Package 'Pi'

October 12, 2021

Type Package

Title Leveraging Genetic Evidence to Prioritise Drug Targets at the Gene and Pathway Level

Version 2.4.0

Date 2021-1-4

Author Hai Fang, the ULTRA-DD Consortium, Julian C Knight

Maintainer Hai Fang <hfang.shanghai@gmail.com>

Depends igraph, dnet, ggplot2, graphics

Imports Matrix, GenomicRanges, GenomeInfoDb, supraHex, scales, grDevices, ggrepel, ROCR, randomForest, glmnet, lattice, caret, plot3D, stats, methods, MASS, IRanges, BiocGenerics, dplyr, tidyr, ggnetwork, osfr, RCircos, purrr, readr, tibble

Suggests foreach, doParallel, BiocStyle, knitr, rmarkdown, png, GGally, gridExtra, ggforce, fgsea, RColorBrewer, ggpubr, rtracklayer, ggbio, Gviz, data.tree, jsonlite

Description Priority index or Pi is developed as a genomic-led target prioritisation system. It integrates functional genomic predictors, knowledge of network connectivity and immune ontologies to prioritise potential drug targets at the gene and pathway level.

URL http://pi314.r-forge.r-project.org

BugReports https://github.com/hfang-bristol/Pi/issues

Collate 'ClassMethod-Pi.r' 'xRWR.r' 'xPier.r' 'xPierGenes.r' 'xPierSNPs.r' 'xPierPathways.r' 'xPierManhattan.r' 'xPierSubnet.r' 'xPierMatrix.r' 'xPierEvidence.r' 'xPredictROCR.r' 'xPredictCompare.r' 'xContour.r' 'xMLrandomforest.r' 'xPierSNPsAdv.r' 'xGSsimulator.r' 'xMLdotplot.r' 'xMLdensity.r' 'xMLzoom.r' 'xPierGSEA.r' 'xGSEAdotplot.r' 'xGSEAbarplot.r' 'xPierTrack.r' 'xPierTrackAdv.r' 'xGSEAconciser.r' 'xPierAnno.r' 'xMLglmnet.r' 'xMLfeatureplot.r' 'xMLparameters.r' 'xMLcaret.r' 'xMLcompare.r' 'xPierCross.r' 'xVisEvidence.r' 'xPierROCR.r' 'xMLrename.r' 'xVisEvidenceAdv.r' 'xCorrelation.r' 'xPierCor.r' 'xPierGRs.r' 'xPierABF.r' 'xPierSNPsAdvABF.r'

'xPierABFheatmap.r' 'xPierMRS.r' 'xHeatmap.r'
'xCheckParallel.r' 'xDefineNet.r' 'xSNPscores.r'
'xSNP2nGenes.r' 'xSNP2eGenes.r' 'xSNP2cGenes.r'
'xSparseMatrix.r' 'xSM2DF.r' 'xDefineEQTL.r' 'xDefineHIC.r'
'xEnricherGenes.r' 'xDefineOntology.r' 'xEnricher.r'
'xDAGanno.r' 'xColormap.r' 'xSubneterGenes.r'
'xSymbol2GeneID.r' 'xSNPlocations.r' 'xGR.r' 'xLiftOver.r'
'xGGnetwork.r' 'xPieplot.r' 'xGR2xGeneScores.r' 'xGRscores.r'
'xGR2xGenes.r' 'xGR2nGenes.r' 'xGRsort.r' 'xMEabf.r'
'xRDataLoader.r' 'xAggregate.r' 'xGeneID2Symbol.r' 'xCircos.r'
'xVisNet.r' 'xEnrichViewer.r' 'xVisKernels.r' 'xEnrichForest.r'
'xCombineNet.r' 'xLayout.r' 'xConverter.r'

License GPL-3

VignetteBuilder knitr

biocViews Software, Genetics, GraphAndNetwork, Pathways, GeneExpression, GeneTarget, GenomeWideAssociation, LinkageDisequilibrium, Network, HiC

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

git_branch RELEASE_3_13

git_last_commit e5fb99d

git_last_commit_date 2021-05-19

Date/Publication 2021-10-12

R topics documented:

`aOnto` *Definition for S3 class* `aOnto`

Description

`aOnto` has 2 components: g, anno.

Usage

```
aOnto(g, anno)

## S3 method for class 'aOnto'
print(x, ...)
```

Arguments

g	an igraph object
anno	a list
x	an object of class aOnto
...	other parameters

Value

an object of S3 class aOnto

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
aOnto(g, anno)

## End(Not run)
```

---

cTarget *Definition for S3 class* cTarget

Description

cTarget has 2 components: priority and predictor.

Usage

```
cTarget(priority, predictor)

## S3 method for class 'cTarget'
print(x, ...)
```

Arguments

priority	a data frame
predictor	a data frame
x	an object of class cTarget
...	other parameters

Value

an object of S3 class cTarget

Examples

```
## Not run:
# Load the library
library(Pi)

## End(Not run)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
cTarget(priority, predictor)

## End(Not run)
```

`dTarget` *Definition for S3 class* `dTarget`

Description

`dTarget` has 3 components: priority, predictor and metag.

Usage

```
dTarget(priority, predictor, metag)

## S3 method for class 'dTarget'
print(x, ...)
```

Arguments

`priority`	a data frame
`predictor`	a data frame
`metag`	an 'igraph' object
`x`	an object of class `dTarget`
`...`	other parameters

Value

an object of S3 class `dTarget`

Examples

```
## Not run:
# Load the library
library(Pi)

## End(Not run)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
```

```
dTarget(priority, predictor, metag)

## End(Not run)
```

EG *Definition for S3 class* EG

Description

EG has 1 component: gene_info.

Usage

```
EG(gene_info)

## S3 method for class 'EG'
print(x, ...)
```

Arguments

gene_info	a data frame
x	an object of class EG
...	other parameters

Value

an object of S3 class EG

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
EG(gene_info)

## End(Not run)
```

eGSEA *Definition for S3 class* eGSEA

Description

eGSEA mush have following components: df_summary, leading, full, cross.

Usage

```
eGSEA(df_summary, leading, full, cross)

## S3 method for class 'eGSEA'
print(x, ...)
```

Arguments

df_summary	a data frame
leading	a list
full	a list
cross	a matrix
x	an object of class eGSEA
...	other parameters

Value

an object of S3 class eGSEA

Examples

```
## Not run:
# Load the library
library(Pi)

## End(Not run)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
eGSEA(df_summary, leading, full, cross)

## End(Not run)
```

eTarget *Definition for S3 class* eTarget

Description

eTarget has 2 components: evidence and metag.

Usage

```
eTarget(evidence, metag)

## S3 method for class 'eTarget'
print(x, ...)
```

Arguments

evidence	a data frame
metag	an 'igraph' object
x	an object of class eTarget
...	other parameters

Value

an object of S3 class eTarget

Examples

```
## Not run:
# Load the library
library(Pi)

## End(Not run)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
eTarget(evidence, metag)

## End(Not run)
```

eTerm *Definition for S3 class* eTerm

Description

eTerm mush have following components: term_info, annotation, g, data, background, overlap, fc, zscore, pvalue, adjp, cross.

Usage

```
eTerm(
term_info,
annotation,
g,
data,
background,
overlap,
fc,
zscore,
pvalue,
adjp,
cross
)

## S3 method for class 'eTerm'
print(x, ...)
```

Arguments

term_info	a data frame
annotation	a list
g	an 'igraph' object
data	a vector
background	a vector
overlap	a vector
fc	a vector
zscore	a vector
pvalue	a vector
adjp	a vector
cross	a matrix
x	an object of class eTerm
...	other parameters

Value

an object of S3 class eTerm

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
eTerm(term_info, annotation, g, data, background, overlap, fc, zscore,
pvalue, adjp, cross)

## End(Not run)
```

GS *Definition for S3 class* GS

Description

GS has 2 components: set_info, gs.

Usage

```
GS(set_info, gs)

## S3 method for class 'GS'
print(x, ...)
```

Arguments

set_info	a data frame
gs	a list
x	an object of class GS
...	other parameters

Value

an object of S3 class GS

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
GS(set_info, gs)

## End(Not run)
```

iSubg *Definition for S3 class* iSubg

Description

iSubg has 2 components: g, ls_subg.

Usage

```
iSubg(g, ls_subg)

## S3 method for class 'iSubg'
print(x, ...)
```

Arguments

g	an igraph object
ls_subg	a list of igraph objects
x	an object of class iSubg
...	other parameters

Value

an object of S3 class iSubg

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
iSubg(g, ls_subg)

## End(Not run)
```

ls_eTerm *Definition for S3 class* ls_eTerm

Description

ls_eTerm has 3 components: df, mat and gp.

Usage

```
ls_eTerm(df, mat, gp)

## S3 method for class 'ls_eTerm'
print(x, ...)
```

Arguments

df	a data frame
mat	a matrix
gp	a ggplot object
x	an object of class ls_eTerm
...	other parameters

Value

an object of S3 class ls_eTerm

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
ls_eTerm(df, mat, gp)

## End(Not run)
```

---

pNode *Definition for S3 class* pNode

Description

pNode has 7 components: priority, g, SNP, Gene2SNP, nGenes, eGenes and cGenes.

Usage

```
pNode(priority, g, SNP, Gene2SNP, nGenes, eGenes, cGenes)

## S3 method for class 'pNode'
print(x, ...)
```

Arguments

priority	a data frame
g	an 'igraph' object
SNP	a data frame
Gene2SNP	a data frame
nGenes	a data frame
eGenes	a data frame
cGenes	a data frame
x	an object of class pNode
...	other parameters

Value

an object of S3 class pNode

Examples

```
## Not run:
# Load the library
library(Pi)

## End(Not run)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
pNode(evidence, metag)

## End(Not run)
```

---

pPerf *Definition for S3 class* pPerf

Description

pPerf mush have following components: PRS, AUROC, Fmax, ROC_perf, PR_perf, Pred_obj.

Usage

```
pPerf(PRS, AUROC, Fmax, ROC_perf, PR_perf, Pred_obj)

## S3 method for class 'pPerf'
print(x, ...)
```

Arguments

PRS	a data frame
AUROC	a scalar
Fmax	a scalar
ROC_perf	a ROCR 'performance' object for ROC curve
PR_perf	a ROCR 'performance' object for PR curve
Pred_obj	a ROCR 'prediction' object for other performance measures
x	an object of class pPerf
...	other parameters

Value

an object of S3 class pPerf

Examples

```
## Not run:
# Load the library
library(Pi)

## End(Not run)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
pPerf(PRS, AUROC, Fmax, ROC_perf, PR_perf, Pred_obj)

## End(Not run)
```

---

sGS *Definition for S3 class* sGS

Description

sGS mush have following components: GSN, GSP, g.

Usage

```
sGS(GSN, GSP, g)

## S3 method for class 'sGS'
print(x, ...)
```

Arguments

GSN	a vector
GSP	a vector
g	an 'igraph' object
x	an object of class sGS
...	other parameters

Value

an object of S3 class sGS

Examples

```
## Not run:
# Load the library
library(Pi)

## End(Not run)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
```

```
sGS(GSN, GSP, g)

## End(Not run)
```

---

sTarget *Definition for S3 class* sTarget

Description

sTarget mush have following components: priority, predictor, performance, importance, evidence.

Usage

```
sTarget(priority, predictor, performance, importance, evidence)

## S3 method for class 'sTarget'
print(x, ...)
```

Arguments

priority	a data frame
predictor	a data frame
performance	a data frame
importance	a data frame
evidence	an 'eTarget' object
x	an object of class sTarget
...	other parameters

Value

an object of S3 class sTarget

Examples

```
## Not run:
# Load the library
library(Pi)

## End(Not run)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
sTarget(priority, predictor, performance, importance, evidence)

## End(Not run)
```

xAggregate *Function to aggregate data respecting number of features*

Description

xAggregate is supposed to aggregate data respecting number of features. Per row, the aggregated is the sum of two items: the number of features, and the sum of all but scaled into [0,0.9999999]. Also supported is the rank-transformation of the input data per column, binned into the predefined number of discrete bins.

Usage

```
xAggregate(data, bin = FALSE, nbin = 10, scale.log = TRUE, verbose =
TRUE)
```

Arguments

data	a data frame. The aggregation is done across columns per row. Each cell should contain positive values or NA; if infinite, it will be replaced with the maximum finite value
bin	logical to indicate whether the input data per column is rank-transformed into the predefined number of discrete bins. By default, it sets to false
nbin	the number of discrete bins. By default, it sets to 10 (only works when bin is true)
scale.log	logical to indicate whether the per-row sum is log-scaled. By default, it sets to true
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

a data frame with an appended column 'Aggregate'

Note

None

See Also

xAggregate

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# HiC-gene pairs per cell types/states
g <- xRDataLoader(RData.customised='ig.PCHiC',
RData.location=RData.location)
df <- do.call(cbind, igraph::edge_attr(g))
# aggregate over cell types/states
data <- df
data[data<5] <- NA
res <- xAggregate(data)

## End(Not run)
```

---

xCheckParallel *Function to check whether parallel computing should be used and how*

Description

`xCheckParallel` is used to check whether parallel computing should be used and how

Usage

```
xCheckParallel(multicores = NULL, verbose = TRUE)
```

Arguments

`multicores`	an integer to specify how many cores will be registered as the multicore parallel backend to the 'foreach' package. If NULL, it will use a half of cores available in a user's computer
`verbose`	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

TRUE for using parallel computing; FALSE otherwise

Note

Whether parallel computation with multicores is used is system-specific. Also, it will depend on whether these two packages "foreach" and "doParallel" have been installed.

See Also

`xCheckParallel`

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
xCheckParallel()

## End(Not run)
```

---

xCircos *Function to visualise a network as a circos plot*

Description

xCircos is used to visualise a network as a circos plot. The network must be a 'igraph' object. The degree of similarity between SNPs (or genes) is visualised by the colour of links. This function can be used either to visualise the most similar links or to plot links involving an input SNP (or gene).

Usage

```
xCircos(
g,
entity = c("SNP", "Gene", "Both"),
top_num = 50,
colormap = c("yr", "bwr", "jet", "gbr", "wyr", "br", "rainbow", "wb",
"lightyellow-orange"),
rescale = TRUE,
nodes.query = NULL,
ideogram = TRUE,
chr.exclude = "auto",
entity.label.cex = 0.7,
entity.label.side = c("in", "out"),
entity.label.track = 1,
entity.label.query = NULL,
GR.SNP = c("dbSNP_GWAS", "dbSNP_Common", "dbSNP_Single"),
GR.Gene = c("UCSC_knownGene", "UCSC_knownCanonical"),
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

g	an object of class "igraph". For example, it stores semantic similarity results with nodes for genes/SNPs and edges for pair-wise semantic similarity between them
entity	the entity of similarity analysis for which results are being plotted. It can be either "SNP" or "Gene"

`top_num` the top number of similarity edges to be plotted

`colormap` short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "lightyellow-orange" (by default), "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names

`rescale` logical to indicate whether the edge values are rescaled to the range [0,1]. By default, it sets to true

`nodes.query` nodes in query for which edges attached to them will be displayed. By default, it sets to NULL meaning no such restriction

`ideogram` logical to indicate whether chromosome banding is plotted

`chr.exclude` a character vector of chromosomes to exclude from the plot, e.g. c("chrX", "chrY"). By defautl, it is 'auto' meaning those chromosomes without data will be excluded. If NULL, no chromosome is excluded

`entity.label.cex` the font size of genes/SNPs labels. Default is 0.8

`entity.label.side` the position of genes/SNPs labels relative to chromosome ideogram. It can be "out" (by default) or "in"

`entity.label.track` an integer specifying the plot track for genes/SNPs labels. Default is 1

`entity.label.query` which genes/SNPs in query will be labelled. By default, it sets to NULL meaning all will be displayed. If labels in query can not be found, then all will be displayed

`GR.SNP` the genomic regions of SNPs. By default, it is 'dbSNP_GWAS', that is, SNPs from dbSNP (version 146) restricted to GWAS SNPs and their LD SNPs (hg19). It can be 'dbSNP_Common', that is, Common SNPs from dbSNP (version 146) plus GWAS SNPs and their LD SNPs (hg19). Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to dbSNP IDs. Then, tell "GR.SNP" with your RData file name (with or without extension), plus specify your file RData path in "RData.location". Note: you can also load your customised GR object directly

`GR.Gene` the genomic regions of genes. By default, it is 'UCSC_knownGene', that is, UCSC known genes (together with genomic locations) based on human genome assembly hg19. It can be 'UCSC_knownCanonical', that is, UCSC known canonical genes (together with genomic locations) based on human genome assembly hg19. Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to Gene Symbols. Then, tell "GR.Gene" with your RData file name (with or without extension), plus specify your file RData path in "RData.location". Note: you can also load your customised GR object directly

verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details
guid	a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details

Value

a circos plot with edge weights between input snps/genes represented by the colour of the links

Note

none

See Also

xRDataLoader

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
library(RCircos)

# provide genes and SNPs reported in GWAS studies
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)

# 1) SNP-based similarity analysis using GWAS Catalog traits (mapped to EF)
## Get lead SNPs reported in AS GWAS
example.snps <- names(ImmunoBase$AS$variants)
SNP.g <- xSocialiserSNPs(example.snps, include.LD=NA,
RData.location=RData.location)
# Circos plot of the EF-based SNP similarity network
#out.file <- "SNP_Circos.pdf"
#pdf(file=out.file, height=12, width=12, compress=TRUE)
xCircos(g=SNP.g, entity="SNP", RData.location=RData.location)
#dev.off()
# Circos plot involving nodes 'rs6871626'
xCircos(g=SNP.g, entity="SNP", nodes.query="rs6871626",
RData.location=RData.location)

# 2) Gene-based similarity analysis using Disease Ontology (DO)
## Get genes within 10kb away from AS GWAS lead SNPs
example.genes <- names(which(ImmunoBase$AS$genes_variants<=10000))
gene.g <- xSocialiserGenes(example.genes, ontology="DO",
RData.location=RData.location)
# Circos plot of the DO-based gene similarity network
#out.file <- "Gene_Circos.pdf"
#pdf(file=out.file, height=12, width=12, compress=TRUE)
```

```
xCircos(g=gene.g, entity="Gene", chr.exclude="chrY",
RData.location=RData.location)
#dev.off()

# 3) Advanced usages: Gene-SNP pairs from trans-eQTL mapping
JKscience_TS3A <- xRDataLoader(RData.customised='JKscience_TS3A',
RData.location=RData.location)
## extract the significant trans-eQTL in IFN
ind <- -1*log10(JKscience_TS3A$IFN_fdr)
ind <- which(!is.na(ind) & ind>2)
relations <- JKscience_TS3A[ind, c("Symbol","variant","IFN_fdr")]
relations <- data.frame(from=relations$Symbol, to=relations$variant,
weight=-log10(relations$IFN_fdr))
ig_Gene2SNP <- igraph::graph.data.frame(d=relations, directed=TRUE)
# Circos plot of the eQTL (Gene-SNP) network
#out.file <- "eQTL_Circos.pdf"
#pdf(file=out.file, height=12, width=12, compress=TRUE)
xCircos(g=ig_Gene2SNP, entity="Both", top_num=NULL,
nodes.query=c("GAD1","TNFRSF1B"), chr.exclude=NULL,
entity.label.side="out", RData.location=RData.location)
#dev.off()

## End(Not run)
```

---

xColormap *Function to define a colormap*

Description

xColormap is supposed to define a colormap. It returns a function, which will take an integer argument specifying how many colors interpolate the given colormap.

Usage

```
xColormap(
colormap = c("bwr", "jet", "gbr", "wyr", "br", "yr", "rainbow", "wb",
"heat",
"terrain", "topo", "cm", "ggplot2", "jet.top", "jet.bottom",
"jet.both", "spectral",
"ggplot2.top", "ggplot2.bottom", "ggplot2.both", "RdYlBu",
"brewer.BrBG",
"brewer.PiYG", "brewer.PRGn", "brewer.PuOr", "brewer.RdBu",
"brewer.RdGy",
"brewer.RdYlBu", "brewer.RdYlGn", "brewer.Spectral", "brewer.Blues",
"brewer.BuGn",
"brewer.BuPu", "brewer.GnBu", "brewer.Greens", "brewer.Greys",
"brewer.Oranges",
"brewer.OrRd", "brewer.PuBu", "brewer.PuBuGn", "brewer.PuRd",
"brewer.Purples",
```

```
"brewer.RdPu", "brewer.Reds", "brewer.YlGn", "brewer.YlGnBu",
"brewer.YlOrBr",
"brewer.YlOrRd", "rainbow_hcl", "heat_hcl", "terrain_hcl",
"diverge_hcl", "hcl_br",
"hcl_bp", "hcl_bb", "hcl_gp", "hcl_go", "hcl_cp", "hcl_cy", "hcl_co",
"sci_jco",
"sci_lancet", "sci_nejm", "sci_locuszoom"),
interpolate = c("spline", "linear"),
data = NULL,
zlim = NULL
)
```

Arguments

colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta), and "ggplot2" (emulating ggplot2 default color palette). Alternatively, any hyphen-separated HTML color names, e.g. "lightyellow-orange" (by default), "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names. It can also be a function of 'colorRampPalette'. It can also be "brewer.*" (RColorBrewer palette; see RColorBrewer::display.brewer.all()). It can be colorspace defaults ("rainbow_hcl","heat_hcl","terrain_hcl","diver or other useful ones ("hcl_br","hcl_bp","hcl_bb","hcl_gp","hcl_go","hcl_cp","hcl_cy","hcl_co")
interpolate	use spline or linear interpolation
data	NULL or a numeric vector
zlim	the minimum and maximum z/patttern values for which colors should be plotted, defaulting to the range of the finite values of z. Each of the given colors will be used to color an equispaced interval of this range. The midpoints of the intervals cover the range, so that values just outside the range will be plotted

Value

palette.name (a function that takes an integer argument for generating that number of colors interpolating the given sequence) or mapped colors if data is provided.

Note

The input colormap includes:

- "jet": jet colormap
- "bwr": blue-white-red
- "gbr": green-black-red
- "wyr": white-yellow-red
- "br": black-red

- "yr": yellow-red
- "wb": white-black
- "rainbow": rainbow colormap, that is, red-yellow-green-cyan-blue-magenta
- "ggplot2": emulating ggplot2 default color palette
- "spectral": emulating RColorBrewer spectral color palette
- Alternatively, any hyphen-separated HTML color names, e.g. "blue-black-yellow", "royalblue-white-sandybrown", "darkblue-lightblue-lightyellow-darkorange", "darkgreen-white-darkviolet", "darkgreen-lightgreen-lightpink-darkred". A list of standard color names can be found in http://html-color-codes.info/color-names

See Also

xColormap

Examples

```
# 1) define "blue-white-red" colormap
palette.name <- xColormap(colormap="bwr")
# use the return function "palette.name" to generate 10 colors spanning "bwr"
palette.name(10)

# 2) define default colormap from ggplot2
palette.name <- xColormap(colormap="ggplot2")
# use the return function "palette.name" to generate 3 default colors used by ggplot2
palette.name(3)

# 3) define brewer colormap called "RdYlBu"
palette.name <- xColormap(colormap="RdYlBu")
# use the return function "palette.name" to generate 3 default colors used by ggplot2
palette.name(3)

# 4) return mapped colors
xColormap(colormap="RdYlBu", data=runif(5))
```

---

xCombineNet *Function to combine networks from a list of igraph objects*

Description

xCombineNet is supposed to combine networks from a list of igraph objects.

