plot tSNE

Usage

```
tSNEPlotter(psn, pheno, ...)
```

Arguments

psn	(matrix) Patient similarity network represented as adjacency matrix (symmetric). Row and column names are patient IDs. Note that NA values will be replaced by very small number (effectively zero).
pheno	(data.frame) Patient labels. ID column is patient ID and STATUS is patient label of interest. tSNE will colour-code nodes by patient label.
...	Parameters for Rtsne() function.

Details

Plots tSNE of integrated patient similarity network using Rtsne

Value

(Rtsne) output of Rtsne call. Side effect of tSNE plot

Examples

```
pid <- paste("P",1:100,sep="")
psn <- matrix(rnorm(100*100),nrow=100,dimnames=list(pid,pid))
psn[lower.tri(psn)] <- NA; diag(psn) <- NA
psn2 <- reshape2::melt(psn); psn2 <- psn2[-which(is.na(psn2[,3])),]
colnames(psn2) <- c("SOURCE","TARGET","WEIGHT")
pheno <- data.frame(ID=pid,STATUS=c(rep("control",50),rep("case",50)))
tSNEPlotter(psn2,pheno)
```

---

updateNets *Synchronize patient set in sample table and network table.*

Description

Synchronize patient set in sample table and network table.

Usage

```
updateNets(
  p_net,
  pheno_DF,
  writeNewNets = TRUE,
  oldNetDir,
  newNetDir,
  verbose = TRUE,
  ...
)
```

Arguments

p_net	(matrix) rows are patients, columns are networks. a[i,j] = 1 if patient i occurs in network j, else 0.
pheno_DF	(data.frame) patient ID and STATUS.
writeNewNets	(logical) if TRUE writes new networks to newNetDir.
oldNetDir	(char) path to directory with networks to be updated
newNetDir	(char) path to directory where updated networks are to be written
verbose	(logical) print messages
...	passed to pruneNets()

Details

This function is useful in applications with highly missing data or where each patient contributes data points not present in the others; e.g. networks based on individual patient CNVs, which are highly sparse. In such a scenario, any kind of patient subsetting - for example, limiting to training samples - changes the population of eligible networks for analysis. Networks that no longer have samples, or that have one patient with the neighbour removed, have to be excluded. This function updates networks and patients so that each network contains at least two patients and only patients in networks are retained. In other words, it keeps pheno_DF and p_net in sync.

Value

list with updated p_net and pheno_DF. pheno_DF will contain IDs in the updated p_net. p_net will contain only those networks with 2+ patients and those patients present in 1+ network.

Examples

```
data(npheno)
netDir <- system.file("extdata","example_nets",package="netDx")
netmat <- countPatientsInNet(netDir,dir(netDir,pattern='txt$'), npheno[,1])
x <- updateNets(netmat, npheno,writeNewNets=FALSE)
```

---

writeNetsSIF *write patient networks in Cytoscape's .sif format*

Description

write patient networks in Cytoscape's .sif format

Usage

```
writeNetsSIF(
  netPath,
  outFile = paste(tempdir(), "out.sif", sep = getFileSep()),
  netSfx = "_cont.txt"
)
```

Arguments

netPath	(char): vector of path to network files; file suffix should be '_cont.txt' networks should be in format: A B 1 where A and B are nodes, and 1 indicates an edge between them
outFile	(char) path to .sif file
netSfx	(char) suffix for network file name

Details

Converts a set of binary interaction networks into Cytoscape's sif format. (http://wiki.cytoscape.org/Cytoscape_User_Manual
This utility permits visualization of feature selected networks.

Value

No value. Side effect of writing all networks to outFile

Examples

```
netDir <- system.file("extdata","example_nets",package="netDx")
netFiles <- paste(netDir,dir(netDir,pattern='txt$'),
sep=getFileSep())
writeNetsSIF(netFiles,'merged.sif',netSfx='.txt')
```

---

writeQueryBatchFile *Write batch.txt file required to create GeneMANIA database*

Description

Write batch.txt file required to create GeneMANIA database

Usage

```
writeQueryBatchFile(
  netDir,
  netList,
  outDir = tempdir(),
  idFile,
  orgName = "predictor",
  orgDesc = "my_predictor",
  orgAlias = "my_predictor",
  taxID = 1339
)
```

Arguments

netDir	(char) path to dir with networks
netList	(char) vector of network names
outDir	(char) directory to write batch file
idFile	(char) path to file with patient IDs
orgName	(char) organism name. Don't change the default unless you know what you are doing.
orgDesc	(char) organism description. Similar to orgName, don't change the default
orgAlias	(char) organism alias. Similar to orgName, don't change the default.
taxID	(integer) taxonomyID required for GeneMANIA . Similar to orgName, don't change the default.