Usage

```
xCombineNet(
list_ig,
combineBy = c("union", "intersect"),
attrBy = c("intersect", "union"),
```

```
keep.all.vertices = FALSE,
verbose = TRUE
)
```

Arguments

list_ig	a list of "igraph" objects or a "igraph" object
combineBy	how to resolve edges from a list of "igraph" objects. It can be "intersect" for intersecting edges and "union" for unionising edges (by default)
attrBy	the method used to extract node attributes. It can be "intersect" for intersecting node attributes (by default) and "union" for unionising node attributes
keep.all.vertices	logical to indicate whether all nodes are kept when intersecting edges. By default, it sets to false
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

an object of class "igraph"

Note

none

See Also

xDefineNet

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
g1 <- xDefineNet(network="KEGG_environmental",
RData.location=RData.location)
g2 <- xDefineNet(network="KEGG_organismal",
RData.location=RData.location)
ls_ig <- list(g1, g2)
ig <- xCombineNet(ls_ig, combineBy='union', attrBy="intersect",
verbose=TRUE)

## End(Not run)
```

xContour *Function to visualise a numeric matrix as a contour plot*

Description

xContour is supposed to visualise a numeric matrix as a contour plot.

Usage

```
xContour(
data,
main = "",
xlab = "",
ylab = "",
key = "",
nlevels = 50,
colormap = c("darkblue-lightblue-lightyellow-darkorange", "bwr", "jet",
"gbr", "wyr",
"br", "yr", "rainbow", "wb"),
highlight = c("none", "min", "max"),
highlight.col = "white",
x.label.cex = 0.95,
x.label.srt = 30,
signature = FALSE,
...
)
```

Arguments

data	a numeric matrix for the contour plot
main	an overall title for the plot
xlab	a title for the x axis. If specified, it will override 'names(dimnames(data))[1]'
ylab	a title for the y axis. If specified, it will override 'names(dimnames(data))[2]'
key	a title for the key plot (on the right)
nlevels	the number of levels to partition the input matrix values. The same level has the same color mapped to
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names

highlight	how to highlight the point. It can be 'none' for no highlight (by default), 'min' for highlighting the point with the minimum value of the matrix, and 'max' for highlighting the point with the maximum value of the matrix
highlight.col	the highlight colors
x.label.cex	the x-axis label size
x.label.srt	the x-axis label angle (in degree from horizontal)
signature	a logical to indicate whether the signature is assigned to the plot caption. By default, it sets FALSE
...	additional graphic parameters. For most parameters, please refer to http://stat.ethz.ch/R-manual/R-devel/library/graphics/html/filled.contour.html

Value

invisible

Note

none

See Also

xContour

Examples

```
x <- y <- seq(-4*pi, 4*pi, len=10)
r <- sqrt(outer(x^2, y^2, "+"))
data <- cos(r^2)*exp(-r/(2*pi))
xContour(data)
#xContour(data, signature=TRUE)
```

---

xConverter *Function to convert an object between graph classes*

Description

xConverter is supposed to convert an object between classes "igraph", "dgCMatrix", "dtree", "lol", and "json".

Usage

```
xConverter(
obj,
from = c("igraph", "dgCMatrix", "dtree", "lol", "json"),
to = c("dgCMatrix", "igraph", "dtree", "lol", "json", "igraph_tree"),
verbose = TRUE
)
```

Arguments

obj	an object of class "igraph", "dgCMatrix", "dtree", "lol", and "json"
from	a character specifying the class converted from. It can be one of "igraph", "dgCMatrix", "dtree", "lol", "json"
to	a character specifying the class converted to. It can be one of "igraph", "dgCMatrix", "dtree", "lol", "json" and "igraph_tree"
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

an object of class "igraph", "dgCMatrix", "dtree", "lol", or "json".

Note

Conversion is supported directly: 1) from 'igraph' to "dgCMatrix","dtree","lol","json","igraph_tree"; 2) from 'dgCMatrix' to "igraph"; 3) from 'dtree' to "igraph","lol","json"; 4) from 'lol' to "dtree","json"; 5) from 'json' to "lol","dtree". In summary: "dgCMatrix" – "igraph" (hub) – "dtree" (hub) – "lol" – "json". Note: 1) igraph –as.igraph– phylo –as.hclust/as.phylo– hclust –as.dendrogram/as.hclust– dendro; 2) igraph –ggraph::den_to_igraph– dendro

See Also

xRDataLoader

Examples

```
# generate a ring graph
g <- make_ring(10, directed=TRUE)

# convert the object from 'igraph' to 'dgCMatrix' class
xConverter(g, from='igraph', to='dgCMatrix')

## Not run:
# Conversion between 'dgCMatrix' and 'igraph'
# ig.EF (an object of class "igraph" storing as a directed graph)
g <- xRDataLoader('ig.EF')
g

# convert the object from 'igraph' to 'dgCMatrix' class
s <- xConverter(g, from='igraph', to='dgCMatrix')
s[1:10,1:10]

# convert the object from 'dgCMatrix' to 'igraph' class
ig <- xConverter(s, from="dgCMatrix", to="igraph")
ig

###############
g <- make_graph("Zachary")
```

```
# from 'igraph' to "dtree","lol","json"
dtree <- xConverter(g, from='igraph', to='dtree')
lol <- xConverter(g, from='igraph', to='lol')
json <- xConverter(g, from='igraph', to='json')

# from "lol","json" to 'dtree'
dtree <- xConverter(lol, from='lol', to='dtree')
dtree <- xConverter(json, from='json', to='dtree')

# from 'dtree' to "igraph"
g <- xConverter(dtree, from='dtree', to='igraph')

# force 'igraph' to a tree
gtree <- xConverter(g, from='igraph', to='igraph_tree')

## End(Not run)
```

`xCorrelation` *Function to calculate and visualise correlation*

Description

`xCorrelation` is supposed to calculate and visualise correlation between a data frame and a named vector (or a list of named vectors).

Usage

```
xCorrelation(
df,
list_vec,
method = c("pearson", "spearman"),
p.type = c("nominal", "empirical"),
seed = 825,
nperm = 2000,
p.adjust.method = c("BH", "BY", "bonferroni", "holm", "hochberg",
"hommel"),
plot = FALSE,
plot.smooth = c(NA, "lm", "loess")
)
```

Arguments

`df`	a data frame with two columns ('name' and 'value')
`list_vec`	a named vector containing numeric values. Alternatively it can be a list of named vectors
`method`	the method used to calcualte correlation. It can be 'pearson' for Pearson's correlation or 'spearman' for Spearman rank correlation

p.type	the type of the p-value calcualted. It can be 'nominal' for nominal p-value or 'empirical' for empirical p-value
seed	an integer specifying the seed
nperm	the number of random permutations
p.adjust.method	the method used to adjust p-values. It can be one of "BH", "BY", "bonferroni", "holm", "hochberg" and "hommel". The first two methods "BH" (widely used) and "BY" control the false discovery rate (FDR: the expected proportion of false discoveries amongst the rejected hypotheses); the last four methods "bonferroni", "holm", "hochberg" and "hommel" are designed to give strong control of the family-wise error rate (FWER). Notes: FDR is a less stringent condition than FWER
plot	logical to indicate whether scatter plot is drawn
plot.smooth	the smooth method for the scatter plot. It can be NA (depending on correlation type), "lm" for the linear line or 'loess' for the loess curve

Value

a list with three componets:

- df_summary: a data frame of n x 5, where n is the number of named vectors, and the 5 columns are "name", "num" (i.e. number of data points used for calculation), "cor" (i.e. correlation), "pval" (i.e. p-value), "fdr"
- ls_gp_curve: NULL if the plot is not drawn; otherwise, a list of 'ggplot' objects for scatter plot together with an estimated curve
- ls_gp_pdf: NULL if the plot is not drawn; otherwise, a list of 'ggplot' objects for pdf plot for null distribution of correlation together with a vertical line for observed correlation

Note

none

See Also

xCorrelation

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# a) provide the seed nodes/genes with the weight info
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
## get genes within 500kb away from AS GWAS lead SNPs
seeds.genes <- ImmunoBase$AS$genes_variants
## seeds weighted according to distance away from lead SNPs
data <- 1- seeds.genes/500000
```

```
# b) prepare a data frame
df <- data.frame(name=names(data), value=data, stringsAsFactors=FALSE)

# c) do correlation
ls_res <- xCorrelation(df, data, method="pearson", p.type="empirical",
nperm=2000, plot=TRUE)

## End(Not run)
```

---

xDAGanno *Function to generate a subgraph of a direct acyclic graph (DAG) induced by the input annotation data*

Description

xDAGanno is supposed to produce a subgraph induced by the input annotation data, given a direct acyclic graph (DAG; an ontology). The input is a graph of "igraph", a list of the vertices containing annotation data, and the mode defining the paths to the root of DAG. The induced subgraph contains vertices (with annotation data) and their ancestors along with the defined paths to the root of DAG. The annotations at these vertices (including their ancestors) can also be updated according to the true-path rule: those annotated to a term should also be annotated by its all ancestor terms.

Usage

```
xDAGanno(
g,
annotation,
path.mode = c("all_paths", "shortest_paths", "all_shortest_paths"),
true.path.rule = TRUE,
verbose = TRUE
)
```

Arguments

g	an object of class "igraph" to represent DAG
annotation	the vertices/nodes for which annotation data are provided. It can be a sparse Matrix of class "dgCMatrix" (with variants/genes as rows and terms as columns), or a list of nodes/terms each containing annotation data, or an object of class 'GS' (basically a list for each node/term with annotation data)
path.mode	the mode of paths induced by vertices/nodes with input annotation data. It can be "all_paths" for all possible paths to the root, "shortest_paths" for only one path to the root (for each node in query), "all_shortest_paths" for all shortest paths to the root (i.e. for each node, find all shortest paths with the equal lengths)
true.path.rule	logical to indicate whether the true-path rule should be applied to propagate annotations. By default, it sets to true
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

- subg: an induced subgraph, an object of class "igraph". In addition to the original attributes to nodes and edges, the return subgraph is also appended by two node attributes: 1) "anno" containing a list of variants/genes either as original annotations (and inherited annotations; 2) "IC" standing for information content defined as negative 10-based log-transformed frequency of variants/genes annotated to that term.

Note

For the mode "shortest_paths", the induced subgraph is the most concise, and thus informative for visualisation when there are many nodes in query, while the mode "all_paths" results in the complete subgraph.

See Also

xDAGanno

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# 1) SNP-based ontology
# 1a) ig.EF (an object of class "igraph" storing as a directed graph)
g <- xRDataLoader('ig.EF')

# 1b) load GWAS SNPs annotated by EF (an object of class "dgCMatrix" storing a spare matrix)
anno <- xRDataLoader(RData='GWAS2EF')

# 1c) prepare for annotation data
# randomly select 5 terms/vertices (and their annotation data)
annotation <- anno[, sample(1:dim(anno)[2],5)]

# 1d) obtain the induced subgraph according to the input annotation data
# based on shortest paths (i.e. the most concise subgraph induced)
dag <- xDAGanno(g, annotation, path.mode="shortest_paths",
verbose=TRUE)

# 1e) color-code nodes/terms according to the number of annotations
data <- sapply(V(dag)$anno, length)
names(data) <- V(dag)$name
dnet::visDAG(g=dag, data=data, node.info="both")

####################
# Below is for those SNPs annotated by the term called 'ankylosing spondylitis'
# The steps 1a) and 1b) are the same as above
# 1c') prepare for annotation data
# select a term 'ankylosing spondylitis'
terms <- V(g)$term_id[grep('ankylosing spondylitis',V(g)$term_name,
perl=TRUE)]
ind <- which(colnames(anno) %in% terms)
annotation <- lapply(ind, function(x){names(which(anno[,x]!=0))})
```

```
names(annotation) <- colnames(anno)[ind]

# 1d') obtain the induced subgraph according to the input annotation data
# based on all possible paths (i.e. the complete subgraph induced)
dag <- xDAGanno(g, annotation, path.mode="all_paths", verbose=TRUE)

# 1e') color-code nodes/terms according to the number of annotations
data <- sapply(V(dag)$anno, length)
names(data) <- V(dag)$name
dnet::visDAG(g=dag, data=data, node.info="both")

############################################################
# 2) Gene-based ontology
# 2a) ig.MP (an object of class "igraph" storing as a directed graph)
g <- xRDataLoader('ig.MP')

# 2b) load human genes annotated by MP (an object of class "GS" containing the 'gs' component)
GS <- xRDataLoader(RData='org.Hs.egMP')
anno <- GS$gs # notes: This is a list

# 2c) prepare for annotation data
# randomly select 5 terms/vertices (and their annotation data)
annotation <- anno[sample(1:length(anno),5)]

# 2d) obtain the induced subgraph according to the input annotation data
# based on shortest paths (i.e. the most concise subgraph induced)
# but without applying true-path rule
dag <- xDAGanno(g, annotation, path.mode="shortest_paths",
true.path.rule=TRUE, verbose=TRUE)

# 2e) color-code nodes/terms according to the number of annotations
data <- sapply(V(dag)$anno, length)
names(data) <- V(dag)$name
dnet::visDAG(g=dag, data=data, node.info="both")

## End(Not run)
```

`xDefineEQTL` *Function to extract eQTL-gene pairs given a list of SNPs or a customised eQTL mapping data*

Description

`xDefineEQTL` is supposed to extract eQTL-gene pairs given a list of SNPs or a customised eQTL mapping data.

Usage

```
xDefineEQTL(
data = NULL,
```

```
include.eQTL = c(NA, "JKscience_CD14", "JKscience_LPS2",
"JKscience_LPS24",
"JKscience_IFN", "JKscience_TS2A", "JKscience_TS2A_CD14",
"JKscience_TS2A_LPS2",
"JKscience_TS2A_LPS24", "JKscience_TS2A_IFN", "JKscience_TS2B",
"JKscience_TS2B_CD14", "JKscience_TS2B_LPS2", "JKscience_TS2B_LPS24",
"JKscience_TS2B_IFN", "JKscience_TS3A", "JKng_bcell", "JKng_bcell_cis",
"JKng_bcell_trans", "JKng_mono", "JKng_mono_cis", "JKng_mono_trans",
"JKpg_CD4",
"JKpg_CD4_cis", "JKpg_CD4_trans", "JKpg_CD8", "JKpg_CD8_cis",
"JKpg_CD8_trans",
"JKnc_neutro", "JKnc_neutro_cis", "JKnc_neutro_trans",
"WESTRAng_blood",
"WESTRAng_blood_cis", "WESTRAng_blood_trans", "JK_nk", "JK_nk_cis",
"JK_nk_trans",
"GTEx_V4_Adipose_Subcutaneous", "GTEx_V4_Artery_Aorta",
"GTEx_V4_Artery_Tibial",
"GTEx_V4_Esophagus_Mucosa", "GTEx_V4_Esophagus_Muscularis",
"GTEx_V4_Heart_Left_Ventricle", "GTEx_V4_Lung",
"GTEx_V4_Muscle_Skeletal",
"GTEx_V4_Nerve_Tibial", "GTEx_V4_Skin_Sun_Exposed_Lower_leg",
"GTEx_V4_Stomach",
"GTEx_V4_Thyroid", "GTEx_V4_Whole_Blood",
"GTEx_V6p_Adipose_Subcutaneous",
"GTEx_V6p_Adipose_Visceral_Omentum", "GTEx_V6p_Adrenal_Gland",
"GTEx_V6p_Artery_Aorta", "GTEx_V6p_Artery_Coronary",
"GTEx_V6p_Artery_Tibial",
"GTEx_V6p_Brain_Anterior_cingulate_cortex_BA24",
"GTEx_V6p_Brain_Caudate_basal_ganglia",
"GTEx_V6p_Brain_Cerebellar_Hemisphere",
"GTEx_V6p_Brain_Cerebellum", "GTEx_V6p_Brain_Cortex",
"GTEx_V6p_Brain_Frontal_Cortex_BA9", "GTEx_V6p_Brain_Hippocampus",
"GTEx_V6p_Brain_Hypothalamus",
"GTEx_V6p_Brain_Nucleus_accumbens_basal_ganglia",
"GTEx_V6p_Brain_Putamen_basal_ganglia",
"GTEx_V6p_Breast_Mammary_Tissue",
"GTEx_V6p_Cells_EBVtransformed_lymphocytes",
"GTEx_V6p_Cells_Transformed_fibroblasts", "GTEx_V6p_Colon_Sigmoid",
"GTEx_V6p_Colon_Transverse",
"GTEx_V6p_Esophagus_Gastroesophageal_Junction",
"GTEx_V6p_Esophagus_Mucosa", "GTEx_V6p_Esophagus_Muscularis",
"GTEx_V6p_Heart_Atrial_Appendage", "GTEx_V6p_Heart_Left_Ventricle",
"GTEx_V6p_Liver",
"GTEx_V6p_Lung", "GTEx_V6p_Muscle_Skeletal", "GTEx_V6p_Nerve_Tibial",
"GTEx_V6p_Ovary", "GTEx_V6p_Pancreas", "GTEx_V6p_Pituitary",
"GTEx_V6p_Prostate", "GTEx_V6p_Skin_Not_Sun_Exposed_Suprapubic",
"GTEx_V6p_Skin_Sun_Exposed_Lower_leg",
"GTEx_V6p_Small_Intestine_Terminal_Ileum",
```

```
"GTEx_V6p_Spleen", "GTEx_V6p_Stomach", "GTEx_V6p_Testis",
"GTEx_V6p_Thyroid",
"GTEx_V6p_Uterus", "GTEx_V6p_Vagina", "GTEx_V6p_Whole_Blood",
"eQTLGen",
"eQTLGen_cis", "eQTLGen_trans", "scRNAseq_eQTL_Bcell",
"scRNAseq_eQTL_CD4",
"scRNAseq_eQTL_CD8", "scRNAseq_eQTL_cMono", "scRNAseq_eQTL_DC",
"scRNAseq_eQTL_Mono",
"scRNAseq_eQTL_ncMono", "scRNAseq_eQTL_NK", "scRNAseq_eQTL_PBMC",
"jpRNAseq_eQTL_Bcell", "jpRNAseq_eQTL_CD4", "jpRNAseq_eQTL_CD8",
"jpRNAseq_eQTL_Mono", "jpRNAseq_eQTL_NK", "jpRNAseq_eQTL_PBMC",
"Pi_eQTL_Bcell",
"Pi_eQTL_Blood", "Pi_eQTL_CD14", "Pi_eQTL_CD4", "Pi_eQTL_CD8",
"Pi_eQTL_IFN",
"Pi_eQTL_LPS2", "Pi_eQTL_LPS24", "Pi_eQTL_Monocyte",
"Pi_eQTL_Neutrophil",
"Pi_eQTL_NK", "Pi_eQTL_shared_CD14", "Pi_eQTL_shared_IFN",
"Pi_eQTL_shared_LPS2",
"Pi_eQTL_shared_LPS24", "Osteoblast_eQTL"),
eQTL.customised = NULL,
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

data	NULL or an input vector containing SNPs. If NULL, all SNPs will be considered. If a input vector containing SNPs, SNPs should be provided as dbSNP ID (ie starting with rs). Alternatively, they can be in the format of 'chrN:xxx', where N is either 1-22 or X, xxx is number; for example, 'chr16:28525386'
include.eQTL	genes modulated by eQTL (also Lead SNPs or in LD with Lead SNPs) are also included. By default, it is 'NA' to disable this option. Otherwise, those genes modulated by eQTL will be included. Pre-built eQTL datasets are detailed in the section 'Note'
eQTL.customised	a user-input matrix or data frame with 4 columns: 1st column for SNPs/eQTLs, 2nd column for Genes, 3rd for eQTL mapping significance level (p-values or FDR), and 4th for contexts (required even though only one context is input). Alternatively, it can be a file containing these 4 columns. It is designed to allow the user analysing their eQTL data. This customisation (if provided) will populate built-in eQTL data; mysql -e "use pi; SELECT rs_id_dbSNP147_GRCh37p13,gene_name,pval_nominal,T FROM GTEx_V7_pair WHERE rs_id_dbSNP147_GRCh37p13!='.';" > /var/www/bigdata/eQTL.custom
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details

guid — a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details

Value

a data frame with following columns:

- SNP: eQTLs
- Gene: eQTL-containing genes
- Sig: the eQTL mapping significant level
- Context: the context in which eQTL data was generated

Note

Pre-built eQTL datasets are described below according to the data sources.
1. Context-specific eQTLs in monocytes: resting and activating states. Sourced from Science 2014, 343(6175):1246949

- JKscience_TS2A: cis-eQTLs in either state (based on 228 individuals with expression data available for all experimental conditions).
- JKscience_TS2A_CD14: cis-eQTLs only in the resting/CD14+ state (based on 228 individuals).
- JKscience_TS2A_LPS2: cis-eQTLs only in the activating state induced by 2-hour LPS (based on 228 individuals).
- JKscience_TS2A_LPS24: cis-eQTLs only in the activating state induced by 24-hour LPS (based on 228 individuals).
- JKscience_TS2A_IFN: cis-eQTLs only in the activating state induced by 24-hour interferon-gamma (based on 228 individuals).
- JKscience_TS2B: cis-eQTLs in either state (based on 432 individuals).
- JKscience_TS2B_CD14: cis-eQTLs only in the resting/CD14+ state (based on 432 individuals).
- JKscience_TS2B_LPS2: cis-eQTLs only in the activating state induced by 2-hour LPS (based on 432 individuals).
- JKscience_TS2B_LPS24: cis-eQTLs only in the activating state induced by 24-hour LPS (based on 432 individuals).
- JKscience_TS2B_IFN: cis-eQTLs only in the activating state induced by 24-hour interferon-gamma (based on 432 individuals).
- JKscience_TS3A: trans-eQTLs in either state.
- JKscience_CD14: cis and trans-eQTLs in the resting/CD14+ state (based on 228 individuals).
- JKscience_LPS2: cis and trans-eQTLs in the activating state induced by 2-hour LPS (based on 228 individuals).
- JKscience_LPS24: cis and trans-eQTLs in the activating state induced by 24-hour LPS (based on 228 individuals).
- JKscience_IFN: cis and trans-eQTLs in the activating state induced by 24-hour interferon-gamma (based on 228 individuals).

2. eQTLs in B cells. Sourced from Nature Genetics 2012, 44(5):502-510

- JKng_bcell: cis- and trans-eQTLs.
- JKng_bcell_cis: cis-eQTLs only.
- JKng_bcell_trans: trans-eQTLs only.

3. eQTLs in monocytes. Sourced from Nature Genetics 2012, 44(5):502-510

- JKng_mono: cis- and trans-eQTLs.
- JKng_mono_cis: cis-eQTLs only.
- JKng_mono_trans: trans-eQTLs only.

4. eQTLs in neutrophils. Sourced from Nature Communications 2015, 7(6):7545

- JKnc_neutro: cis- and trans-eQTLs.
- JKnc_neutro_cis: cis-eQTLs only.
- JKnc_neutro_trans: trans-eQTLs only.

5. eQTLs in NK cells. Unpublished (restricted access)

- JK_nk: cis- and trans-eQTLs.
- JK_nk_cis: cis-eQTLs only.
- JK_nk_trans: trans-eQTLs only.

6. Tissue-specific eQTLs from GTEx (version 4; including 13 tissues). Sourced from Science 2015, 348(6235):648-60

- GTEx_V4_Adipose_Subcutaneous: cis-eQTLs in tissue 'Adipose Subcutaneous'.
- GTEx_V4_Artery_Aorta: cis-eQTLs in tissue 'Artery Aorta'.
- GTEx_V4_Artery_Tibial: cis-eQTLs in tissue 'Artery Tibial'.
- GTEx_V4_Esophagus_Mucosa: cis-eQTLs in tissue 'Esophagus Mucosa'.
- GTEx_V4_Esophagus_Muscularis: cis-eQTLs in tissue 'Esophagus Muscularis'.
- GTEx_V4_Heart_Left_Ventricle: cis-eQTLs in tissue 'Heart Left Ventricle'.
- GTEx_V4_Lung: cis-eQTLs in tissue 'Lung'.
- GTEx_V4_Muscle_Skeletal: cis-eQTLs in tissue 'Muscle Skeletal'.
- GTEx_V4_Nerve_Tibial: cis-eQTLs in tissue 'Nerve Tibial'.
- GTEx_V4_Skin_Sun_Exposed_Lower_leg: cis-eQTLs in tissue 'Skin Sun Exposed Lower leg'.
- GTEx_V4_Stomach: cis-eQTLs in tissue 'Stomach'.
- GTEx_V4_Thyroid: cis-eQTLs in tissue 'Thyroid'.
- GTEx_V4_Whole_Blood: cis-eQTLs in tissue 'Whole Blood'.

7. eQTLs in CD4 T cells. Sourced from PLoS Genetics 2017, 13(3):e1006643

- JKpg_CD4: cis- and trans-eQTLs.
- JKpg_CD4_cis: cis-eQTLs only.

- JKpg_CD4_trans: trans-eQTLs only.

8. eQTLs in CD8 T cells. Sourced from PLoS Genetics 2017, 13(3):e1006643

   - JKpg_CD8: cis- and trans-eQTLs.
   - JKpg_CD8_cis: cis-eQTLs only.
   - JKpg_CD8_trans: trans-eQTLs only.

9. eQTLs in blood. Sourced from Nature Genetics 2013, 45(10):1238-1243

   - WESTRAng_blood: cis- and trans-eQTLs.
   - WESTRAng_blood_cis: cis-eQTLs only.
   - WESTRAng_blood_trans: trans-eQTLs only.

10. Tissue-specific eQTLs from GTEx (version 6p; including 44 tissues). Sourced from http://www.biorxiv.org/content/early/

    - GTEx_V6p_Adipose_Subcutaneous: cis-eQTLs in tissue "Adipose Subcutaneous".
    - GTEx_V6p_Adipose_Visceral_Omentum: cis-eQTLs in tissue "Adipose Visceral (Omentum)".
    - GTEx_V6p_Adrenal_Gland: cis-eQTLs in tissue "Adrenal Gland".
    - GTEx_V6p_Artery_Aorta: cis-eQTLs in tissue "Artery Aorta".
    - GTEx_V6p_Artery_Coronary: cis-eQTLs in tissue "Artery Coronary".
    - GTEx_V6p_Artery_Tibial: cis-eQTLs in tissue "Artery Tibial".
    - GTEx_V6p_Brain_Anterior_cingulate_cortex_BA24: cis-eQTLs in tissue "Brain Anterior cingulate cortex (BA24)".
    - GTEx_V6p_Brain_Caudate_basal_ganglia: cis-eQTLs in tissue "Brain Caudate (basal ganglia)".
    - GTEx_V6p_Brain_Cerebellar_Hemisphere: cis-eQTLs in tissue "Brain Cerebellar Hemisphere".
    - GTEx_V6p_Brain_Cerebellum: cis-eQTLs in tissue "Brain Cerebellum".
    - GTEx_V6p_Brain_Cortex: cis-eQTLs in tissue "Brain Cortex".
    - GTEx_V6p_Brain_Frontal_Cortex_BA9: cis-eQTLs in tissue "Brain Frontal Cortex (BA9)".
    - GTEx_V6p_Brain_Hippocampus: cis-eQTLs in tissue "Brain Hippocampus".
    - GTEx_V6p_Brain_Hypothalamus: cis-eQTLs in tissue "Brain Hypothalamus".
    - GTEx_V6p_Brain_Nucleus_accumbens_basal_ganglia: cis-eQTLs in tissue "Brain Nucleus accumbens (basal ganglia)".
    - GTEx_V6p_Brain_Putamen_basal_ganglia: cis-eQTLs in tissue "Brain Putamen (basal ganglia)".
    - GTEx_V6p_Breast_Mammary_Tissue: cis-eQTLs in tissue "Breast Mammary Tissue".
    - GTEx_V6p_Cells_EBVtransformed_lymphocytes: cis-eQTLs in tissue "Cells EBV-transformed lymphocytes".
    - GTEx_V6p_Cells_Transformed_fibroblasts: cis-eQTLs in tissue "Cells Transformed fibroblasts".
    - GTEx_V6p_Colon_Sigmoid: cis-eQTLs in tissue "Colon Sigmoid".

- GTEx_V6p_Colon_Transverse: cis-eQTLs in tissue "Colon Transverse".
- GTEx_V6p_Esophagus_Gastroesophageal_Junction: cis-eQTLs in tissue "Esophagus Gastroesophageal Junction".
- GTEx_V6p_Esophagus_Mucosa: cis-eQTLs in tissue "Esophagus Mucosa".
- GTEx_V6p_Esophagus_Muscularis: cis-eQTLs in tissue "Esophagus Muscularis".
- GTEx_V6p_Heart_Atrial_Appendage: cis-eQTLs in tissue "Heart Atrial Appendage".
- GTEx_V6p_Heart_Left_Ventricle: cis-eQTLs in tissue "Heart Left Ventricle".
- GTEx_V6p_Liver: cis-eQTLs in tissue "Liver".
- GTEx_V6p_Lung: cis-eQTLs in tissue "Lung".
- GTEx_V6p_Muscle_Skeletal: cis-eQTLs in tissue "Muscle Skeletal".
- GTEx_V6p_Nerve_Tibial: cis-eQTLs in tissue "Nerve Tibial".
- GTEx_V6p_Ovary: cis-eQTLs in tissue "Ovary".
- GTEx_V6p_Pancreas: cis-eQTLs in tissue "Pancreas".
- GTEx_V6p_Pituitary: cis-eQTLs in tissue "Pituitary".
- GTEx_V6p_Prostate: cis-eQTLs in tissue "Prostate".
- GTEx_V6p_Skin_Not_Sun_Exposed_Suprapubic: cis-eQTLs in tissue "Skin Not Sun Exposed (Suprapubic)".
- GTEx_V6p_Skin_Sun_Exposed_Lower_leg: cis-eQTLs in tissue "Skin Sun Exposed (Lower leg)".
- GTEx_V6p_Small_Intestine_Terminal_Ileum: cis-eQTLs in tissue "Small Intestine Terminal Ileum".
- GTEx_V6p_Spleen: cis-eQTLs in tissue "Spleen".
- GTEx_V6p_Stomach: cis-eQTLs in tissue "Stomach".
- GTEx_V6p_Testis: cis-eQTLs in tissue "Testis".
- GTEx_V6p_Thyroid: cis-eQTLs in tissue "Thyroid".
- GTEx_V6p_Uterus: cis-eQTLs in tissue "Uterus".
- GTEx_V6p_Vagina: cis-eQTLs in tissue "Vagina".
- GTEx_V6p_Whole_Blood: cis-eQTLs in tissue "Whole Blood".

11. eQTLs in eQTLGen. Sourced from bioRxiv, 2018, doi:10.1101/447367

- eQTLGen: cis- and trans-eQTLs.
- eQTLGen_cis: cis-eQTLs only.
- eQTLGen_trans: trans-eQTLs only.

12. Single-cell-RNA-identified celltype-specific cis-eQTLs (including 9 cell types). Sourced from Nature Genetics 2018, 50(4):493-497

- scRNAseq_eQTL_Bcell: cis-eQTLs in B cells.
- scRNAseq_eQTL_CD4: cis-eQTLs in CD4+ T cells.
- scRNAseq_eQTL_CD8: cis-eQTLs in CD8+ T cells.

- scRNAseq_eQTL_DC: cis-eQTLs in dendritic cells.
- scRNAseq_eQTL_cMono: cis-eQTLs in classical monocytes.
- scRNAseq_eQTL_ncMono: cis-eQTLs in nonclassical monocytes.
- scRNAseq_eQTL_Mono: cis-eQTLs in monocytes.
- scRNAseq_eQTL_NK: cis-eQTLs in NK cells.
- scRNAseq_eQTL_PBMC: cis-eQTLs in PBMC.

13. Japanese celltype-specific cis-eQTLs (including 6 cell types). Sourced from Nature Genetics 2017, 49(7):1120-1125

- jpRNAseq_eQTL_Bcell: cis-eQTLs in B cells.
- jpRNAseq_eQTL_CD4: cis-eQTLs in CD4+ T cells.
- jpRNAseq_eQTL_CD8: cis-eQTLs in CD8+ T cells.
- jpRNAseq_eQTL_Mono: cis-eQTLs in monocytes.
- jpRNAseq_eQTL_NK: cis-eQTLs in NK cells.
- jpRNAseq_eQTL_PBMC: cis-eQTLs in PBMC.

14. Pi eQTL

- Pi_eQTL_CD14: cis and trans-eQTLs in the resting/CD14+ state.
- Pi_eQTL_LPS2: cis and trans-eQTLs in the activating state induced by 2-hour LPS.
- Pi_eQTL_LPS24: cis and trans-eQTLs in the activating state induced by 24-hour LPS.
- Pi_eQTL_IFN: cis and trans-eQTLs in the activating state induced by 24-hour interferon-gamma.
- Pi_eQTL_Bcell: cis and trans-eQTLs in B cells.
- Pi_eQTL_Blood: cis and trans-eQTLs in the blood.
- Pi_eQTL_CD4: cis and trans-eQTLs in the CD4 cells.
- Pi_eQTL_CD8: cis and trans-eQTLs in the CD8 cells.
- Pi_eQTL_Monocyte: cis and trans-eQTLs in the monocytes.
- Pi_eQTL_Neutrophil: cis and trans-eQTLs in the neutrophils.
- Pi_eQTL_NK: cis and trans-eQTLs in the NK cells.
- Pi_eQTL_shared_CD14: cis and trans-eQTLs in the resting/CD14+ state (based on 228 individuals).
- Pi_eQTL_shared_LPS2: cis and trans-eQTLs in the activating state induced by 2-hour LPS (based on 228 individuals).
- Pi_eQTL_shared_LPS24: cis and trans-eQTLs in the activating state induced by 24-hour LPS (based on 228 individuals).
- Pi_eQTL_shared_IFN: cis and trans-eQTLs in the activating state induced by 24-hour interferon-gamma (based on 228 individuals).

15. Osteoblast cis-eQTLs. Sourced from Genome Research 2009, 19(11):1942-52

- Osteoblast_eQTL: cis-eQTLs in Osteoblast.

See Also

xSNPlocations, xGR, xRDataLoader

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# a) provide the SNPs with the significance info
data(ImmunoBase)
gr <- ImmunoBase$AS$variants
data <- gr$Variant

# b) define eQTL genes
df_SGS <- xDefineEQTL(data, include.eQTL="JKscience_TS2A",
RData.location=RData.location)

## End(Not run)
```

---

xDefineHIC *Function to extract promoter capture HiC-gene pairs given a list of SNPs*

Description

xDefineHIC is supposed to extract HiC-gene pairs given a list of SNPs.

Usage

```
xDefineHIC(
data = NULL,
entity = c("SNP", "chr:start-end", "data.frame", "bed", "GRanges"),
include.HiC = c(NA, "Monocytes", "Macrophages_M0", "Macrophages_M1",
"Macrophages_M2", "Neutrophils", "Megakaryocytes",
"Endothelial_precursors",
"Erythroblasts", "Fetal_thymus", "Naive_CD4_T_cells",
"Total_CD4_T_cells",
"Activated_total_CD4_T_cells", "Nonactivated_total_CD4_T_cells",
"Naive_CD8_T_cells",
"Total_CD8_T_cells", "Naive_B_cells", "Total_B_cells", "PE.Monocytes",
"PE.Macrophages_M0", "PE.Macrophages_M1", "PE.Macrophages_M2",
"PE.Neutrophils",
"PE.Megakaryocytes", "PE.Erythroblasts", "PE.Naive_CD4_T_cells",
"PE.Naive_CD8_T_cells", "Combined", "Combined_PE"),
GR.SNP = c("dbSNP_GWAS", "dbSNP_Common", "dbSNP_Single"),
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

data	NULL or an input vector containing SNPs. If NULL, all SNPs will be considered. If a input vector containing SNPs, SNPs should be provided as dbSNP ID (ie starting with rs) or in the format of 'chrN:xxx', where N is either 1-22 or X, xxx is number; for example, 'chr16:28525386'. Alternatively, it can be other formats/entities (see the next parameter 'entity')
entity	the data entity. By default, it is "SNP". For general use, it can also be one of "chr:start-end", "data.frame", "bed" or "GRanges"
include.HiC	genes linked to input SNPs are also included. By default, it is 'NA' to disable this option. Otherwise, those genes linked to SNPs will be included according to Promoter Capture HiC (PCHiC) datasets. Pre-built HiC datasets are detailed in the section 'Note'
GR.SNP	the genomic regions of SNPs. By default, it is 'dbSNP_GWAS', that is, SNPs from dbSNP (version 146) restricted to GWAS SNPs and their LD SNPs (hg19). It can be 'dbSNP_Common', that is, Common SNPs from dbSNP (version 146) plus GWAS SNPs and their LD SNPs (hg19). Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to dbSNP IDs. Then, tell "GR.SNP" with your RData file name (with or without extension), plus specify your file RData path in "RData.location". Note: you can also load your customised GR object directly
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details
guid	a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details

Value

If input data is NULL, a data frame with following columns:

- from: baited genomic regions (baits)
- to: preyed (other end) genomic regions of interactions (preys)
- score: CHiCAGO scores quantifying the strength of physical interactions between harbors and partners

If input data is not NULL, a list with two components: "df" and "ig". "df" is a data frame with following columns:

- from: 'from/bait' genomic regions
- to: 'to/prey' genomic regions
- score: CHiCAGO scores quantifying the strength of physical interactions between baits and preys
- from_genes: genes associated with 'from/bait' genomic regions
- to_genes: genes associated with 'to/prey' genomic regions

- SNP: input SNPs (in query)
- SNP_end: specify which end SNPs in query fall into (either 'bait/from' or 'prey/to')
- SNP_harbor: genomic regions harbors the SNPs in query
- Context: the context in which PCHiC data was generated

"ig" is an object of both classes "igraph" and "PCHiC", a direct graph with nodes for genomic regions and edges for CHiCAGO scores between them. Also added node attribute is 1) 'target' storing genes assocated and 2) 'SNP' for input SNPs (if the node harboring input SNPs). If several cell types are queried, "ig" is actually a list of "igraph"/"PCHiC" objects.

Note

Pre-built HiC datasets are described below according to the data sources.
1. Promoter Capture HiC datasets in 17 primary blood cell types. Sourced from Cell 2016, 167(5):1369-1384.e19

- Monocytes: physical interactions (CHiCAGO score >=5) of promoters (baits) with the other end (preys) in Monocytes.
- Macrophages_M0: promoter interactomes in Macrophages M0.
- Macrophages_M1: promoter interactomes in Macrophages M1.
- Macrophages_M2: promoter interactomes in Macrophages M2.
- Neutrophils: promoter interactomes in Neutrophils.
- Megakaryocytes: promoter interactomes in Megakaryocytes.
- Endothelial_precursors: promoter interactomes in Endothelial precursors.
- Erythroblasts: promoter interactomes in Erythroblasts.
- Fetal_thymus: promoter interactomes in Fetal thymus.
- Naive_CD4_T_cells: promoter interactomes in Naive CD4+ T cells.
- Total_CD4_T_cells: promoter interactomes in Total CD4+ T cells.
- Activated_total_CD4_T_cells: promoter interactomes in Activated total CD4+ T cells.
- Nonactivated_total_CD4_T_cells: promoter interactomes in Nonactivated total CD4+ T cells.
- Naive_CD8_T_cells: promoter interactomes in Naive CD8+ T cells.
- Total_CD8_T_cells: promoter interactomes in Total CD8+ T cells.
- Naive_B_cells: promoter interactomes in Naive B cells.
- Total_B_cells: promoter interactomes in Total B cells.
- Combined: promoter interactomes combined above; with score for the number of significant cell types plus scaled average.

2. Promoter Capture HiC datasets (involving active promoters and enhancers) in 9 primary blood cell types. Sourced from Cell 2016, 167(5):1369-1384.e19

- PE.Monocytes: physical interactions (CHiCAGO score >=5) of promoters (baits) with the other end (enhancers as preys) in Monocytes.
- PE.Macrophages_M0: promoter-enhancer interactomes in Macrophages M0.

- PE.Macrophages_M1: promoter-enhancer interactomes in Macrophages M1.
- PE.Macrophages_M2: promoter-enhancer interactomes in Macrophages M2.
- PE.Neutrophils: promoter-enhancer interactomes in Neutrophils.
- PE.Megakaryocytes: promoter-enhancer interactomes in Megakaryocytes.
- PE.Erythroblasts: promoter-enhancer interactomes in Erythroblasts.
- PE.Naive_CD4_T_cells: promoter-enhancer interactomes in Naive CD4+ T cells.
- PE.Naive_CD8_T_cells: promoter-enhancer interactomes in Naive CD8+ T cells.
- Combined_PE: promoter interactomes combined above; with score for the number of significant cell types plus scaled average.

See Also

xRDataLoader, xAggregate

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# a) provide the SNPs with the significance info
data(ImmunoBase)
data <- names(ImmunoBase$AS$variants)

# b) extract HiC-gene pairs given a list of AS SNPs
PCHiC <- xDefineHIC(data, include.HiC="Monocytes", GR.SNP="dbSNP_GWAS",
RData.location=RData.location)
head(PCHiC$df)

# c) visualise the interaction (a directed graph: bait->prey)
g <- PCHiC$ig
## a node with SNPs colored in 'skyblue' and the one without SNPs in 'pink'
## the width in an edge is proportional to the interaction strength
xPCHiCplot(g, vertex.label.cex=0.5)
xPCHiCplot(g, glayout=layout_in_circle, vertex.label.cex=0.5)

## End(Not run)
```

---

xDefineNet *Function to define a gene network*

Description

xDefineNet is supposed to define a gene network sourced from the STRING database or the Pathway Commons database. It returns an object of class "igraph".

Usage

```
xDefineNet(
network = c("STRING_highest", "STRING_high", "STRING_medium",
"STRING_low",
"PCommonsUN_high", "PCommonsUN_medium", "PCommonsDN_high",
"PCommonsDN_medium",
"PCommonsDN_Reactome", "PCommonsDN_KEGG", "PCommonsDN_HumanCyc",
"PCommonsDN_PID",
"PCommonsDN_PANTHER", "PCommonsDN_ReconX", "PCommonsDN_TRANSFAC",
"PCommonsDN_PhosphoSite", "PCommonsDN_CTD", "KEGG", "KEGG_metabolism",
"KEGG_genetic", "KEGG_environmental", "KEGG_cellular",
"KEGG_organismal",
"KEGG_disease", "REACTOME", "TRRUST"),
STRING.only = c(NA, "neighborhood_score", "fusion_score",
"cooccurence_score",
"coexpression_score", "experimental_score", "database_score",
"textmining_score")[1],
weighted = FALSE,
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

network	the built-in network. Currently two sources of network information are supported: the STRING database (version 10) and the Pathway Commons database (version 7). STRING is a meta-integration of undirect interactions from the functional aspect, while Pathways Commons mainly contains both undirect and direct interactions from the physical/pathway aspect. Both have scores to control the confidence of interactions. Therefore, the user can choose the different quality of the interactions. In STRING, "STRING_highest" indicates interactions with highest confidence (confidence scores>=900), "STRING_high" for interactions with high confidence (confidence scores>=700), "STRING_medium" for interactions with medium confidence (confidence scores>=400), and "STRING_low" for interactions with low confidence (confidence scores>=150). For undirect/physical interactions from Pathways Commons, "PCommonsUN_high" indicates undirect interactions with high confidence (supported with the PubMed references plus at least 2 different sources), "PCommonsUN_medium" for undirect interactions with medium confidence (supported with the PubMed references). For direct (pathway-merged) interactions from Pathways Commons, "PCommonsDN_high" indicates direct interactions with high confidence (supported with the PubMed references plus at least 2 different sources), and "PCommonsUN_medium" for direct interactions with medium confidence (supported with the PubMed references). In addition to pooled version of pathways from all data sources, the user can also choose the pathway-merged network from individual sources, that is, "PCommonsDN_Reactome" for those from Reactome, "PCommonsDN_KEGG" for those from KEGG, "PCommonsDN_HumanCyc"

	for those from HumanCyc, "PCommonsDN_PID" for those froom PID, "PCommonsDN_PANTHER" for those from PANTHER, "PCommonsDN_ReconX" for those from ReconX, "PCommonsDN_TRANSFAC" for those from TRANSFAC, "PCommonsDN_PhosphoSite" for those from PhosphoSite, and "PCommonsDN_CTD" for those from CTD. For direct (pathway-merged) interactions sourced from KEGG, it can be 'KEGG' for all, 'KEGG_metabolism' for pathways grouped into 'Metabolism', 'KEGG_genetic' for 'Genetic Information Processing' pathways, 'KEGG_environmental' for 'Environmental Information Processing' pathways, 'KEGG_cellular' for 'Cellular Processes' pathways, 'KEGG_organismal' for 'Organismal Systems' pathways, and 'KEGG_disease' for 'Human Diseases' pathways. 'REACTOME' for protein-protein interactions derived from Reactome pathways. 'TRRUST' for TRRUST curated TF-target relations
STRING.only	the further restriction of STRING by interaction type. If NA, no such restriction. Otherwide, it can be one or more of "neighborhood_score","fusion_score","cooccurence_score","coexpres Useful options are c("experimental_score","database_score"): only experimental data (extracted from BIND, DIP, GRID, HPRD, IntAct, MINT, and PID) and curated data (extracted from Biocarta, BioCyc, GO, KEGG, and Reactome) are used
weighted	logical to indicate whether edge weights should be considered. By default, it sets to false. If true, it only works for the network from the STRING database
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details
guid	a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details

Value

an object of class "igraph"

Note

The input graph will treat as an unweighted graph if there is no 'weight' edge attribute associated with