Details

This file is used to compile features into a single database for feature selection.

Value

No value. Side effect of writing batch file to <outDir>/batch.txt.

Examples

```
data(npheno)
netDir <- system.file("extdata","example_nets",package="netDx")
netList <- dir(netDir,pattern='txt$')
writeQueryBatchFile(netDir,netList, tempdir(), npheno$ID)
```

---

writeQueryFile *Wrapper to write GeneMANIA query file*

Description

Wrapper to write GeneMANIA query file

Usage

```
writeQueryFile(
  qSamps,
  incNets = "all",
  numReturn = 1L,
  outFile,
  orgName = "predictor"
)
```

Arguments

qSamps	(char) vector of patient IDs in query
incNets	(char) vector of networks to include in this analysis (features/pathway names). Useful for subset-based feature selection
numReturn	(integer) number of patients to return in ranking file
outFile	(char) path to output file
orgName	(char) organism name

Value

No value. Side effect of writing the query file to `outFile`

Examples

```
data(pheno)
writeQueryFile(pheno$ID[seq_len(5)], 'all',nrow(pheno), 'myquery.txt')
```

---

writeWeightedNets *Write an integrated similarity network consisting of selected networks.*

Description

Write an integrated similarity network consisting of selected networks.

Usage

```
writeWeightedNets(
  patientIDs,
  netIDs,
  netDir,
  keepNets,
  filterEdgeWt = 0,
  aggNetFunc = "MAX",
  limitToTop = 50L,
  plotEdgeDensity = FALSE,
  verbose = FALSE
)
```

Arguments

patientIDs	(data.frame) patient identifiers. Columns include internally-generated identifiers (GM_ID) and user-provided identifiers (ID)
netIDs	(data.frame) network metadata. Columns include internal network name (NET_ID), user-provided name (NETWORK). If a third optional column named "isBinary" is provided, and contains binary values (i.e. 1 and 0), that indicates that the network contains only binary weights and an alternate similarity computation (PropBinary) will be used (see description).

netDir	(char) path to directory containing interaction networks. Note that these are networks where the node IDs have been recoded by GeneMANIA (e.g. 1,2,3)
keepNets	(char or data.frame) networks to include in integrated net If data.frame must be in "NETWORK" column,other columns will be ignored. Mainly included as convenience so pathway scores can passed in table format (NETWORK), and a multiplier constant for edges in that network (WEIGHT)
filterEdgeWt	(numeric) keep edges with raw edge weight strictly greater than this value. Note that "raw" refers to this filter being applied before the multiplier is applied.
aggNetFunc	(char, one of: [MEAN|MAX]) Aggregate the network 2) MEAN: average of weighted edges (raw x netDx score) 3) MAX: max of raw edge weight
limitToTop	(integer) limit to top strongest connections. Set to Inf to list all connections.
plotEdgeDensity	(logical) plot density plot of edge weights, one per input net. Used to troubleshoot problems introduced by specific nets.
verbose	(logical) print messages if TRUE

Value

(list) 1) filterEdgeWt (numeric) Value of filterEdgeWt parameter 2) aggNetFunc (char) Value of aggNetFunc parameter 3) limitToTop (integer) Value of limitToTop parameter 4) aggNet (matrix) Value of limitToTop parameter File format is: 1) source patient (SOURCE) 2) target patient (TARGET) 3) network name (NET_NAME) 4) weight similarity for the network (WT_SIM)

---

xpr *Example expression matrix*

Description

data.frame with gene expression for 727 genes (rows) and 40 patients (columns). Data from TCGA breast cancer subtyping study.

Usage

```
data(xpr)
```

Source

The Cancer Genome Atlas. (2012). Nature 490:61-70.

References

The Cancer Genome Atlas. (2012). Nature 490:61-70.

Examples

```
data(xpr)
head(xpr)
```

Index