See Also

xRDataLoader

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# STRING (high quality)
g <- xDefineNet(network="STRING_high", RData.location=RData.location)
# STRING (high quality), with edges weighted
g <- xDefineNet(network="STRING_high", weighted=T,
```

```
RData.location=RData.location)
# STRING (high quality), only edges sourced from experimental or curated data
g <- xDefineNet(network="STRING_high",
STRING.only=c("experimental_score","database_score"),
RData.location=RData.location)

# Pathway Commons
g <- xDefineNet(network="PCommonsDN_medium",
RData.location=RData.location)

# KEGG (all)
g <- xDefineNet(network="KEGG", RData.location=RData.location)
# KEGG ('Organismal Systems')
g <- xDefineNet(network="KEGG_organismal",
RData.location=RData.location)

## End(Not run)
```

---

xDefineOntology *Function to define ontology and its annotations*

Description

xDefineOntology is supposed to define ontology and its annotations. It returns an object of class "aOnto".

Usage

```
xDefineOntology(
ontology = c(NA, "GOBP", "GOMF", "GOCC", "PSG", "PS", "PS2", "SF",
"Pfam", "DO",
"HPPA", "HPMI", "HPCM", "HPMA", "MP", "EF", "MsigdbH", "MsigdbC1",
"MsigdbC2CGP",
"MsigdbC2CPall", "MsigdbC2CP", "MsigdbC2KEGG", "MsigdbC2REACTOME",
"MsigdbC2BIOCARTA", "MsigdbC3TFT", "MsigdbC3MIR", "MsigdbC4CGN",
"MsigdbC4CM",
"MsigdbC5BP", "MsigdbC5MF", "MsigdbC5CC", "MsigdbC6", "MsigdbC7",
"DGIdb", "GTExV4",
"GTExV6p", "GTExV7", "CreedsDisease", "CreedsDiseaseUP",
"CreedsDiseaseDN",
"CreedsDrug", "CreedsDrugUP", "CreedsDrugDN", "CreedsGene",
"CreedsGeneUP",
"CreedsGeneDN", "KEGG", "KEGGmetabolism", "KEGGgenetic",
"KEGGenvironmental",
"KEGGcellular", "KEGGorganismal", "KEGGdisease", "REACTOME",
"REACTOME_ImmuneSystem",
"REACTOME_SignalTransduction", "CGL", "SIFTS2GOBP", "SIFTS2GOMF",
"SIFTS2GOCC",
```

```
"EnrichrARCHS4Cells", "EnrichrARCHS4Tissues", "EnrichrHumanGeneAtlas",
"EnrichrTissueHumanProteomeMap", "EnrichrTissueProteomicsDB",
"EnrichrAchillesFitnessD", "EnrichrAchillesFitnessI", "EnrichrDSigDB",
"EnrichrOMIM",
"EnrichrOMIMexpanded", "EnrichrdbGaP", "EnrichrJensenDiseases",
"EnrichrJensenTissues", "EnrichrBioCarta", "EnrichrKEGG",
"EnrichrNCIpathway",
"EnrichrPanther", "EnrichrReactome", "EnrichrWikiPathways",
"EnrichrhuMAP",
"EnrichrChEA", "EnrichrConsensusTFs", "EnrichrEncodeTF",
"EnrichrTFlof",
"EnrichrTFpert"),
ontology.customised = NULL,
anno.identity = c("GeneID", "Symbol"),
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

ontology	the ontology supported currently. It can be "GOBP" for Gene Ontology Biological Process, "GOMF" for Gene Ontology Molecular Function, "GOCC" for Gene Ontology Cellular Component, "PSG" for phylostratigraphy (phylostratific age), "PS" for sTOL-based phylostratific age information, "PS2" for the collapsed PS version (inferred ancestors being collapsed into one with the known taxonomy information), "SF" for SCOP domain superfamilies, "Pfam" for Pfam domain families, "DO" for Disease Ontology, "HPPA" for Human Phenotype Phenotypic Abnormality, "HPMI" for Human Phenotype Mode of Inheritance, "HPCM" for Human Phenotype Clinical Modifier, "HPMA" for Human Phenotype Mortality Aging, "MP" for Mammalian Phenotype, "EF" for Experimental Factor Ontology (used to annotate GWAS Catalog genes), Drug-Gene Interaction database ("DGIdb") for druggable categories, tissue-specific eQTL-containing genes from GTEx ("GTExV4", "GTExV6p" and "GTExV7"), crowd extracted expression of differential signatures from CREEDS ("CreedsDisease", "CreedsDiseaseUP", "CreedsDiseaseDN", "CreedsDrug", "CreedsDrugUP", "CreedsDrugDN", "CreedsGene", "CreedsGeneUP" and "CreedsGeneDN"), KEGG pathways (including 'KEGG' for all, 'KEGGmetabolism' for 'Metabolism' pathways, 'KEGGgenetic' for 'Genetic Information Processing' pathways, 'KEGGenvironmental' for 'Environmental Information Processing' pathways, 'KEGGcellular' for 'Cellular Processes' pathways, 'KEGGorganismal' for 'Organismal Systems' pathways, and 'KEGGdisease' for 'Human Diseases' pathways), 'REACTOME' for REACTOME pathways or 'REACTOME_x' for its sub-ontologies (where x can be 'CellCellCommunication', 'CellCycle', 'CellularResponsesToExternalStimuli', 'ChromatinOrganization', 'CircadianClock', 'DevelopmentalBiology', 'DigestionAndAbsorption', 'Disease', 'DNARepair', 'DNAReplication', 'ExtracellularMatrixOrganization', 'GeneExpression(Transcription)', 'Hemostasis', 'ImmuneSystem', 'Metabolism', 'MetabolismOfProteins', 'MetabolismOfRNA', 'Mitophagy', 'MuscleContraction', 'NeuronalSystem', 'OrganelleBiogenesisAnd-

Maintenance', 'ProgrammedCellDeath', 'Reproduction', 'SignalTransduction', 'TransportOfSmallMolecules', 'VesicleMediatedTransport'), and the molecular signatures database (Msigdb, including "MsigdbH", "MsigdbC1", "MsigdbC2CGP", "MsigdbC2CPall", "MsigdbC2CP", "MsigdbC2KEGG", "MsigdbC2REACTOME", "MsigdbC2BIOCARTA", "MsigdbC3TFT", "MsigdbC3MIR", "MsigdbC4CGN", "MsigdbC4CM", "MsigdbC5BP", "MsigdbC5MF", "MsigdbC5CC", "MsigdbC6", "MsigdbC7"), and the SIFTS database ("SIFTS2GOBP" for Gene Ontology Biological Process, "SIFTS2GOMF" for Gene Ontology Molecular Function, "SIFTS2GOCC" for Gene Ontology Cellular Component), and 'EnrichrX' for Enrichr libraries (where X can be "AchillesFitnessD","AchillesFitnessI","ARCHS4Cells","ARCHS4Tissu

`ontology.customised`
: an object 'GS'. Higher priority over 'ontology' above. Required, otherwise it will return NULL

`anno.identity`
: identity for gene annotations. It can be "GeneID" for Gene ID and "Symbol" for gene symbol. Does not support the customised ontology

`verbose`
: logical to indicate whether the messages will be displayed in the screen. By default, it sets to false for no display

`RData.location`
: the characters to tell the location of built-in RData files. See xRDataLoader for details

`guid`
: a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details

Value

an object of class "aOnto", a list with two components: an igraph object 'g' (with graph attributes 'ontology' and 'type' [either 'dag' or 'iso']) and a list 'anno'

Note

none

See Also

xRDataLoader, xGeneID2Symbol

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

## Not run:
aOnto <- xDefineOntology("HPPA", RData.location=RData.location)

# only support internally (please contact us if you would like to use)
aOnto <- xDefineOntology("REACTOME_ImmuneSystem",
RData.location=RData.location)
aOnto <- xDefineOntology("KEGGenvironmental",
RData.location=RData.location)
aOnto <- xDefineOntology("CGL", RData.location=RData.location)
```

```
# advanced use: customisation
GS <- xRDataLoader('org.Mm.egKEGG', RData.location=RData.location)
res <- xDefineOntology(ontology.customised=GS)

## End(Not run)
```

---

xEnricher *Function to conduct enrichment analysis given the input data and the ontology and its annotation*

Description

xEnricher is supposed to conduct enrichment analysis given the input data and the ontology direct acyclic graph (DAG) and its annotation. It returns an object of class "eTerm". Enrichment analysis is based on either Fisher's exact test or Hypergeometric test. The test can respect the hierarchy of the ontology.

Usage

```
xEnricher(
data,
annotation,
g,
background = NULL,
size.range = c(10, 2000),
min.overlap = 5,
which.distance = NULL,
test = c("fisher", "hypergeo", "binomial"),
background.annotatable.only = NULL,
p.tail = c("one-tail", "two-tails"),
p.adjust.method = c("BH", "BY", "bonferroni", "holm", "hochberg",
"hommel"),
ontology.algorithm = c("none", "pc", "elim", "lea"),
elim.pvalue = 0.01,
lea.depth = 2,
path.mode = c("all_paths", "shortest_paths", "all_shortest_paths"),
true.path.rule = TRUE,
verbose = TRUE
)
```

Arguments

data	an input vector containing a list of genes or SNPs of interest
annotation	the vertices/nodes for which annotation data are provided. It can be a sparse Matrix of class "dgCMatrix" (with variants/genes as rows and terms as columns), or a list of nodes/terms each containing annotation data, or an object of class 'GS' (basically a list for each node/term with annotation data)

`g` an object of class "igraph" to represent DAG. It must have node/vertice attributes: "name" (i.e. "Term ID"), "term_id" (i.e. "Term ID"), "term_name" (i.e "Term Name") and "term_distance" (i.e. Term Distance: the distance to the root; always 0 for the root itself)

`background` a background vector. It contains a list of genes or SNPs as the test background. If NULL, by default all annotatable are used as background

`size.range` the minimum and maximum size of members of each term in consideration. By default, it sets to a minimum of 10 but no more than 2000

`min.overlap` the minimum number of overlaps. Only those terms with members that overlap with input data at least min.overlap (3 by default) will be processed

`which.distance` which terms with the distance away from the ontology root (if any) is used to restrict terms in consideration. By default, it sets to 'NULL' to consider all distances

`test` the test statistic used. It can be "fisher" for using fisher's exact test, "hypergeo" for using hypergeometric test, or "binomial" for using binomial test. Fisher's exact test is to test the independence between gene group (genes belonging to a group or not) and gene annotation (genes annotated by a term or not), and thus compare sampling to the left part of background (after sampling without replacement). Hypergeometric test is to sample at random (without replacement) from the background containing annotated and non-annotated genes, and thus compare sampling to background. Unlike hypergeometric test, binomial test is to sample at random (with replacement) from the background with the constant probability. In terms of the ease of finding the significance, they are in order: hypergeometric test > fisher's exact test > binomial test. In other words, in terms of the calculated p-value, hypergeometric test < fisher's exact test < binomial test

`background.annotatable.only` logical to indicate whether the background is further restricted to the annotatable. By default, it is NULL: if ontology.algorithm is not 'none', it is always TRUE; otherwise, it depends on the background (if not provided, it will be TRUE; otherwise FALSE). Surely, it can be explicitly stated

`p.tail` the tail used to calculate p-values. It can be either "two-tails" for the significance based on two-tails (ie both over- and under-overrepresentation) or "one-tail" (by default) for the significance based on one tail (ie only over-representation)

`p.adjust.method` the method used to adjust p-values. It can be one of "BH", "BY", "bonferroni", "holm", "hochberg" and "hommel". The first two methods "BH" (widely used) and "BY" control the false discovery rate (FDR: the expected proportion of false discoveries amongst the rejected hypotheses); the last four methods "bonferroni", "holm", "hochberg" and "hommel" are designed to give strong control of the family-wise error rate (FWER). Notes: FDR is a less stringent condition than FWER

`ontology.algorithm` the algorithm used to account for the hierarchy of the ontology. It can be one of "none", "pc", "elim" and "lea". For details, please see 'Note' below

elim.pvalue	the parameter only used when "ontology.algorithm" is "elim". It is used to control how to declare a signficantly enriched term (and subsequently all genes in this term are eliminated from all its ancestors)
lea.depth	the parameter only used when "ontology.algorithm" is "lea". It is used to control how many maximum depth is used to consider the children of a term (and subsequently all genes in these children term are eliminated from the use for the recalculation of the signifance at this term)
path.mode	the mode of paths induced by vertices/nodes with input annotation data. It can be "all_paths" for all possible paths to the root, "shortest_paths" for only one path to the root (for each node in query), "all_shortest_paths" for all shortest paths to the root (i.e. for each node, find all shortest paths with the equal lengths)
true.path.rule	logical to indicate whether the true-path rule should be applied to propagate annotations. By default, it sets to true
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

an object of class "eTerm", a list with following components:

- term_info: a matrix of nTerm X 4 containing snp/gene set information, where nTerm is the number of terms, and the 4 columns are "id" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- annotation: a list of terms containing annotations, each term storing its annotations. Always, terms are identified by "id"
- g: an igraph object to represent DAG
- data: a vector containing input data in consideration. It is not always the same as the input data as only those mappable are retained
- background: a vector containing the background data. It is not always the same as the input data as only those mappable are retained
- overlap: a list of overlapped snp/gene sets, each storing snps/genes overlapped between a snp/gene set and the given input data (i.e. the snps/genes of interest). Always, gene sets are identified by "id"
- fc: a vector containing fold changes
- zscore: a vector containing z-scores
- pvalue: a vector containing p-values
- adjp: a vector containing adjusted p-values. It is the p value but after being adjusted for multiple comparisons
- or: a vector containing odds ratio
- CIl: a vector containing lower bound confidence interval for the odds ratio
- CIu: a vector containing upper bound confidence interval for the odds ratio
- cross: a matrix of nTerm X nTerm, with an on-diagnal cell for the overlapped-members observed in an individaul term, and off-diagnal cell for the overlapped-members shared between two terms
- call: the call that produced this result

Note

The interpretation of the algorithms used to account for the hierarchy of the ontology is:

- "none": does not consider the ontology hierarchy at all.
- "lea": estimates the significance of a term in terms of the significance of its children at the maximum depth (e.g. 2). Precisely, once snps/genes are already annotated to any children terms with a more signficance than itself, then all these snps/genes are eliminated from the use for the recalculation of the signifance at that term. The final p-values takes the maximum of the original p-value and the recalculated p-value.
- "elim": estimates the significance of a term in terms of the significance of its all children. Precisely, once snps/genes are already annotated to a signficantly enriched term under the cutoff of e.g. pvalue<1e-2, all these snps/genes are eliminated from the ancestors of that term).
- "pc": requires the significance of a term not only using the whole snps/genes as background but also using snps/genes annotated to all its direct parents/ancestors as background. The final p-value takes the maximum of both p-values in these two calculations.
- "Notes": the order of the number of significant terms is: "none" > "lea" > "elim" > "pc".

See Also

xDAGanno

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# 1) SNP-based enrichment analysis using GWAS Catalog traits (mapped to EF)
# 1a) ig.EF (an object of class "igraph" storing as a directed graph)
g <- xRDataLoader('ig.EF')

# 1b) load GWAS SNPs annotated by EF (an object of class "dgCMatrix" storing a spare matrix)
anno <- xRDataLoader(RData='GWAS2EF')

# 1c) optionally, provide the test background (if not provided, all annotatable SNPs)
background <- rownames(anno)

# 1d) provide the input SNPs of interest (eg 'EFO:0002690' for 'systemic lupus erythematosus')
ind <- which(colnames(anno)=='EFO:0002690')
data <- rownames(anno)[anno[,ind]==1]
data

# 1e) perform enrichment analysis
eTerm <- xEnricher(data=data, annotation=anno, background=background,
g=g, path.mode=c("all_paths"))

# 1f) view enrichment results for the top significant terms
xEnrichViewer(eTerm)

# 1f') save enrichment results to the file called 'EF_enrichments.txt'
res <- xEnrichViewer(eTerm, top_num=length(eTerm$adjp), sortBy="adjp",
```

```
details=TRUE)
output <- data.frame(term=rownames(res), res)
utils::write.table(output, file="EF_enrichments.txt", sep="\t",
row.names=FALSE)

# 1g) barplot of significant enrichment results
bp <- xEnrichBarplot(eTerm, top_num="auto", displayBy="adjp")
print(bp)

# 1h) visualise the top 10 significant terms in the ontology hierarchy
# color-code terms according to the adjust p-values (taking the form of 10-based negative logarithm)
xEnrichDAGplot(eTerm, top_num=10, displayBy="adjp",
node.info=c("full_term_name"))
# color-code terms according to the z-scores
xEnrichDAGplot(eTerm, top_num=10, displayBy="zscore",
node.info=c("full_term_name"))

## End(Not run)
```

`xEnricherGenes` *Function to conduct enrichment analysis given a list of genes and the ontology in query*

Description

`xEnricherGenes` is supposed to conduct enrichment analysis given the input data and the ontology in query. It returns an object of class "eTerm". Enrichment analysis is based on either Fisher's exact test or Hypergeometric test. The test can respect the hierarchy of the ontology. Now it supports enrichment analysis using a wide variety of ontologies such as Gene Ontology and Phenotype Ontologies.

Usage

```
xEnricherGenes(
data,
background = NULL,
check.symbol.identity = FALSE,
ontology = NA,
ontology.customised = NULL,
size.range = c(10, 2000),
min.overlap = 5,
which.distance = NULL,
test = c("fisher", "hypergeo", "binomial"),
background.annotatable.only = NULL,
p.tail = c("one-tail", "two-tails"),
p.adjust.method = c("BH", "BY", "bonferroni", "holm", "hochberg",
"hommel"),
ontology.algorithm = c("none", "pc", "elim", "lea"),
```

```
elim.pvalue = 0.01,
lea.depth = 2,
path.mode = c("all_paths", "shortest_paths", "all_shortest_paths"),
true.path.rule = FALSE,
verbose = TRUE,
silent = FALSE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

data	an input vector containing gene symbols
background	a background vector containing gene symbols as the test background. If NULL, by default all annotatable are used as background
check.symbol.identity	logical to indicate whether to match the input data/background via Synonyms for those unmatchable by official gene symbols. By default, it sets to false
ontology	the ontology supported currently. By default, it is 'NA' to disable this option. Pre-built ontology and annotation data are detailed in xDefineOntology.
ontology.customised	an object 'GS'. Higher priority over 'ontology' above. Required, otherwise it will return NULL
size.range	the minimum and maximum size of members of each term in consideration. By default, it sets to a minimum of 10 but no more than 2000
min.overlap	the minimum number of overlaps. Only those terms with members that overlap with input data at least min.overlap (3 by default) will be processed
which.distance	which terms with the distance away from the ontology root (if any) is used to restrict terms in consideration. By default, it sets to 'NULL' to consider all distances
test	the test statistic used. It can be "fisher" for using fisher's exact test, "hypergeo" for using hypergeometric test, or "binomial" for using binomial test. Fisher's exact test is to test the independence between gene group (genes belonging to a group or not) and gene annotation (genes annotated by a term or not), and thus compare sampling to the left part of background (after sampling without replacement). Hypergeometric test is to sample at random (without replacement) from the background containing annotated and non-annotated genes, and thus compare sampling to background. Unlike hypergeometric test, binomial test is to sample at random (with replacement) from the background with the constant probability. In terms of the ease of finding the significance, they are in order: hypergeometric test > fisher's exact test > binomial test. In other words, in terms of the calculated p-value, hypergeometric test < fisher's exact test < binomial test
background.annotatable.only	logical to indicate whether the background is further restricted to the annotatable. By default, it is NULL: if ontology.algorithm is not 'none', it is always

TRUE; otherwise, it depends on the background (if not provided, it will be TRUE; otherwise FALSE). Surely, it can be explicitly stated

`p.tail` the tail used to calculate p-values. It can be either "two-tails" for the significance based on two-tails (ie both over- and under-overrepresentation) or "one-tail" (by default) for the significance based on one tail (ie only over-representation)

`p.adjust.method`
the method used to adjust p-values. It can be one of "BH", "BY", "bonferroni", "holm", "hochberg" and "hommel". The first two methods "BH" (widely used) and "BY" control the false discovery rate (FDR: the expected proportion of false discoveries amongst the rejected hypotheses); the last four methods "bonferroni", "holm", "hochberg" and "hommel" are designed to give strong control of the family-wise error rate (FWER). Notes: FDR is a less stringent condition than FWER

`ontology.algorithm`
the algorithm used to account for the hierarchy of the ontology. It can be one of "none", "pc", "elim" and "lea". For details, please see 'Note' below

`elim.pvalue` the parameter only used when "ontology.algorithm" is "elim". It is used to control how to declare a signficantly enriched term (and subsequently all genes in this term are eliminated from all its ancestors)

`lea.depth` the parameter only used when "ontology.algorithm" is "lea". It is used to control how many maximum depth is used to consider the children of a term (and subsequently all genes in these children term are eliminated from the use for the recalculation of the signifance at this term)

`path.mode` the mode of paths induced by vertices/nodes with input annotation data. It can be "all_paths" for all possible paths to the root, "shortest_paths" for only one path to the root (for each node in query), "all_shortest_paths" for all shortest paths to the root (i.e. for each node, find all shortest paths with the equal lengths)

`true.path.rule` logical to indicate whether the true-path rule should be applied to propagate annotations. By default, it sets to false

`verbose` logical to indicate whether the messages will be displayed in the screen. By default, it sets to false for no display

`silent` logical to indicate whether the messages will be silent completely. By default, it sets to false. If true, verbose will be forced to be false

`RData.location` the characters to tell the location of built-in RData files. See `xRDataLoader` for details

`guid` a valid (5-character) Global Unique IDentifier for an OSF project. See `xRDataLoader` for details

Value

an object of class "eTerm", a list with following components:

- `term_info`: a matrix of nTerm X 4 containing snp/gene set information, where nTerm is the number of terms, and the 4 columns are "id" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"

- annotation: a list of terms containing annotations, each term storing its annotations. Always, terms are identified by "id"
- g: an igraph object to represent DAG
- data: a vector containing input data in consideration. It is not always the same as the input data as only those mappable are retained
- background: a vector containing the background data. It is not always the same as the input data as only those mappable are retained
- overlap: a list of overlapped snp/gene sets, each storing snps overlapped between a snp/gene set and the given input data (i.e. the snps of interest). Always, gene sets are identified by "id"
- fc: a vector containing fold changes
- zscore: a vector containing z-scores
- pvalue: a vector containing p-values
- adjp: a vector containing adjusted p-values. It is the p value but after being adjusted for multiple comparisons
- or: a vector containing odds ratio
- CIl: a vector containing lower bound confidence interval for the odds ratio
- CIu: a vector containing upper bound confidence interval for the odds ratio
- cross: a matrix of nTerm X nTerm, with an on-diagnal cell for the overlapped-members observed in an individaul term, and off-diagnal cell for the overlapped-members shared betwene two terms
- call: the call that produced this result

Note

The interpretation of the algorithms used to account for the hierarchy of the ontology is:

- "none": does not consider the ontology hierarchy at all.
- "lea": computers the significance of a term in terms of the significance of its children at the maximum depth (e.g. 2). Precisely, once snps are already annotated to any children terms with a more signficance than itself, then all these snps are eliminated from the use for the recalculation of the signifance at that term. The final p-values takes the maximum of the original p-value and the recalculated p-value.
- "elim": computers the significance of a term in terms of the significance of its all children. Precisely, once snps are already annotated to a signficantly enriched term under the cutoff of e.g. pvalue<1e-2, all these snps are eliminated from the ancestors of that term).
- "pc": requires the significance of a term not only using the whole snps as background but also using snps annotated to all its direct parents/ancestors as background. The final p-value takes the maximum of both p-values in these two calculations.
- "Notes": the order of the number of significant terms is: "none" > "lea" > "elim" > "pc".

See Also

xDefineOntology, xSymbol2GeneID, xEnricher, xSymbol2GeneID, xRDataLoader

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# Gene-based enrichment analysis using REACTOME pathways
# a) provide the input Genes of interest (eg 500 randomly chosen human genes)
## load human genes
org.Hs.eg <- xRDataLoader(RData='org.Hs.eg',
RData.location=RData.location)
set.seed(825)
data <- as.character(sample(org.Hs.eg$gene_info$Symbol, 500))
data

# optionally, provide the test background (if not provided, all human genes)
#background <- as.character(org.Hs.eg$gene_info$Symbol)

# b) perform enrichment analysis
eTerm <- xEnricherGenes(data=data, ontology="MsigdbC2REACTOME",
RData.location=RData.location)

# c) view enrichment results for the top significant terms
xEnrichViewer(eTerm)

# d) save enrichment results to the file called 'REACTOME_enrichments.txt'
res <- xEnrichViewer(eTerm, top_num=length(eTerm$adjp), sortBy="adjp",
details=TRUE)
output <- data.frame(term=rownames(res), res)
utils::write.table(output, file="REACTOME_enrichments.txt", sep="\t",
row.names=FALSE)

# e) barplot of significant enrichment results
gp <- xEnrichBarplot(eTerm, top_num="auto", displayBy="adjp")
print(gp)

# f) visualise the top 10 significant terms in the ontology hierarchy
# color-code terms according to the adjust p-values (taking the form of 10-based negative logarithm)
xEnrichDAGplot(eTerm, top_num=10, displayBy="adjp",
node.info=c("full_term_name"), graph.node.attrs=list(fontsize=25))
# color-code terms according to the z-scores
xEnrichDAGplot(eTerm, top_num=10, displayBy="zscore",
node.info=c("full_term_name"), graph.node.attrs=list(fontsize=25))

# g) visualise the significant terms in the ontology hierarchy
# restricted to Immune System ('R-HSA-168256') or Signal Transduction ('R-HSA-162582')
g <- xRDataLoader(RData.customised='ig.REACTOME',
RData.location=RData.location)
neighs.out <- igraph::neighborhood(g, order=vcount(g),
nodes=c("R-HSA-162582","R-HSA-168256"), mode="out")
nodeInduced <- V(g)[unique(unlist(neighs.out))]$name
ig <- igraph::induced.subgraph(g, vids=nodeInduced)
xEnrichDAGplot(eTerm, top_num="auto", ig=ig, displayBy="adjp",
node.info=c("full_term_name"), graph.node.attrs=list(fontsize=25))
```

```
## End(Not run)
```

---

xEnrichForest *Function to visualise enrichment results using a forest plot*

Description

xEnrichForest is supposed to visualise enrichment results using a forest plot. A point is colored by the significance level, and a horizontal line for the 95 the wider the CI, the less reliable). It returns an object of class "ggplot".

Usage

```
xEnrichForest(
eTerm,
top_num = 10,
FDR.cutoff = 0.05,
CI.one = TRUE,
colormap = "ggplot2.top",
ncolors = 64,
zlim = NULL,
barwidth = 0.5,
barheight = NULL,
wrap.width = NULL,
font.family = "sans",
drop = FALSE,
sortBy = c("or", "adjp", "fdr", "pvalue", "zscore", "fc", "nAnno",
"nOverlap",
"none")
)
```

Arguments

eTerm	an object of class "eTerm" or "ls_eTerm". Alterntively, it can be a data frame having all these columns (named as 'group','ontology','name','adjp','or','CIl','CIu')
top_num	the number of the top terms (sorted according to OR). For the eTerm object, if it is 'auto' (for eTerm), only the significant terms (see below FDR.cutoff) will be displayed
FDR.cutoff	FDR cutoff used to declare the significant terms. By default, it is set to 0.05. Only works when top_num is 'auto' above
CI.one	logical to indicate whether to allow the inclusion of one in CI. By default, it is TURE (allowed)
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb"

	(white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
ncolors	the number of colors specified over the colormap
zlim	the minimum and maximum z values for which colors should be plotted, defaulting to the range of the -log10(FDR)
barwidth	the width of the colorbar. Default value is 'legend.key.width' or 'legend.key.size' in 'theme' or theme
barheight	the height of the colorbar. Default value is 'legend.key.height' or 'legend.key.size' in 'theme' or theme
wrap.width	a positive integer specifying wrap width of name
font.family	the font family for texts
drop	logical to indicate whether all factor levels not used in the data will automatically be dropped. If FALSE (by default), all factor levels will be shown, regardless of whether or not they appear in the data
sortBy	which statistics will be used for sorting and viewing gene sets (terms). It can be "adjp" or "fdr" for adjusted p value (FDR), "pvalue" for p value, "zscore" for enrichment z-score, "fc" for enrichment fold change, "nAnno" for the number of sets (terms), "nOverlap" for the number in overlaps, "or" for the odds ratio, and "none" for ordering according to ID of terms. It only works when the input is an eTerm object

Value

an object of class "ggplot"

Note

none

See Also

xEnricherGenes, xEnrichViewer

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:

# provide the input Genes of interest (eg 100 randomly chosen human genes)
## load human genes
org.Hs.eg <- xRDataLoader(RData='org.Hs.eg',
RData.location=RData.location)
set.seed(825)
data <- as.character(sample(org.Hs.eg$gene_info$Symbol, 100))
data
```

```
# optionally, provide the test background (if not provided, all human genes)
#background <- as.character(org.Hs.eg$gene_info$Symbol)

# 1) Gene-based enrichment analysis using REACTOME pathways
# perform enrichment analysis
eTerm <- xEnricherGenes(data, ontology="REACTOME",
RData.location=RData.location)
## forest plot of enrichment results
gp <- xEnrichForest(eTerm, top_num="auto", FDR.cutoff=0.05)

# 2) Gene-based enrichment analysis using ontologies (REACTOME and GOMF)
# perform enrichment analysis
ls_eTerm <- xEnricherGenesAdv(data, ontologies=c("REACTOME","GOMF"),
RData.location=RData.location)
## forest plot of enrichment results
gp <- xEnrichForest(ls_eTerm, FDR.cutoff=0.1)

## End(Not run)
```

---

xEnrichViewer *Function to view enrichment results*

Description

xEnrichViewer is supposed to view results of enrichment analysis.

Usage

```
xEnrichViewer(
eTerm,
top_num = 10,
sortBy = c("adjp", "fdr", "pvalue", "zscore", "fc", "nAnno",
"nOverlap", "or",
"none"),
decreasing = NULL,
details = FALSE
)
```

Arguments

eTerm	an object of class "eTerm"
top_num	the number of the top terms (sorted according to 'sortBy' below) will be viewed
sortBy	which statistics will be used for sorting and viewing gene sets (terms). It can be "adjp" or "fdr" for adjusted p value (FDR), "pvalue" for p value, "zscore" for enrichment z-score, "fc" for enrichment fold change, "nAnno" for the number of sets (terms), "nOverlap" for the number in overlaps, "or" for the odds ratio, and "none" for ordering according to ID of terms

decreasing	logical to indicate whether to sort in a decreasing order. If it is null, it would be true for "zscore", "nAnno" or "nOverlap"; otherwise it would be false
details	logical to indicate whether the detailed information of gene sets (terms) is also viewed. By default, it sets to false for no inclusion

Value

a data frame with following components:

- id: term ID; as rownames
- name: term name
- nAnno: number in members annotated by a term
- nOverlap: number in overlaps
- fc: enrichment fold changes
- zscore: enrichment z-score
- pvalue: nominal p value
- adjp: adjusted p value (FDR)
- or: a vector containing odds ratio
- CIl: a vector containing lower bound confidence interval for the odds ratio
- CIu: a vector containing upper bound confidence interval for the odds ratio
- distance: term distance or other information; optional, it is only appended when "details" is true
- members_Overlap: members (represented as Gene Symbols) in overlaps; optional, it is only appended when "details" is true
- members_Anno: members (represented as Gene Symbols) in annotations; optional, it is only appended when "details" is true

Note

none

See Also

xEnrichViewer

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
xEnrichViewer(eTerm)

## End(Not run)
```

xGeneID2Symbol *Function to convert gene symbols to entrez geneid*

Description

xGeneID2Symbol is supposed to convert gene symbols to entrez geneid.

Usage

```
xGeneID2Symbol(
data,
org = c("human", "mouse"),
details = FALSE,
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

data	an input vector containing gene symbols
org	a character specifying an organism. Currently supported organisms are 'human' and 'mouse'. It can be an object 'EG'
details	logical to indicate whether to result in a data frame (in great details). By default, it sets to false
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to false for no display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details
guid	a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details

Value

a vector containing symbol with 'NA' for the unmatched if (details set to false); otherwise, a data frame is returned

Note

none.

See Also

xRDataLoader

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# a) provide the input Genes of interest (eg 100 randomly chosen human genes)
## load human genes
org.Hs.eg <- xRDataLoader(RData='org.Hs.eg')
GeneID <- sample(org.Hs.eg$gene_info$GeneID, 100)
GeneID

# b) convert into GeneID
Symbol <- xGeneID2Symbol(GeneID)

# c) convert into a data frame
df <- xGeneID2Symbol(GeneID, details=TRUE)


# advanced use
df <- xGeneID2Symbol(GeneID, org=org.Hs.eg, details=TRUE)

## End(Not run)
```

---

xGGnetwork *Function to visualise an igraph object using ggnetwork*

Description

xGGnetwork is supposed to visualise an igraph object using ggnetwork.

Usage

```
xGGnetwork(
g,
node.label = NULL,
label.wrap.width = NULL,
label.wrap.lineheight = 0.8,
node.label.size = NULL,
node.label.fontface = "plain",
node.label.color = "darkblue",
node.label.alpha = 0.9,
node.label.padding = 1,
node.label.arrow = 0.01,
node.label.force = 1,
node.shape = 19,
node.shape.title = NULL,
node.xcoord = NULL,
node.ycoord = NULL,
node.color = NULL,
node.color.title = NULL,
```

```
colormap = "grey-orange-darkred",
ncolors = 64,
zlim = NULL,
na.color = "grey80",
node.color.alpha = 1,
node.size = NULL,
node.size.title = NULL,
node.size.range = c(1, 4),
slim = NULL,
title = "",
edge.size = 0.5,
edge.color = "black",
edge.color.alpha = 0.5,
edge.curve = 0.1,
edge.arrow = 2,
edge.arrow.gap = 0.02,
ncolumns = NULL
)
```

Arguments

g	an object of class "igraph". For an advanced use, it can be a list of igraph objects; in this case, multiple panels will be shown (particularly useful when visualising the same network but color-coded differently)
node.label	either a vector labelling nodes or a character specifying which node attribute used for the labelling. If NULL (by default), no node labelling
label.wrap.width	a positive integer specifying wrap width of node labelling
label.wrap.lineheight	line height spacing for text in ggplot. By default it is 0.8
node.label.size	a character specifying which node attribute used for node label size
node.label.fontface	a character specifying which node attribute used for node label fontface ('plain', 'bold', 'italic', 'bold.italic')
node.label.color	a character specifying which node attribute used for the node label color
node.label.alpha	the 0-1 value specifying transparency of node labelling
node.label.padding	the padding around the labeled node
node.label.arrow	the arrow pointing to the labeled node
node.label.force	the repelling force between overlapping labels
node.shape	an integer specifying node shape or a character specifying which node attribute used for the node shape (no matter whether it is numeric or character)

node.shape.title
: a character specifying the title for node shaping

node.xcoord
: a vector specifying x coordinates. If NULL, it will be created using igraph::layout_as_tree

node.ycoord
: a vector specifying y coordinates. If NULL, it will be created using igraph::layout_as_tree

node.color
: a character specifying which node attribute used for node coloring

node.color.title
: a character specifying the title for node coloring

colormap
: short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta), and "ggplot2" (emulating ggplot2 default color palette). Alternatively, any hyphen-separated HTML color names, e.g. "lightyellow-orange" (by default), "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names

ncolors
: the number of colors specified over the colormap

zlim
: the minimum and maximum values for which colors should be plotted

na.color
: the color for NAs. By default, it is 'grey80'

node.color.alpha
: the 0-1 value specifying transparency of node colors

node.size
: either a vector specifying node size or a character specifying which node attribute used for the node size

node.size.title
: a character specifying the title for node sizing

node.size.range
: the range of actual node size. Can be two values (range) or a value (fixed size)

slim
: the minimum and maximum values for which sizes should be plotted

title
: a character specifying the title for the plot

edge.size
: a numeric value specifying the edge size. By default, it is 0.5. It can be a character specifying which edge attribute defining the edge size (though without the legend)

edge.color
: a character specifying the edge color. By default, it is "black". It can be a character specifying which edge attribute defining the edge color (though

edge.color.alpha
: the 0-1 value specifying transparency of edge color. By default, it is 0.5. It can be a character specifying which edge attribute defining the transparency of edge color (though without the legend)

edge.curve
: a numeric value specifying the edge curve. 0 for the straight line

edge.arrow
: a numeric value specifying the edge arrow. By default, it is 2

edge.arrow.gap
: a gap between the arrow and the node

ncolumns
: an integer specifying the number of columns for facet_wrap. By defaul, it is NULL (decided on according to the number of groups that will be visualised)

Value

a ggplot object, appended with 'data_nodes' and 'data_edges'

Note

none

See Also

xGGnetwork

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
############################
# load REACTOME
# restricted to Immune System ('R-HSA-168256') or Signal Transduction ('R-HSA-162582')
g <- xRDataLoader(RData.customised='ig.REACTOME',
RData.location=RData.location)
neighs.out <- igraph::neighborhood(g, order=vcount(g),
nodes="R-HSA-168256", mode="out")
nodeInduced <- V(g)[unique(unlist(neighs.out))]$name
ig <- igraph::induced.subgraph(g, vids=nodeInduced)

# visualise the graph with vertices being color-coded
V(ig)$degree <- igraph::degree(ig)
gp <- xGGnetwork(g=ig, node.label='term_id', label.wrap.width=30,
node.label.size=2, node.label.color='black', node.label.alpha=0.8,
node.label.padding=0, node.label.arrow=0, node.label.force=1,
node.shape=19, node.xcoord='xcoord', node.ycoord='ycoord',
node.color='degree', node.color.title='Degree',
colormap='grey-orange-darkred', ncolors=64, zlim=c(0,10),
node.size.range=3,
edge.color="black",edge.color.alpha=0.3,edge.curve=0.05,edge.arrow.gap=0.02,
title='')
# advanced use: visualise the list of graphs
ls_ig <- list(ig, ig)
gp <- xGGnetwork(g=ls_ig, node.label='term_id', label.wrap.width=30,
node.label.size=2, node.label.color='black', node.label.alpha=0.8,
node.label.padding=0, node.label.arrow=0, node.label.force=1,
node.shape=19, node.xcoord='xcoord', node.ycoord='ycoord',
node.color='degree', node.color.title='Degree',
colormap='grey-orange-darkred', ncolors=64, zlim=c(0,10),
node.size.range=3,
edge.color="black",edge.color.alpha=0.3,edge.curve=0.05,edge.arrow.gap=0.02,
title='')

############################
# load PhasedTargets
# restricted to disease ('EFO:0000408') or immune system disease ('EFO:0000540')
g <- xRDataLoader(RData.customised='ig.PhasedTargets',
```

```
RData.location=RData.location)
neighs.out <- igraph::neighborhood(g, order=vcount(g),
nodes="EFO:0000408", mode="out")
nodeInduced <- V(g)[unique(unlist(neighs.out))]$name
ig <- igraph::induced.subgraph(g, vids=nodeInduced)

# append with the number of approved and phased targets
dag <- ig
V(dag)$num_approved <- sapply(V(ig)$max_phase,function(x)
sum(x$max_phase>=4))
V(dag)$num_phased <- sapply(V(ig)$max_phase,function(x)
sum(x$max_phase>=0))
# keep nodes with num_approved >=20
dag_ig <- igraph::induced.subgraph(dag,
vids=which(V(dag)$num_approved>=20))
# (optional) further restricted to the direct children of the root
root <- dnet::dDAGroot(dag_ig)
neighs.out <- igraph::neighborhood(dag_ig, order=1, nodes=root,
mode="out")
nodeInduced <- V(dag_ig)[unique(unlist(neighs.out))]$name
dag_ig <- igraph::induced.subgraph(dag_ig, vids=nodeInduced)
# nodes colored by num_approved
V(dag_ig)$node_color <- log2(V(dag_ig)$num_approved)
glayout <- igraph::layout_with_kk(dag_ig)
V(dag_ig)$xcoord <- glayout[,1]
V(dag_ig)$ycoord <- glayout[,2]
gp <- xGGnetwork(g=dag_ig, node.label='term_name', label.wrap.width=30,
node.label.size=2, node.label.color='black', node.label.alpha=0.9,
node.label.padding=0, node.label.arrow=0, node.label.force=0.5,
node.shape=19, node.xcoord='xcoord', node.ycoord='ycoord',
node.color='node_color', node.color.title='Approved\n(log2-scale)',
colormap='ggplot2.top', ncolors=64, node.size.range=3,
edge.color="orange",edge.color.alpha=0.5,edge.curve=0.05,edge.arrow.gap=0.02,
title='')

############################
# visualise gene network
glayout <- igraph::layout_with_kk(g)
V(g)$xcoord <- glayout[,1]
V(g)$ycoord <- glayout[,2]
V(g)$degree <- igraph::degree(g)
gp <- xGGnetwork(g=g, node.label='name', node.label.size=2,
node.label.color='black', node.label.alpha=0.8, node.label.padding=0,
node.label.arrow=0, node.label.force=0.01, node.shape=19,
node.xcoord='xcoord', node.ycoord='ycoord', node.color='priority',
node.color.title='5-star\nrating', colormap='yellow-red', ncolors=64,
zlim=c(0,5), node.size='degree', node.size.title='Degree', slim=c(0,5),
edge.color="orange",edge.color.alpha=0.5,edge.curve=0,edge.arrow.gap=0.025,
title='')
gp_rating <- xGGnetwork(g=g, node.label='name', node.label.size=2,
node.label.color='black', node.label.alpha=0.8, node.label.padding=0.1,
node.label.arrow=0, node.label.force=0.01, node.shape=19,
node.xcoord='xcoord', node.ycoord='ycoord', node.color='priority',
```

```
node.color.title='5-star\nrating', colormap='white-yellow-red',
ncolors=64, zlim=c(0,5), node.size.range=5,
edge.color="orange",edge.color.alpha=0.3,edge.curve=0,edge.arrow.gap=0.02,
title='')

##########################
# use edge weight to color/size edges (without legends)
# edge color
#E(g)$color <- xColormap(colormap='RdYlBu', data=E(g)$weight)
#E(g)$size <- (E(g)$weight - min(E(g)$weight)) / (max(E(g)$weight) - min(E(g)$weight))
e.color <- subset(gp$data, !is.na(na.y))$e.color
gp + ggnetwork::geom_edges(color=e.color, show.legend=FALSE)
# edge size/thickness
e.size <- subset(gp$data, !is.na(na.y))$e.size
gp + ggnetwork::geom_edges(size=e.size, show.legend=FALSE)

## End(Not run)
```

---

xGR *Function to create a GRanges object given a list of genomic regions*

Description

xGR is supposed to create a GRanges object given a list of genomic regions.

Usage

```
xGR(
data,
format = c("chr:start-end", "data.frame", "bed", "GRanges"),
build.conversion = c(NA, "hg38.to.hg19", "hg18.to.hg19"),
add.name = TRUE,
remove.mcol = FALSE,
include.strand = FALSE,
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

data	input genomic regions (GR). If formatted as "chr:start-end" (see the next parameter 'format' below), GR should be provided as a vector in the format of 'chrN:start-end', where N is either 1-22 or X, start (or end) is genomic positional number; for example, 'chr1:13-20'. If formatted as a 'data.frame', the first three columns correspond to the chromosome (1st column), the starting chromosome position (2nd column), and the ending chromosome position (3rd column). If the format is indicated as 'bed' (browser extensible data), the same as 'data.frame'

format but the position is 0-based offset from chromomose position. If the genomic regions provided are not ranged but only the single position, the ending chromosome position (3rd column) is allowed not to be provided. The data could also be an object of 'GRanges' (in this case, formatted as 'GRanges')

`format` the format of the input data. It can be one of "chr:start-end", "data.frame", "bed" or "GRanges"

`build.conversion` the conversion from one genome build to another. The conversions supported are "hg38.to.hg19" and "hg18.to.hg19". By default it is NA (no need to do so)

`add.name` logical to add names. By default, it sets to true

`remove.mcol` logical to remove meta-columns. By default, it sets to false

`include.strand` logical to include strand. By default, it sets to false. It only works when the format is "data.frame" or "bed" and the input data has 4 columns

`verbose` logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

`RData.location` the characters to tell the location of built-in RData files. See xRDataLoader for details

`guid` a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details

Value

a GenomicRanges object

See Also

xLiftOver

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# a) provide the genomic regions
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
## get lead SNPs reported in AS GWAS and their significance info (p-values)
gr <- ImmunoBase$AS$variant
df <- as.data.frame(gr, row.names=NULL)
chr <- df$seqnames
start <- df$start
end <- df$end
data <- paste(chr,':',start,'-',end, sep='')

# b) create a GRanges object
GR <- xGR(data=data, format="chr:start-end",
RData.location=RData.location)

## End(Not run)
```

xGR2nGenes *Function to define nearby genes given a list of genomic regions*

Description

xGR2nGenes is supposed to define nearby genes given a list of genomic regions (GR) within certain distance window. The distance weight is calcualted as a decaying function of the gene-to-GR distance.

Usage

```
xGR2nGenes(
data,
format = c("chr:start-end", "data.frame", "bed", "GRanges"),
build.conversion = c(NA, "hg38.to.hg19", "hg18.to.hg19"),
distance.max = 50000,
decay.kernel = c("rapid", "slow", "linear", "constant"),
decay.exponent = 2,
GR.Gene = c("UCSC_knownGene", "UCSC_knownCanonical"),
scoring = FALSE,
scoring.scheme = c("max", "sum", "sequential"),
scoring.rescale = FALSE,
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

data	input genomic regions (GR). If formatted as "chr:start-end" (see the next parameter 'format' below), GR should be provided as a vector in the format of 'chrN:start-end', where N is either 1-22 or X, start (or end) is genomic positional number; for example, 'chr1:13-20'. If formatted as a 'data.frame', the first three columns correspond to the chromosome (1st column), the starting chromosome position (2nd column), and the ending chromosome position (3rd column). If the format is indicated as 'bed' (browser extensible data), the same as 'data.frame' format but the position is 0-based offset from chromomose position. If the genomic regions provided are not ranged but only the single position, the ending chromosome position (3rd column) is allowed not to be provided. The data could also be an object of 'GRanges' (in this case, formatted as 'GRanges')
format	the format of the input data. It can be one of "data.frame", "chr:start-end", "bed" or "GRanges"
build.conversion	the conversion from one genome build to another. The conversions supported are "hg38.to.hg19" and "hg18.to.hg19". By default it is NA (no need to do so)

`distance.max` the maximum distance between genes and GR. Only those genes no far way from this distance will be considered as seed genes. This parameter will influence the distance-component weights calculated for nearby GR per gene

`decay.kernel` a character specifying a decay kernel function. It can be one of 'slow' for slow decay, 'linear' for linear decay, and 'rapid' for rapid decay. If no distance weight is used, please select 'constant'

`decay.exponent` a numeric specifying a decay exponent. By default, it sets to 2

`GR.Gene` the genomic regions of genes. By default, it is 'UCSC_knownGene', that is, UCSC known genes (together with genomic locations) based on human genome assembly hg19. It can be 'UCSC_knownCanonical', that is, UCSC known canonical genes (together with genomic locations) based on human genome assembly hg19. Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to Gene Symbols. Then, tell "GR.Gene" with your RData file name (with or without extension), plus specify your file RData path in "RData.location". Note: you can also load your customised GR object directly

`scoring` logical to indicate whether gene-level scoring will be further calculated. By default, it sets to false

`scoring.scheme` the method used to calculate seed gene scores under a set of GR. It can be one of "sum" for adding up, "max" for the maximum, and "sequential" for the sequential weighting. The sequential weighting is done via: $\sum_{i=1} \frac{R_i}{i}$, where R_i is the i^{th} rank (in a descreasing order)

`scoring.rescale` logical to indicate whether gene scores will be further rescaled into the [0,1] range. By default, it sets to false

`verbose` logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

`RData.location` the characters to tell the location of built-in RData files. See `xRDataLoader` for details

`guid` a valid (5-character) Global Unique IDentifier for an OSF project. See `xRDataLoader` for details

Value

If scoring sets to false, a data frame with following columns:

- `Gene`: nearby genes
- `GR`: genomic regions
- `Dist`: the genomic distance between the gene and the GR
- `Weight`: the distance weight based on the genomic distance

If scoring sets to true, a data frame with following columns:

- `Gene`: nearby genes
- `Score`: gene score taking into account the distance weight based on the genomic distance

Note

For details on the decay kernels, please refer to xVisKernels

See Also

xGR, xRDataLoader, xSparseMatrix

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:

# a) provide the genomic regions
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
## get lead SNPs reported in AS GWAS and their significance info (p-values)
gr <- ImmunoBase$AS$variant
df <- as.data.frame(gr, row.names=NULL)
chr <- df$seqnames
start <- df$start
end <- df$end
data <- paste(chr,':',start,'-',end, sep='')

# b) define nearby genes taking into acount distance weight
# without gene scoring
df_nGenes <- xGR2nGenes(data=data, format="chr:start-end",
distance.max=10000, decay.kernel="slow", decay.exponent=2,
RData.location=RData.location)
# with their scores
df_nGenes <- xGR2nGenes(data=data, format="chr:start-end",
distance.max=10000, decay.kernel="slow", decay.exponent=2,
scoring=TRUE, scoring.scheme="max", RData.location=RData.location)

# c) define nearby genes without taking into acount distance weight
# without gene scoring
df_nGenes <- xGR2nGenes(data=data, format="chr:start-end",
distance.max=10000, decay.kernel="constant",
RData.location=RData.location)
# with their scores
df_nGenes <- xGR2nGenes(data=data, format="chr:start-end",
distance.max=10000, decay.kernel="constant", scoring=TRUE,
scoring.scheme="max", RData.location=RData.location)

## End(Not run)
```

---

xGR2xGenes *Function to define genes from an input list of genomic regions given the crosslink info*

Description

xGR2xGenes is supposed to define genes crosslinking to an input list of genomic regions (GR). Also required is the crosslink info with a score quantifying the link of a GR to a gene. Currently supported built-in crosslink info is enhancer genes, eQTL genes, conformation genes and nearby genes (purely), though the user can customise it via 'crosslink.customised'; if so, it has priority over the built-in data.

Usage

```
xGR2xGenes(
data,
format = c("chr:start-end", "data.frame", "bed", "GRanges"),
build.conversion = c(NA, "hg38.to.hg19", "hg18.to.hg19"),
crosslink = c("genehancer", "PCHiC_PMID27863249_combined",
"GTEx_V6p_combined",
"nearby"),
crosslink.customised = NULL,
cdf.function = c("original", "empirical"),
scoring = FALSE,
scoring.scheme = c("max", "sum", "sequential"),
scoring.rescale = FALSE,
nearby.distance.max = 50000,
nearby.decay.kernel = c("rapid", "slow", "linear", "constant"),
nearby.decay.exponent = 2,
verbose = TRUE,
silent = FALSE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

data	input genomic regions (GR). If formatted as "chr:start-end" (see the next parameter 'format' below), GR should be provided as a vector in the format of 'chrN:start-end', where N is either 1-22 or X, start (or end) is genomic positional number; for example, 'chr1:13-20'. If formatted as a 'data.frame', the first three columns correspond to the chromosome (1st column), the starting chromosome position (2nd column), and the ending chromosome position (3rd column). If the format is indicated as 'bed' (browser extensible data), the same as 'data.frame' format but the position is 0-based offset from chromomose position. If the genomic regions provided are not ranged but only the single position, the ending chromosome position (3rd column) is allowed not to be provided. The data could also be an object of 'GRanges' (in this case, formatted as 'GRanges')
format	the format of the input data. It can be one of "data.frame", "chr:start-end", "bed" or "GRanges"
build.conversion	the conversion from one genome build to another. The conversions supported are "hg38.to.hg19" and "hg18.to.hg19". By default it is NA (no need to do so)

`crosslink` the built-in crosslink info with a score quantifying the link of a GR to a gene. It can be one of 'genehancer' (enhancer genes; PMID:28605766), 'nearby' (nearby genes; if so, please also specify the relevant parameters 'nearby.distance.max', 'nearby.decay.kernel' and 'nearby.decay.exponent' below), 'PCHiC_PMID27863249_combined' (conformation genes; PMID:27863249), 'PCHiC_PMID31501517_combined' (conformation genes; PMID:31501517), 'GTEx_V6p_combined' (eQTL genes; PMID:29022597), 'eQTL_scRNAseq_combined' (eQTL genes; PMID:29610479), 'eQTL_jpRNAseq_combined' (eQTL genes; PMID:28553958), 'eQTL_ImmuneCells_combined' (eQTL genes; PMID:24604202,22446964,26151758,28248954,24013639), 'eQTL_DICE_combined' (eQTL genes; PMID:30449622)

`crosslink.customised` the crosslink info with a score quantifying the link of a GR to a gene. A user-input matrix or data frame with 4 columns: 1st column for genomic regions (formatted as "chr:start-end", genome build 19), 2nd column for Genes, 3rd for crosslink score (crosslinking a genomic region to a gene, such as -log10 significance level), and 4th for contexts (optional; if not provided, it will be added as 'C'). Alternatively, it can be a file containing these 4 columns. Required, otherwise it will return NULL

`cdf.function` a character specifying how to transform the input crosslink score. It can be one of 'original' (no such transformation), and 'empirical' for looking at empirical Cumulative Distribution Function (cdf; as such it is converted into pvalue-like values [0,1])

`scoring` logical to indicate whether gene-level scoring will be further calculated. By default, it sets to false

`scoring.scheme` the method used to calculate seed gene scores under a set of GR. It can be one of "sum" for adding up, "max" for the maximum, and "sequential" for the sequential weighting. The sequential weighting is done via: $\sum_{i=1} \frac{R_i}{i}$, where R_i is the i^{th} rank (in a descreasing order)

`scoring.rescale` logical to indicate whether gene scores will be further rescaled into the [0,1] range. By default, it sets to false

`nearby.distance.max` the maximum distance between genes and GR. Only those genes no far way from this distance will be considered as seed genes. This parameter will influence the distance-component weights calculated for nearby GR per gene

`nearby.decay.kernel` a character specifying a decay kernel function. It can be one of 'slow' for slow decay, 'linear' for linear decay, and 'rapid' for rapid decay. If no distance weight is used, please select 'constant'

`nearby.decay.exponent` a numeric specifying a decay exponent. By default, it sets to 2

`verbose` logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

`silent` logical to indicate whether the messages will be silent completely. By default, it sets to false. If true, verbose will be forced to be false

RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details
guid	a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details

Value

If scoring sets to false, a data frame with following columns:

- GR: genomic regions
- Gene: crosslinked genes
- Score: the original score between the gene and the GR (if cdf.function is 'original'); otherwise cdf (based on the whole crosslink inputs)
- Context: the context

If scoring sets to true, a data frame with following columns:

- Gene: crosslinked genes
- Score: gene score summarised over its list of crosslinked GR
- Pval: p-value-like significance level transformed from gene scores
- Context: the context

See Also

xGR, xRDataLoader, xSymbol2GeneID, xGR2nGenes

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:

# 1) provide the genomic regions
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
## get lead SNPs reported in AS GWAS and their significance info (p-values)
gr <- ImmunoBase$AS$variant
names(gr) <- NULL
dGR <- xGR(gr, format="GRanges")

# 2) using built-in crosslink info
## enhancer genes
df_xGenes <- xGR2xGenes(dGR, format="GRanges", crosslink="genehancer",
RData.location=RData.location)
## conformation genes
df_xGenes <- xGR2xGenes(dGR, format="GRanges",
crosslink="PCHiC_combined", RData.location=RData.location)
## eQTL genes
df_xGenes <- xGR2xGenes(dGR, format="GRanges",
crosslink="GTEx_V6p_combined", RData.location=RData.location)
```

```
## nearby genes (50kb, decaying rapidly)
df_xGenes <- xGR2xGenes(dGR, format="GRanges", crosslink="nearby",
nearby.distance.max=50000, nearby.decay.kernel="rapid",
RData.location=RData.location)

# 3) advanced use
# 3a) provide crosslink.customised
## illustration purpose only (see the content of 'crosslink.customised')
df <- xGR2nGenes(dGR, format="GRanges", RData.location=RData.location)
crosslink.customised <- data.frame(GR=df$GR, Gene=df$Gene,
Score=df$Weight, Context=rep('C',nrow(df)), stringsAsFactors=FALSE)
#crosslink.customised <- data.frame(GR=df$GR, Gene=df$Gene, Score=df$Weight, stringsAsFactors=FALSE)
# 3b) define crosslinking genes
# without gene scoring
df_xGenes <- xGR2xGenes(dGR, format="GRanges",
crosslink.customised=crosslink.customised,
RData.location=RData.location)
# with gene scoring
df_xGenes <- xGR2xGenes(dGR, format="GRanges",
crosslink.customised=crosslink.customised, scoring=TRUE,
scoring.scheme="max", RData.location=RData.location)

## End(Not run)
```

---

xGR2xGeneScores *Function to identify likely modulated seed genes from an input list of genomic regions together with the significance level given the crosslink info*

Description

xGR2xGeneScores is supposed to identify likely modulated seed genes from a list of genomic regions (GR) together with the significance level (measured as p-values or fdr). To do so, it defines seed genes and their scores given the crosslink info with a score quantifying the link of a GR to a gene. It returns an object of class "mSeed".

Usage

```
xGR2xGeneScores(
data,
significance.threshold = NULL,
score.cap = NULL,
build.conversion = c(NA, "hg38.to.hg19", "hg18.to.hg19"),
crosslink = c("genehancer", "PCHiC_combined", "GTEx_V6p_combined",
"nearby"),
crosslink.customised = NULL,
cdf.function = c("original", "empirical"),
scoring.scheme = c("max", "sum", "sequential"),
nearby.distance.max = 50000,
```

```
nearby.decay.kernel = c("rapid", "slow", "linear", "constant"),
nearby.decay.exponent = 2,
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

`data`	a named input vector containing the sinificance level for genomic regions (GR). For this named vector, the element names are GR, in the format of 'chrN:start-end', where N is either 1-22 or X, start (or end) is genomic positional number; for example, 'chr1:13-20'. The element values for the significance level (measured as p-value or fdr). Alternatively, it can be a matrix or data frame with two columns: 1st column for GR, 2nd column for the significance level.
`significance.threshold`	the given significance threshold. By default, it is set to NULL, meaning there is no constraint on the significance level when transforming the significance level of GR into scores. If given, those GR below this are considered significant and thus scored positively. Instead, those above this are considered insignificant and thus receive no score
`score.cap`	the maximum score being capped. By default, it is set to NULL, meaning that no capping is applied
`build.conversion`	the conversion from one genome build to another. The conversions supported are "hg38.to.hg19" and "hg18.to.hg19". By default it is NA (no need to do so)
`crosslink`	the built-in crosslink info with a score quantifying the link of a GR to a gene. See `xGR2xGenes` for details
`crosslink.customised`	the crosslink info with a score quantifying the link of a GR to a gene. A user-input matrix or data frame with 4 columns: 1st column for genomic regions (formatted as "chr:start-end", genome build 19), 2nd column for Genes, 3rd for crosslink score (crosslinking a genomic region to a gene, such as -log10 significance level), and 4th for contexts (optional; if nor provided, it will be added as 'C'). Alternatively, it can be a file containing these 4 columns. Required, otherwise it will return NULL
`cdf.function`	a character specifying how to transform the input crosslink score. It can be one of 'original' (no such transformation), and 'empirical' for looking at empirical Cumulative Distribution Function (cdf; as such it is converted into pvalue-like values [0,1])
`scoring.scheme`	the method used to calculate seed gene scores under a set of GR (also over Contexts if many). It can be one of "sum" for adding up, "max" for the maximum, and "sequential" for the sequential weighting. The sequential weighting is done via: $\sum_{i=1} \frac{R_i}{i}$, where R_i is the i^{th} rank (in a descreasing order)
`nearby.distance.max`	the maximum distance between genes and GR. Only those genes no far way from this distance will be considered as seed genes. This parameter will influence the distance-component weights calculated for nearby GR per gene

`nearby.decay.kernel`
: a character specifying a decay kernel function. It can be one of 'slow' for slow decay, 'linear' for linear decay, and 'rapid' for rapid decay. If no distance weight is used, please select 'constant'

`nearby.decay.exponent`
: a numeric specifying a decay exponent. By default, it sets to 2

`verbose`
: logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

`RData.location`
: the characters to tell the location of built-in RData files. See `xRDataLoader` for details

`guid`
: a valid (5-character) Global Unique IDentifier for an OSF project. See `xRDataLoader` for details

Value

an object of class "mSeed", a list with following components:

- `GR`: a matrix of nGR X 3 containing GR information, where nGR is the number of GR, and the 3 columns are "GR" (genomic regions), "Score" (the scores for GR calculated based on p-values taking into account the given threshold of the significant level), "Pval" (the input p-values for GR)
- `Gene`: a matrix of nGene X 3 containing Gene information, where nGene is the number of seed genes, and the 3 columns are "Gene" (gene symbol), "Score" (the scores for seed genes), "Pval" (p-value-like significance level transformed from gene scores)
- `Link`: a matrix of nLink X 5 containing GR-Gene link information, where nLink is the number of links, and the 5 columns are "GR" (genomic regions), "Gene" (gene symbol), "Score" (the scores for the link multiplied by the GR score), "Score_GR" (the scores for GR), "Score_link" (the original scores for the link if cdf.function is 'original'; otherwise cdf based on the whole crosslink inputs)

Note

This function uses `xGRscores` and `xGR2xGenes` to define and score seed genes from input genomic regions.

See Also

`xGRscores`, `xGR2xGenes`, `xGRsort`

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:

# a) provide the seed SNPs with the significance info
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
## get lead SNPs reported in AS GWAS and their significance info (p-values)
```

```
gr <- ImmunoBase$AS$variant
df <- as.data.frame(gr, row.names=NULL)
GR <- paste0(df$seqnames,':',df$start,'-',df$end)
data <- cbind(GR=GR, Sig=df$Pvalue)

# b) define and score seed geens
mSeed <- xGR2xGeneScores(data=data, crosslink="genehancer",
RData.location=RData.location)

## End(Not run)
```

`xGRscores` *Function to score genomic regions based on the given significance level*

Description

`xGRscores` is supposed to score a list of genomic regions together with the significance level.

Usage

```
xGRscores(data, significance.threshold = 0.05, score.cap = 10, verbose
= TRUE)
```

Arguments

`data`	a named input vector containing the sinificance level for genomic regions (GR). For this named vector, the element names are GR, in the format of 'chrN:start-end', where N is either 1-22 or X, start (or end) is genomic positional number; for example, 'chr1:13-20'. The element values for the significance level (measured as p-value or fdr). Alternatively, it can be a matrix or data frame with two columns: 1st column for GR, 2nd column for the significance level.
`significance.threshold`	the given significance threshold. By default, it is set to NULL, meaning there is no constraint on the significance level when transforming the significance level of GR into scores. If given, those GR below this are considered significant and thus scored positively. Instead, those above this are considered insigificant and thus receive no score
`score.cap`	the maximum score being capped. By default, it is set to 10. If NULL, no capping is applied
`verbose`	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

a data frame with following columns:

- GR: genomic regions
- Score: the scores for GR calculated based on p-values taking into account the given threshold of the significant level
- Pval: the input p-values for GR

Note

None

See Also

xGRscores

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:

# a) provide the seed SNPs with the significance info
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
## get lead SNPs reported in AS GWAS and their significance info (p-values)
gr <- ImmunoBase$AS$variant
df <- as.data.frame(gr, row.names=NULL)
chr <- df$seqnames
start <- df$start
end <- df$end
sig <- df$Pvalue
GR <- paste(chr,':',start,'-',end, sep='')
data <- cbind(GR=GR, Sig=sig)

# b) calculate GR scores (considering significant cutoff 5e-5)
df_GR <- xGRscores(data=data, significance.threshold=5e-5)

## End(Not run)
```

---

xGRsort *Function to sort by chromosomes/seqnames, start and end coordinates of the intervals.*

Description

xGRsort is supposed to sort by chromosomes/seqnames, start and end coordinates of the intervals.

Usage

```
xGRsort(data)
```

Arguments

data	input genomic regions (GR). GR should be provided as a vector in the format of 'chrN:start-end', where N is either 1-22 or X, start (or end) is genomic positional number; for example, 'chr1:13-20'

Value

index

See Also

xGR

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:

# a) provide the genomic regions
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
## get lead SNPs reported in AS GWAS and their significance info (p-values)
gr <- ImmunoBase$AS$variant
cse <- xGRcse(gr)

# b) sort index
ind <- xGRsort(cse)
data <- cse[ind]

## End(Not run)
```

---

xGSEAbarplot *Function to visualise GSEA results using a barplot*

Description

xGSEAbarplot is supposed to visualise GSEA results using a barplot. It returns an object of class "ggplot".

Usage

```
xGSEAbarplot(
eGSEA,
top_num = 10,
displayBy = c("nes", "adjp", "fdr", "pvalue"),
FDR.cutoff = 0.05,
bar.label = TRUE,
bar.label.size = 3,
bar.color = "lightyellow-orange",
bar.width = 0.8,
wrap.width = NULL,
font.family = "sans",
signature = TRUE
)
```

Arguments

eGSEA	an object of class "eGSEA"
top_num	the number of the top terms (sorted according to FDR or adjusted p-values). If it is 'auto', only the significant terms (see below FDR.cutoff) will be displayed
displayBy	which statistics will be used for displaying. It can be "nes" for normalised enrichment score (by default), "adjp" or "fdr" for adjusted p value (or FDR), "pvalue" for p value
FDR.cutoff	FDR cutoff used to declare the significant terms. By default, it is set to 0.05. This option only works when setting top_num (see above) is 'auto'
bar.label	logical to indicate whether to label each bar with FDR. By default, it sets to true for bar labelling
bar.label.size	an integer specifying the bar labelling text size. By default, it sets to 3
bar.color	either NULL or fill color names ('lightyellow-orange' by default)
bar.width	bar width. By default, 80 data
wrap.width	a positive integer specifying wrap width of name
font.family	the font family for texts
signature	logical to indicate whether the signature is assigned to the plot caption. By default, it sets TRUE showing which function is used to draw this graph

Value

an object of class "ggplot"

Note

none

See Also

xGSEAbarplot

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
bp <- xGSEAbarplot(eGSEA, top_num="auto", displayBy="nes")
#pdf(file="GSEA_barplot.pdf", height=6, width=12, compress=TRUE)
print(bp)
#dev.off()

## End(Not run)
```

---

xGSEAconciser *Function to make GSEA results conciser by removing redundant terms*

Description

xGSEAconciser is supposed to make GSEA results conciser by removing redundant terms. A redundant term (called 'B') is claimed if its overlapped part (A&B) with a more significant term (called 'A') meets both criteria: 1) |A&B| > 0.9*|B|; and 2) |A&B| > 0.5*|A|.

Usage

```
xGSEAconciser(eGSEA, cutoff = c(0.9, 0.5), verbose = TRUE)
```

Arguments

eGSEA	an object of class "eGSEA"
cutoff	a cutoff vector used to remove redundant terms. By default, it has the first element 0.9 and the second element 0.5. It means, for a term (less significant; called 'B'), if there is a more significant term (called 'A'), their overlapped members cover at least 90 this term B will be defined as redundant and thus being removed
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to false for no display

Value

an object of class "eGSEA", after redundant terms being removed.

Note

none

See Also

xGSEAconciser

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
eGSEA_concise <- xGSEAconciser(eGSEA)

## End(Not run)
```

---

xGSEAdotplot *Function to visualise GSEA results using dot plot*

Description

xGSEAdotplot is supposed to visualise GSEA results using dot plot. It returns an object of class "ggplot" or a list of "ggplot" objects.

Usage

```
xGSEAdotplot(
eGSEA,
top = 1,
colormap = "lightblue-darkblue",
zlim = NULL,
ncolors = 64,
xlab = NULL,
title = c("name", "setID", "none"),
subtitle = c("leading", "enrichment", "both", "none"),
clab = "Pi rating",
x.scale = c("normal", "sqrt", "log"),
peak = TRUE,
peak.color = "red",
leading = FALSE,
leading.size = 2,
leading.color = "black",
leading.alpha = 0.6,
leading.padding = 0.2,
leading.arrow = 0.01,
leading.force = 0.01,
leading.query = NULL,
leading.query.only = FALSE,
leading.edge.only = FALSE,
compact = FALSE,
font.family = "sans",
signature = TRUE,
...
)
```

Arguments

eGSEA	an object of class "eGSEA"
top	the number of the top enrichments to be visualised. Alternatively, the gene set names can be queried
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
zlim	the minimum and maximum z values for which colors should be plotted
ncolors	the number of colors specified over the colormap
xlab	the label for x-axis. If NULL, it is 'Target ranks'
title	the title. If NULL, it is term name followed by the number of its annotations
subtitle	the subtitle. It can be used to show 'leading' info, 'enrichment' info or 'both'
clab	the label for colorbar. By default, it is '5-star ratings'
x.scale	how to transform the x scale. It can be "normal" for no transformation, "sqrt" for square root transformation, and "log" for log-based transformation
peak	logical to indicate whether the peak location is shown
peak.color	the peak color
leading	logical to indicate whether the leading targets are texted. Alterntively, leading can be numeric to restict the top targets displayed
leading.size	the size of leading targets' texts. It only works when the parameter 'leading' is enabled
leading.color	the label color of leading targets' texts
leading.alpha	the 0-1 value specifying transparency of leading targets' texts
leading.padding	the padding around the leading targets' texts
leading.arrow	the arrow pointing to the leading targets
leading.force	the repelling force between leading targets' texts
leading.query	which genes in query will be labelled. By default, it sets to NULL meaning all genes will be displayed. If labels in query can not be found, then all will be displayed
leading.query.only	logical to indicate whether only genes in query will be displayed. By default, it sets to FALSE. It only works when labels in query are enabled/found
leading.edge.only	logical to indicate whether only the leading edge will be shown. By default, it sets to FALSE

compact	logical to indicate whether the compact/void theme is used. If TRUE, axes and legend info will be hidden
font.family	the font family for texts
signature	logical to indicate whether the signature is assigned to the plot caption. By default, it sets TRUE showing which function is used to draw this graph
...	additional paramters associated with ggrepel::geom_text_repel. If queried, it has high priority (eg, color='darkred',size=2,alpha=0.6,fontface='bold')

Value

an object of class "ggplot" or a list of "ggplot" objects.

Note

none

See Also

xGSEAdotplot

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
gp <- xGSEAdotplot(eGSEA, top=1)
#gp <- xGSEAdotplot(eGSEA, top=1, peak=FALSE, compact=TRUE, signature=FALSE)
gp

ls_gp <- xGSEAdotplot(eGSEA, top=1:4, signature=FALSE)
library(gridExtra)
grid.arrange(grobs=ls_gp, ncol=2)

## End(Not run)
```

---

xGSsimulator *Function to simulate gold standard negatives (GSN) given gold standard positives (GSP) and a gene network*

Description

xGSsimulator is supposed to simulate gold standard negatives (GSN) given gold standard positives (GSP) and an input gene network. GSN targets are those after excluding GSP targets and their 1-order (by default) neighbors in the gene network.

Usage

```
xGSsimulator(
GSP,
population = NULL,
network = c("STRING_medium", "STRING_low", "STRING_high",
"STRING_highest",
"PCommonsUN_high", "PCommonsUN_medium")[c(1, 6)],
network.customised = NULL,
neighbor.order = 1,
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

GSP	a vector containing Gold Standard Positives (GSP)
population	a vector containing population space in which gold standard negatives (GSN) will be considered. By default, it is NULL, meaning genes in the network will be used instead
network	the built-in network. Currently two sources of network information are supported: the STRING database (version 10) and the Pathways Commons database (version 7). STRING is a meta-integration of undirect interactions from the functional aspect, while Pathways Commons mainly contains both undirect and direct interactions from the physical/pathway aspect. Both have scores to control the confidence of interactions. Therefore, the user can choose the different quality of the interactions. In STRING, "STRING_highest" indicates interactions with highest confidence (confidence scores>=900), "STRING_high" for interactions with high confidence (confidence scores>=700), "STRING_medium" for interactions with medium confidence (confidence scores>=400), and "STRING_low" for interactions with low confidence (confidence scores>=150). For undirect/physical interactions from Pathways Commons, "PCommonsUN_high" indicates undirect interactions with high confidence (supported with the PubMed references plus at least 2 different sources), "PCommonsUN_medium" for undirect interactions with medium confidence (supported with the PubMed references). By default, "STRING_medium" and "PCommonsUN_medium" are used
network.customised	an object of class "igraph". By default, it is NULL. It is designed to allow the user analysing their customised network data that are not listed in the above argument 'network'. This customisation (if provided) has the high priority over built-in network
neighbor.order	an integer giving the order of the neighborhood. By default, it is 1-order neighborhood
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details

guid — a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details

Value

a list with following components:

- GSN: a vector containing simulated GSN
- GSP: a vector containing GSP after considering the population space
- g: an "igraph" object

Note

If multiple graphs are provided, they will be unionised for use.

See Also

xDefineNet

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
sGS <- xGSsimulator(GSP, population=NULL,
network=c("STRING_medium","PCommonsUN_medium"),
RData.location=RData.location)

## End(Not run)
```

---

xHeatmap *Function to draw heatmap using ggplot2*

Description

xHeatmap is supposed to draw heatmap using ggplot2.

Usage

```
xHeatmap(
data,
reorder = c("none", "row", "col", "both"),
colormap = "spectral",
ncolors = 64,
zlim = NULL,
barwidth = 0.3,
barheight = NULL,
nbin = 64,
legend.title = "",
```

```
x.rotate = 45,
x.text.size = 6,
x.text.hjust = 0,
y.text.size = 6,
legend.text.size = 4,
legend.title.size = 6,
shape = 19,
size = 2,
plot.margin = unit(c(5.5, 5.5, 5.5, 5.5), "pt"),
font.family = "sans",
na.color = "transparent",
data.label = NULL,
label.size = 1,
label.color = "black",
...
)
```

Arguments

data	a data frame/matrix for coloring. The coloring can be continuous (numeric matrix) or discrete (factor matrix)
reorder	how to reorder rows and columns. It can be "none" for no reordering, "row" for reordering rows according to number of sharings (by default), "col" for reordering columns, and "both" for reordering rows and columns
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
ncolors	the number of colors specified over the colormap
zlim	the minimum and maximum z values for which colors should be plotted, defaulting to the range of the finite values of displayed matrix
barwidth	the width of the colorbar. Default value is 'legend.key.width' or 'legend.key.size' in 'theme' or theme
barheight	the height of the colorbar. Default value is 'legend.key.height' or 'legend.key.size' in 'theme' or theme
nbin	the number of bins for drawing colorbar
legend.title	the title of the colorbar. By default, it is ''
x.rotate	the angle to rotate the x tick labelings. By default, it is 60
x.text.size	the text size of the x tick labelings. By default, it is 6
x.text.hjust	the hjust of the x tick labelings. By default, it is 0.5
y.text.size	the text size of the y tick labelings. By default, it is 6

legend.text.size	the text size of the legend tick labelings. By default, it is 5
legend.title.size	the text size of the legend titles. By default, it is 6
shape	the number specifying the shape. By default, it is 19
size	the number specifying the shape size. By default, it is 2
plot.margin	the margin (t, r, b, l) around plot. By default, it is unit(c(5.5,5.5,5.5,5.5),"pt")
font.family	the font family for texts
na.color	the color for NAs. By default, it is 'transparent'
data.label	a data frame/matrix used for the labelling
label.size	the label size
label.color	the label color
...	additional graphic parameters for supraHex::visTreeBootstrap

Value

a ggplot2 object

Note

none

See Also

xHeatmap

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
data(mtcars)
gp <- xHeatmap(mtcars, reorder="none", colormap='jet.top', x.rotate=45,
shape=19, size=3, x.text.size=8,y.text.size=8, legend.title='mtcars')
gp + theme(legend.position="bottom",legend.direction="horizontal") +
guides(color=guide_colorbar(title="mtcars",title.position="top",barwidth=5,barheight=0.3))
gp + theme(legend.position="bottom",legend.direction="horizontal") +
guides(color=guide_legend(title="mtcars",title.position="top",barwidth=5,barheight=0.3))
gp + geom_text(aes(x, y,
label=val),size=1.8,color='black',fontface='bold',na.rm=TRUE,angle=45)

## End(Not run)
```

xLayout *Function to define graph node coordinates according to igraph- or sna-style layout*

Description

xLayout is supposed to define graph node coordinates according to igraph- or sna-style layout.

Usage

```
xLayout(
g,
layout = c("layout_nicely", "layout_randomly", "layout_in_circle",
"layout_on_sphere", "layout_with_fr", "layout_with_kk",
"layout_as_tree",
"layout_with_lgl", "layout_with_graphopt", "layout_with_sugiyama",
"layout_with_dh",
"layout_with_drl", "layout_with_gem", "layout_with_mds",
"layout_as_bipartite",
"gplot.layout.adj", "gplot.layout.circle", "gplot.layout.circrand",
"gplot.layout.eigen", "gplot.layout.fruchtermanreingold",
"gplot.layout.geodist",
"gplot.layout.hall", "gplot.layout.kamadakawai", "gplot.layout.mds",
"gplot.layout.princoord", "gplot.layout.random", "gplot.layout.rmds",
"gplot.layout.segeo", "gplot.layout.seham", "gplot.layout.spring",
"gplot.layout.springrepulse", "gplot.layout.target",
"graphlayouts.layout_with_stress", "graphlayouts.layout_as_backbone",
"gephi.forceatlas2"),
seed = 825,
flip = F
)
```

Arguments

g	an object of class "igraph" (or "graphNEL") for a graph
layout	a character specifying graph layout function. This character can be used to indicate igraph-style layout ("layout_nicely","layout_randomly","layout_in_circle","layout_on_sphere","layo or sna-style layout ("gplot.layout.adj","gplot.layout.circle","gplot.layout.circrand","gplot.layout.eigen","g or graphlayouts-style layout ("graphlayouts.layout_with_stress","graphlayouts.layout_as_backbone"), or ForeceAtlas2 layout used in Dephi ("gephi.forceatlas2")
seed	an integer specifying the seed
flip	logical to indicate whether x- and y-coordiates flip. By default, it sets to false

Value

It returns an igraph object, appended by node attributes including "xcoord" for x-coordinates, "ycoord" for y-coordiates.

See Also

xGGnetwork

Examples

```
## Not run:
# Load the library
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata/"

# load REACTOME
# restricted to Immune System ('R-HSA-168256') or Signal Transduction ('R-HSA-162582')
g <- xRDataLoader('ig.REACTOME', RData.location=RData.location)
neighs.out <- igraph::neighborhood(g, order=vcount(g),
nodes="R-HSA-168256", mode="out")
nodeInduced <- V(g)[unique(unlist(neighs.out))]$name
ig <- igraph::induced.subgraph(g, vids=nodeInduced)

# compare Fruchterman and Reingold force-directed placement algorithm
## based on igraph layout
ig1 <- xLayout(ig, layout="layout_with_fr")
gp1 <- xGGnetwork(ig1, node.xcoord="xcoord", node.ycoord="ycoord")
## based on sna layout
ig2 <- xLayout(ig, layout="gplot.layout.fruchtermanreingold")
gp2 <- xGGnetwork(ig2, node.xcoord="xcoord", node.ycoord="ycoord")

# compare Kamada-Kawai force-directed placement algorithm
## based on igraph layout
ig1 <- xLayout(ig, layout="layout_with_kk")
gp1 <- xGGnetwork(ig1, node.xcoord="xcoord", node.ycoord="ycoord")
## based on sna layout
ig2 <- xLayout(ig, layout="gplot.layout.kamadakawai")
gp2 <- xGGnetwork(ig2, node.xcoord="xcoord", node.ycoord="ycoord")
## do together
layouts <-
c("layout_with_fr","gplot.layout.fruchtermanreingold","layout_with_kk","gplot.layout.kamadakawai",
"gephi.forceatlas2")
ls_ig <- lapply(layouts, function(x) xLayout(ig, layout=x))
names(ls_ig) <- layouts
gp <- xGGnetwork(ls_ig, node.xcoord='xcoord', node.ycoord='ycoord',
ncolumns=5)

## End(Not run)
```

`xLiftOver` *Function to lift genomic intervals from one genome build to another.*

Description

`xLiftOver` is supposed to lift genomic intervals from one genome build to another. Supported are the conversions between genome builds 'hg38' (GRCh38), 'hg19' (GRCh37) and 'h18'.

Usage

```
xLiftOver(
data.file,
format.file = c("data.frame", "bed", "chr:start-end", "GRanges"),
build.conversion = c(NA, "hg38.to.hg19", "hg19.to.hg38",
"hg19.to.hg18",
"hg18.to.hg38", "hg18.to.hg19"),
merged = TRUE,
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

data.file	an input data file, containing a list of genomic regions to test. If the input file is formatted as a 'data.frame' (specified by the parameter 'format.file' below), the first three columns correspond to the chromosome (1st column), the starting chromosome position (2nd column), and the ending chromosome position (3rd column). If the format is indicated as 'bed' (browser extensible data), the same as 'data.frame' format but the position is 0-based offset from chromomose position. If the genomic regions provided are not ranged but only the single position, the ending chromosome position (3rd column) is allowed not to be provided. If the format is indicated as "chr:start-end", instead of using the first 3 columns, only the first column will be used and processed. If the file also contains other columns, these additional columns will be ignored. Alternatively, the input file can be the content itself assuming that input file has been read. Note: the file should use the tab delimiter as the field separator between columns
format.file	the format for input files. It can be one of "data.frame", "chr:start-end", "bed"
build.conversion	the conversion from one genome build to another. The conversions supported are "hg38.to.hg19", "hg19.to.hg38", "hg19.to.hg18", "hg18.to.hg38" and "hg18.to.hg19". By default it is NA, forcing the user to specify the corrent one.
merged	logical to indicate whether multiple ranges should be merged into the one per a range in query. By default, it sets to true
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to false for no display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details
guid	a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details

Value

an GR oject storing converted genomic intervals.

See Also

xRDataLoader

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:

# Provide UCSC known genes (hg19)
UCSC_genes <- xRDataLoader('UCSC_knownGene',
RData.location=RData.location)
UCSC_genes

# Lift over to hg38
gr <- xLiftOver(UCSC_genes, format.file="GRanges",
build.conversion="hg19.to.hg38", RData.location=RData.location)
gr

## End(Not run)
```

---

xMEabf *Function to conduct colocalisation analysis through Wakefield's Approximate Bayes Factor approach integrating GWAS and eQTL summary data*

Description

xMEabf is supposed to conduct colocalisation analysis integrating GWAS and eQTL summary data through Wakefield's Approximate Bayes Factor (ABF).

Usage

```
xMEabf(
eqtl.summary,
gwas.summary,
prior.eqtl = 1e-04,
prior.gwas = 1e-04,
prior.both = 1e-05
)
```

Arguments

eqtl.summary	an input eQTL summary data for a region (eg the eQTLs for a gene), a list with mandatory components 'beta' (a vector for eQTL effect size), 'varbeta' (a vector for beta variance), 'N' (an integer specifying number of samples), 'MAF' (minor allele frequency, eg effect allele frequency), 'snp' (a vector for dbSNP identity)

gwas.summary	an input GWAS summary data, a list with mandatory components 'beta' (a vector for GWAS SNP effect size), 'varbeta' (a vector for beta variance), 'snp' (a vector for dbSNP identity)
prior.eqtl	the prior probability an eQTL associated with the eQTL trait. The default value is 1e-4
prior.gwas	the prior probability an SNP associated with the GWAS trait. The default value is 1e-4
prior.both	the prior probability an eQTL/SNP associated with both eQTL/GWAS traits. The default value is 1e-5

Value

a list with two compenents (1) the component 'summary', a vector of 'nsnps' (number of SNPs analysed), 'PP.H0.abf' (posterior probabilities of H0 - no causal variant), 'PP.H1.abf' (posterior probabilities of H1 - causal variant for eQTL trait only), 'PP.H2.abf' (posterior probabilities of H2 - causal variant for GWAS trait only), 'PP.H3.abf' (posterior probabilities of H3 - two distinct causal variants), and 'PP.H4.abf' (posterior probabilities of H4 - one shared causal variant), and (2) the component 'results', a data frame with a column 'snp' (SNPs analysed), columns for eQTL statistics calcualted ('eqtl.V', 'eqtl.z', 'eqtl.r' and 'eqtl.lABF'), columns for GWAS statistics calculated ('gwas.V', 'gwas.z', 'gwas.r' and 'gwas.lABF'), a column 'both.sum.lABF' (the sum of 'eqtl.lABF' and 'gwas.lABF') and a column 'SNP.PP.H4' (the posterior probability of the SNP being causal for both traits).

See Also

xMEabf

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
res <- xMEabf(eqtl.summary, gwas.summary)
utils::write.table(res$results, file="df_abf.txt", row.names=FALSE,
col.names=TRUE, quote=FALSE, sep="\t")

## End(Not run)
```

---

xMLcaret *Function to integrate predictor matrix in a supervised manner via machine learning algorithms using caret.*

Description

xMLcaret is supposed to integrate predictor matrix in a supervised manner via machine learning algorithms using caret. The caret package streamlines model building and performance evaluation. It requires three inputs: 1) Gold Standard Positive (GSP) targets; 2) Gold Standard Negative (GSN) targets; 3) a predictor matrix containing genes in rows and predictors in columns, with their predictive scores inside it. It returns an object of class 'sTarget'.

Usage

```
xMLcaret(
list_pNode = NULL,
df_predictor = NULL,
GSP,
GSN,
method = c("gbm", "svmRadial", "rda", "knn", "pls", "nnet", "rf",
"myrf", "cforest",
"glmnet", "glm", "bayesglm", "LogitBoost", "xgbLinear", "xgbTree"),
nfold = 3,
nrepeat = 10,
seed = 825,
aggregateBy = c("none", "logistic", "Ztransform", "fishers",
"orderStatistic"),
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

list_pNode	a list of "pNode" objects or a "pNode" object
df_predictor	a data frame containing genes (in rows) and predictors (in columns), with their predictive scores inside it. This data frame must has gene symbols as row names
GSP	a vector containing Gold Standard Positive (GSP)
GSN	a vector containing Gold Standard Negative (GSN)
method	machine learning method. It can be one of "gbm" for Gradient Boosting Machine (GBM), "svmRadial" for Support Vector Machines with Radial Basis Function Kernel (SVM), "rda" for Regularized Discriminant Analysis (RDA), "knn" for k-nearest neighbor (KNN), "pls" for Partial Least Squares (PLS), "nnet" for Neural Network (NNET), "rf" for Random Forest (RF), "myrf" for customised Random Forest (RF), "cforest" for Conditional Inference Random Forest, "glmnet" for glmnet, "glm" for Generalized Linear Model (GLM), "bayesglm" for Bayesian Generalized Linear Model (BGLM), "LogitBoost" for Boosted Logistic Regression (BLR), "xgbLinear" for eXtreme Gradient Boosting as linear booster (XGBL), "xgbTree" for eXtreme Gradient Boosting as tree booster (XGBT)
nfold	an integer specifying the number of folds for cross validataion. Per fold creates balanced splits of the data preserving the overall distribution for each class (GSP and GSN), therefore generating balanced cross-vallidation train sets and testing sets. By default, it is 3 meaning 3-fold cross validation
nrepeat	an integer specifying the number of repeats for cross validataion. By default, it is 10 indicating the cross-validation repeated 10 times
seed	an integer specifying the seed
aggregateBy	the aggregate method used to aggregate results from repeated cross validataion. It can be either "none" for no aggregration (meaning the best model based on all

	data used for cross validation is used), or "orderStatistic" for the method based on the order statistics of p-values, or "fishers" for Fisher's method, "Ztransform" for Z-transform method, "logistic" for the logistic method. Without loss of generality, the Z-transform method does well in problems where evidence against the combined null is spread widely (equal footings) or when the total evidence is weak; Fisher's method does best in problems where the evidence is concentrated in a relatively small fraction of the individual tests or when the evidence is at least moderately strong; the logistic method provides a compromise between these two. Notably, the aggregate methods 'Ztransform' and 'logistic' are preferred here
`verbose`	logical to indicate whether the messages will be displayed in the screen. By default, it sets to TRUE for display
`RData.location`	the characters to tell the location of built-in RData files. See `xRDataLoader` for details
`guid`	a valid (5-character) Global Unique IDentifier for an OSF project. See `xRDataLoader` for details

Value

an object of class "sTarget", a list with following components:

- `model`: an object of class "train" as a best model
- `ls_model`: a list of best models from repeated cross-validation
- `priority`: a data frame of n X 5 containing gene priority information, where n is the number of genes in the input data frame, and the 5 columns are "GS" (either 'GSP', or 'GSN', or 'Putative'), "name" (gene names), "rank" (priority rank), "rating" (5-star priority score/rating), and "description" (gene description)
- `predictor`: a data frame, which is the same as the input data frame but inserting two additional columns ('GS' in the first column, 'name' in the second column)
- `performance`: a data frame of 1+nPredictor X 2 containing the supervised/predictor performance info, where nPredictor is the number of predictors, two columns are "ROC" (AUC values) and "Fmax" (F-max values)
- `performance_cv`: a data frame of nfold*nrepeat X 2 containing the repeated cross-validation performance, where two columns are "ROC" (AUC values) and "Fmax" (F-max values)
- `importance`: a data frame of nPredictor X 1 containing the predictor importance info
- `gp`: a ggplot object for the ROC curve
- `gp_cv`: a ggplot object for the ROC curves from repeated cross-validation
- `evidence`: an object of the class "eTarget", a list with following components "evidence" and "metag"
- `list_pNode`: a list of "pNode" objects

Note

It will depend on whether a package "caret" and its suggested packages have been installed. It can be installed via: `BiocManager::install(c("caret","e1071","gbm","kernlab","klaR","pls","nnet","randomForest"`

See Also

xPierMatrix, xPredictROCR, xPredictCompare, xSparseMatrix, xSymbol2GeneID

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
sTarget <- xMLcaret(df_prediction, GSP, GSN, method="myrf")

## End(Not run)
```

`xMLcompare` *Function to visualise cross-validation performance against tuning parameters*

Description

`xMLcompare` is supposed to visualise cross-validation performance against tuning parameters.

Usage

```
xMLcompare(
list_ML,
metric = c("ROC", "Sens", "Spec"),
xlab = NA,
xlimits = c(0.5, 1),
font.family = "sans"
)
```

Arguments

`list_ML`	a list of class "train" or "train.formula" objects (resulting from caret::train)
`metric`	the performance metric to plot. It can be one of 'ROC', 'Sens' (Sensitivity) and 'Spec' (Specificity)
`xlab`	a title for the x axis
`xlimits`	the limit for the x axis
`font.family`	the font family for texts

Value

an object of class "ggplot"

Note

none

See Also

xMLcompare

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
gp <- xMLcompare(ls_ML, xlimits=c(0.5,1))

## End(Not run)
```

---

xMLdensity *Function to visualise machine learning results using density plot*

Description

xMLdensity is supposed to visualise machine learning results using density plot. It returns an object of class "ggplot".

Usage

```
xMLdensity(
xTarget,
displayBy = c("All", "GS", "GSN", "GSP", "NEW"),
x.scale = c("sqrt", "normal"),
font.family = "sans",
signature = TRUE
)
```

Arguments

xTarget	an object of class "xTarget" or "dTarget" (with the component 'pPerf')
displayBy	which targets will be used for displaying. It can be one of "GS" for gold standard targets, "GSN" for gold standard negatives, "GSP" for gold standard positives, "NEW" for putative/new targets (non-GS), "All" for all targets (by default)
x.scale	how to transform the x scale. It can be "normal" for no transformation, and "sqrt" for square root transformation (by default)
font.family	the font family for texts
signature	logical to indicate whether the signature is assigned to the plot caption. By default, it sets TRUE showing which function is used to draw this graph

Value

an object of class "ggplot"

Note

none

See Also

xColormap

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
gp <- xMLdensity(xTarget, displayBy="All")
gp

## End(Not run)
```

---

xMLdotplot *Function to visualise machine learning results using dot plot*

Description

xMLdotplot is supposed to visualise machine learning results using dot plot. It returns an object of class "ggplot".

Usage

```
xMLdotplot(
sTarget,
displayBy = c("importance2fold", "roc2fold", "fmax2fold",
"importance_accurancy",
"importance_gini", "ROC", "Fmax"),
ML.included = TRUE,
font.family = "sans",
signature = TRUE
)
```

Arguments

sTarget	an object of class "sTarget"
displayBy	which statistics will be used for displaying. It can be either statistics across folds ("importance2fold" for predictor importance, "roc2fold" for AUC in ROC, "fmax2fold" for F-max in Precision-Recall curve) or overall statistics ("importance_accurancy" for predictor importance measured by accuracy decrease, "importance_gini" for predictor importance measured by Gini decrease, "ROC" for AUC in ROC, "Fmax" for F-max in Precision-Recall curve)
ML.included	logical to indicate whether to use ML results
font.family	the font family for texts

signature — logical to indicate whether the signature is assigned to the plot caption. By default, it sets TRUE showing which function is used to draw this graph

Value

an object of class "ggplot"

Note

none

See Also

xMLdotplot

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
gp <- xMLdotplot(sTarget, displayBy="importance_accurancy")
gp

## End(Not run)
```

---

xMLfeatureplot *Function to visualise/assess features used for machine learning*

Description

xMLfeatureplot is supposed to visualise/assess features used for machine learning. Visualisation can be made using either boxplot or dot plot for AUC and F-max. It returns an object of class "ggplot" for AUC and F-max, and an object of class "trellis" for boxplot.

Usage

```
xMLfeatureplot(
df_predictor,
GSP,
GSN,
displayBy = c("boxplot", "ROC", "Fmax"),
font.family = "sans",
...
)
```

Arguments

df_predictor	a data frame containing genes (in rows) and predictors (in columns), with their predictive scores inside it. This data frame must has gene symbols as row names
GSP	a vector containing Gold Standard Positive (GSP)
GSN	a vector containing Gold Standard Negative (GSN)
displayBy	which statistics will be used for displaying. It can be either "boxplot" for features themselves, "ROC" for AUC in ROC, "Fmax" for F-max in Precision-Recall curve)
font.family	the font family for texts
...	additional parameters. Please refer to 'lattice::bwplot' for the complete list.

Value

an object of class "ggplot" for AUC and F-max, and an object of class "trellis" for boxplot

Note

none

See Also

xPredictROCR, xPredictCompare

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
gp <- xMLfeatureplot(df_predictor, GSP, GSN, displayBy="ROC")

## End(Not run)
```

---

xMLglmnet *Function to integrate predictor matrix in a supervised manner via machine learning algorithm glmnet.*

Description

xMLglmnet is supposed to integrate predictor matrix in a supervised manner via machine learning algorithm glmnet. It requires three inputs: 1) Gold Standard Positive (GSP) targets; 2) Gold Standard Negative (GSN) targets; 3) a predictor matrix containing genes in rows and predictors in columns, with their predictive scores inside it. It returns an object of class 'pTarget'.

Usage

```
xMLglmnet(
df_predictor,
GSP,
GSN,
family = c("binomial", "gaussian"),
type.measure = c("auc", "mse"),
nfold = 3,
alphas = seq(0, 1, 0.1),
standardize = TRUE,
lower.limits = -Inf,
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL,
...
)
```

Arguments

df_predictor	a data frame containing genes (in rows) and predictors (in columns), with their predictive scores inside it. This data frame must has gene symbols as row names
GSP	a vector containing Gold Standard Positive (GSP)
GSN	a vector containing Gold Standard Negative (GSN)
family	response family type. It can be one of "binomial" for two-class logistic model or "gaussian" for gaussian model
type.measure	loss to use for cross-validation. It can be one of "auc" for two-class logistic model, "mse" for the deviation from the fitted mean to the response using gaussian model
nfold	an integer specifying the number of folds for cross validataion
alphas	a vector specifying a range of alphas. Alpha is an elasticnet mixing parameter, with 0<=alpha<=1. By default, seq(0,1,by-0.1)
standardize	logical specifying whether to standardise the predictor. If yes (by default), the predictor standardised prior to fitting the model. The coefficients are always returned on the original scale
lower.limits	vector of lower limits for each coefficient (by default, '-Inf'; all should be non-positive). A single value provided will apply to every coefficient
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to TRUE for display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details
guid	a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details
...	additional parameters. Please refer to 'glmnet::cv.glmnet' for the complete list.

Value

an object of class "pTarget", a list with following components:

- `model`: an object of class "cv.glmnet" as a best model
- `priority`: a data frame of nGene X 5 containing gene priority information, where nGene is the number of genes in the input data frame, and the 5 columns are "GS" (either 'GSP', or 'GSN', or 'NEW'), "name" (gene names), "rank" (ranks of the priority scores), "priority" (priority score; rescaled into the 5-star ratings), and "description" (gene description)
- `predictor`: a data frame, which is the same as the input data frame but inserting an additional column 'GS' in the first column
- `cvm2alpha`: a data frame of nAlpha X 2 containing mean cross-validated error, where nAlpha is the number of alpha and the two columns are "min" (lambda.min) and "1se" (lambda.1se)
- `nonzero2alpha`: a data frame of nAlpha X 2 containing the number of non-zero coefficients, where nAlpha is the number of alpha and the two columns are "min" (lambda.min) and "1se" (lambda.1se)
- `importance`: a data frame of nPredictor X 1 containing the predictor importance/coefficient info
- `performance`: a data frame of 1+nPredictor X 2 containing the supervised/predictor performance info predictor importance info, where nPredictor is the number of predictors, two columns are "ROC" (AUC values) and "Fmax" (F-max values)
- gp: a ggplot object for the ROC curve
- `call`: the call that produced this result

Note

none

See Also

xPredictROCR, xPredictCompare, xSymbol2GeneID

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
pTarget <- xMLglmnet(df_prediction, GSP, GSN)

## End(Not run)
```

`xMLparameters` *Function to visualise cross-validation performance against tuning parameters*

Description

`xMLparameters` is supposed to visualise cross-validation performance against tuning parameters.

Usage

```
xMLparameters(
data,
nD = c("auto", "1D", "2D", "3D"),
contour = TRUE,
main = "Repeated cross-validation",
xlab = NA,
ylab = NA,
zlab = NA,
clab = "AUC (repeated CV)",
nlevels = 50,
colormap = c("lightblue-lightyellow-darkorange-darkred", "bwr", "jet",
"gbr", "wyr",
"br", "yr", "rainbow", "wb"),
highlight = TRUE,
x.label.cex = 0.8,
x.label.srt = 30,
theta.3D = 40,
phi.3D = 25
)
```

Arguments

`data`	an object of the class "train" or "train.formula" (resulting from caret::train) used for 1D or 2D visualisation. Alternatively, it can be a data frame used for 2D or 3D visualisation
`nD`	an integer specifying the dimension of the visualisation. It can be one of '1D', '2D' and '3D' and 'auto' (if input data is a "train" object)
`contour`	logical to indicate whether coutour plot should be also included
`main`	a title for the plot
`xlab`	a title for the x axis
`ylab`	a title for the y axis
`zlab`	a title for the z axis
`clab`	a title for the colorbar
`nlevels`	the number of levels to partition the input matrix values. The same level has the same color mapped to

colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
highlight	logical whether to highlight the point with the maximum value
x.label.cex	the x-axis label size
x.label.srt	the x-axis label angle (in degree from horizontal)
theta.3D	the azimuthal direction. By default, it is 40
phi.3D	the colatitude direction. By default, it is 20

Value

invisible

Note

none

See Also

xSparseMatrix

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
xMLparameters(df_fit, nD="2D")
xMLparameters(df_fit, nD="3D", theta.3D=40, phi.3D=60)

## End(Not run)
```

---

xMLrandomforest *Function to integrate predictor matrix in a supervised manner via machine learning algorithm random forest.*

Description

xMLrandomforest is supposed to integrate predictor matrix in a supervised manner via machine learning algorithm random forest. It requires three inputs: 1) Gold Standard Positive (GSP) targets; 2) Gold Standard Negative (GSN) targets; 3) a predictor matrix containing genes in rows and predictors in columns, with their predictive scores inside it. It returns an object of class 'sTarget'.

Usage

```
xMLrandomforest(
list_pNode = NULL,
df_predictor = NULL,
GSP,
GSN,
nfold = 3,
nrepeat = 10,
seed = 825,
mtry = NULL,
ntree = 1000,
fold.aggregateBy = c("logistic", "Ztransform", "fishers",
"orderStatistic"),
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL,
...
)
```

Arguments

list_pNode	a list of "pNode" objects or a "pNode" object
df_predictor	a data frame containing genes (in rows) and predictors (in columns), with their predictive scores inside it. This data frame must has gene symbols as row names
GSP	a vector containing Gold Standard Positive (GSP)
GSN	a vector containing Gold Standard Negative (GSN)
nfold	an integer specifying the number of folds for cross validataion. Per fold creates balanced splits of the data preserving the overall distribution for each class (GSP and GSN), therefore generating balanced cross-vallidation train sets and testing sets. By default, it is 3 meaning 3-fold cross validation
nrepeat	an integer specifying the number of repeats for cross validataion. By default, it is 10 indicating the cross-validation repeated 10 times
seed	an integer specifying the seed
mtry	an integer specifying the number of predictors randomly sampled as candidates at each split. If NULL, it will be tuned by 'randomForest::tuneRF', with starting value as sqrt(p) where p is the number of predictors. The minimum value is 3
ntree	an integer specifying the number of trees to grow. By default, it sets to 2000
fold.aggregateBy	the aggregate method used to aggregate results from k-fold cross validataion. It can be either "orderStatistic" for the method based on the order statistics of p-values, or "fishers" for Fisher's method, "Ztransform" for Z-transform method, "logistic" for the logistic method. Without loss of generality, the Z-transform method does well in problems where evidence against the combined null is spread widely (equal footings) or when the total evidence is weak; Fisher's method does best in problems where the evidence is concentrated in a relatively small fraction of the individual tests or when the evidence is at least moderately

	strong; the logistic method provides a compromise between these two. Notably, the aggregate methods 'Ztransform' and 'logistic' are preferred here
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to TRUE for display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details
guid	a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details
...	additional parameters. Please refer to 'randomForest::randomForest' for the complete list.

Value

an object of class "sTarget", a list with following components:

- model: a list of models, results from per-fold train set
- priority: a data frame of nGene X 5 containing gene priority information, where nGene is the number of genes in the input data frame, and the 5 columns are "GS" (either 'GSP', or 'GSN', or 'NEW'), "name" (gene names), "rank" (priority rank), "rating" (the 5-star priority score/rating), and "description" (gene description)
- predictor: a data frame, which is the same as the input data frame but inserting an additional column 'GS' in the first column
- pred2fold: a list of data frame, results from per-fold test set
- prob2fold: a data frame of nGene X 2+nfold containing the probability of being GSP, where nGene is the number of genes in the input data frame, nfold is the number of folds for cross validataion, and the first two columns are "GS" (either 'GSP', or 'GSN', or 'NEW'), "name" (gene names), and the rest columns storing the per-fold probability of being GSP
- importance2fold: a data frame of nPredictor X 4+nfold containing the predictor importance info per fold, where nPredictor is the number of predictors, nfold is the number of folds for cross validataion, and the first 4 columns are "median" (the median of the importance across folds), "mad" (the median of absolute deviation of the importance across folds), "min" (the minimum of the importance across folds), "max" (the maximum of the importance across folds), and the rest columns storing the per-fold importance
- roc2fold: a data frame of 1+nPredictor X 4+nfold containing the supervised/predictor ROC info (AUC values), where nPredictor is the number of predictors, nfold is the number of folds for cross validataion, and the first 4 columns are "median" (the median of the AUC values across folds), "mad" (the median of absolute deviation of the AUC values across folds), "min" (the minimum of the AUC values across folds), "max" (the maximum of the AUC values across folds), and the rest columns storing the per-fold AUC values
- fmax2fold: a data frame of 1+nPredictor X 4+nfold containing the supervised/predictor PR info (F-max values), where nPredictor is the number of predictors, nfold is the number of folds for cross validataion, and the first 4 columns are "median" (the median of the F-max values across folds), "mad" (the median of absolute deviation of the F-max values across folds), "min" (the minimum of the F-max values across folds), "max" (the maximum of the F-max values across folds), and the rest columns storing the per-fold F-max values

- importance: a data frame of nPredictor X 2 containing the predictor importance info, where nPredictor is the number of predictors, two columns for two types ("MeanDecreaseAccuracy" and "MeanDecreaseGini") of predictor importance measures. "MeanDecreaseAccuracy" sees how worse the model performs without each predictor (a high decrease in accuracy would be expected for very informative predictors), while "MeanDecreaseGini" measures how pure the nodes are at the end of the tree (a high score means the predictor was important if each predictor is taken out)
- performance: a data frame of 1+nPredictor X 2 containing the supervised/predictor performance info predictor performance info, where nPredictor is the number of predictors, two columns are "ROC" (AUC values) and "Fmax" (F-max values)
- evidence: an object of the class "eTarget", a list with following components "evidence" and "metag"
- list_pNode: a list of "pNode" objects

Note

none

See Also

xPierMatrix, xSparseMatrix, xPredictROCR, xPredictCompare, xSymbol2GeneID

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
sTarget <- xMLrandomforest(df_prediction, GSP, GSN)

## End(Not run)
```

---

xMLrename *Function to rename predictors used in machine learning*

Description

xMLrename is supposed to rename predictors used in machine learning. It returns an object of class "sTarget".

Usage

```
xMLrename(sTarget, old_names, new_names)
```

Arguments

sTarget	an object of class "sTarget"
old_names	a vector for the original names of predictors to be renamed
new_names	a vector for the new names

Value

an object of class "sTarget"

Note

none

See Also

xMLrename

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
old_names <- colnames(sTarget$predictor)[-c(1,2)]
old_names_1 <- c('nGene_20000_constant', 'eGene_CD14', 'eGene_LPS2',
'eGene_LPS24', 'eGene_IFN', 'eGene_Bcell', 'eGene_CD4', 'eGene_CD8',
'eGene_Neutrophil', 'eGene_NK', 'eGene_Blood', 'cGene_Monocytes',
'cGene_Macrophages_M0', 'cGene_Macrophages_M1', 'cGene_Macrophages_M2',
'cGene_Neutrophils', 'cGene_Naive_CD4_T_cells',
'cGene_Total_CD4_T_cells', 'cGene_Naive_CD8_T_cells',
'cGene_Total_CD8_T_cells', 'cGene_Naive_B_cells',
'cGene_Total_B_cells', 'dGene', 'pGene', 'fGene')
old_names_2 <- c('nGene_20000_constant', 'eGene_Pi_eQTL_CD14',
'eGene_Pi_eQTL_LPS2', 'eGene_Pi_eQTL_LPS24', 'eGene_Pi_eQTL_IFN',
'eGene_Pi_eQTL_Bcell', 'eGene_Pi_eQTL_CD4', 'eGene_Pi_eQTL_CD8',
'eGene_Pi_eQTL_Neutrophil', 'eGene_Pi_eQTL_NK', 'eGene_Pi_eQTL_Blood',
'cGene_Monocytes', 'cGene_Macrophages_M0', 'cGene_Macrophages_M1',
'cGene_Macrophages_M2', 'cGene_Neutrophils', 'cGene_Naive_CD4_T_cells',
'cGene_Total_CD4_T_cells', 'cGene_Naive_CD8_T_cells',
'cGene_Total_CD8_T_cells', 'cGene_Naive_B_cells',
'cGene_Total_B_cells', 'dGene', 'pGene', 'fGene')
new_names <- c('nearbyGenes_nGene: nearby genes', 'eQTL_eGene: resting
state (CD14+)', 'eQTL_eGene: activating state (CD14+ by LPS2h)',
'eQTL_eGene: activating state (CD14+ by LPS24h)', 'eQTL_eGene:
activating state (CD14+ by IFN24h)', 'eQTL_eGene: B cells',
'eQTL_eGene: CD4+ T cells', 'eQTL_eGene: CD8+ T cells', 'eQTL_eGene:
neutrophils', 'eQTL_eGene: NK cells','eQTL_eGene: peripheral blood',
'HiC_cGene: monocytes', 'HiC_cGene: macrophages (M0)', 'HiC_cGene:
macrophages (M1)', 'HiC_cGene: macrophages (M2)', 'HiC_cGene:
neutrophils', 'HiC_cGene: CD4+ T cells (naive)', 'HiC_cGene: CD4+ T
cells (total)', 'HiC_cGene: CD8+ T cells (naive)', 'HiC_cGene: CD8+ T
cells (total)', 'HiC_cGene: B cells (naive)', 'HiC_cGene: B cells
(total)', 'Annotation_dGene: disease genes', 'Annotation_pGene:
phenotype genes', 'Annotation_fGene: function genes')
sTarget_renamed <- xMLrename(sTarget, old_names_1, new_names)
sTarget_renamed <- xMLrename(sTarget_renamed, old_names_2, new_names)

## End(Not run)
```

xMLzoom *Function to visualise machine learning results using zoom plot*

Description

xMLzoom is supposed to visualise machine learning results using zoom plot. It returns an object of class "ggplot".

Usage

```
xMLzoom(
xTarget,
top = 20,
top.label.type = c("box", "text"),
top.label.size = 3,
top.label.query = NULL,
point.shape = 3,
font.family = "sans",
signature = TRUE
)
```

Arguments

xTarget	an object of class "sTarget" or "dTarget" (with the component 'pPerf')
top	the number of the top targets to be labelled/highlighted
top.label.type	how to label the top targets. It can be "box" drawing a box around the labels , and "text" for the text only
top.label.size	the highlight label size
top.label.query	which top genes in query will be labelled. By default, it sets to NULL meaning all top genes will be displayed. If labels in query can not be found, then none will be displayed
point.shape	an integer specifying point shapes. By default, it is 3 for cross. For details, please refere to http://sape.inf.usi.ch/quick-reference/ggplot2/shape
font.family	the font family for texts
signature	logical to indicate whether the signature is assigned to the plot caption. By default, it sets TRUE

Value

an object of class "ggplot"

Note

none

See Also

xColormap

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
gp <- xMLzoom(sTarget)
gp

## End(Not run)
```

---

xPieplot *Function to visualise data frame using pie plots*

Description

xPieplot is supposed to visualise data frame using pie plots. It returns an object of class "ggplot".

Usage

```
xPieplot(
df,
columns,
colormap = "ggplot2",
pie.radius = NULL,
pie.color = "transparent",
pie.color.alpha = 1,
pie.thick = 0.1,
legend.title = "",
gp = NULL
)
```

Arguments

df	a data frame
columns	a vector containing column names of the input data frame. These columns are used to draw pie charts
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta), and "ggplot2" (emulating ggplot2 default color palette). Alternatively, any hyphen-separated HTML color names, e.g. "lightyellow-orange" (by default), "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names

pie.radius	the radius of a pie. If NULL, it equals roughly 1/75
pie.color	the border color of a pie
pie.color.alpha	the 0-1 value specifying transparency of pie border colors
pie.thick	the pie border thickness
legend.title	the legend title
gp	an existing ggplot object or NULL. It is used for overlapping

Value

a ggplot object.

See Also

xColormap

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
gp <- xPieplot(df,
columns=c('dGene','pGene','fGene','nGene','eGene','cGene'),
legend.title='Seeds')

## End(Not run)
```

---

xPier *Function to do prioritisation through random walk techniques*

Description

xPier is supposed to prioritise nodes given an input graph and a list of seed nodes. It implements Random Walk with Restart (RWR) and calculates the affinity score of all nodes in the graph to the seeds. The priority score is the affinity score. Parallel computing is also supported for Linux-like or Windows operating systems. It returns an object of class "pNode".

Usage

```
xPier(
seeds,
g,
seeds.inclusive = TRUE,
normalise = c("laplacian", "row", "column", "none"),
restart = 0.7,
normalise.affinity.matrix = c("none", "quantile"),
parallel = TRUE,
multicores = NULL,
verbose = TRUE
)
```

Arguments

seeds	a named input vector containing a list of seed nodes. For this named vector, the element names are seed/node names (e.g. gene symbols), the element (non-zero) values used to weight the relative importance of seeds. Alternatively, it can be a matrix or data frame with two columns: 1st column for seed/node names, 2nd column for the weight values
g	an object of class "igraph" to represent network. It can be a weighted graph with the node attribute 'weight'
seeds.inclusive	logical to indicate whether non-network seed genes are included for prioritisation. If TRUE (by default), these genes will be added to the netowrk
normalise	the way to normalise the adjacency matrix of the input graph. It can be 'laplacian' for laplacian normalisation, 'row' for row-wise normalisation, 'column' for column-wise normalisation, or 'none'
restart	the restart probability used for Random Walk with Restart (RWR). The restart probability takes the value from 0 to 1, controlling the range from the starting nodes/seeds that the walker will explore. The higher the value, the more likely the walker is to visit the nodes centered on the starting nodes. At the extreme when the restart probability is zero, the walker moves freely to the neighbors at each step without restarting from seeds, i.e., following a random walk (RW)
normalise.affinity.matrix	the way to normalise the output affinity matrix. It can be 'none' for no normalisation, 'quantile' for quantile normalisation to ensure that columns (if multiple) of the output affinity matrix have the same quantiles
parallel	logical to indicate whether parallel computation with multicores is used. By default, it sets to true, but not necessarily does so. Partly because parallel backends available will be system-specific (now only Linux or Mac OS). Also, it will depend on whether these two packages "foreach" and "doMC" have been installed
multicores	an integer to specify how many cores will be registered as the multicore parallel backend to the 'foreach' package. If NULL, it will use a half of cores available in a user's computer. This option only works when parallel computation is enabled
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

an object of class "pNode", a list with following components:

- priority: a matrix of nNode X 5 containing node priority information, where nNode is the number of nodes in the input graph, and the 5 columns are "name" (node names), "node" (1 for network genes, 0 for non-network seed genes), "seed" (1 for seeds, 0 for non-seeds), "weight" (weight values), "priority" (the priority scores that are rescaled to the range [0,1]), "rank" (ranks of the priority scores)
- g: an input "igraph" object

Note

The input graph will treat as an unweighted graph if there is no 'weight' edge attribute associated with

See Also

xRWR

Examples

```
# a) provide the input nodes/genes with the significance info
sig <- rbeta(500, shape1=0.5, shape2=1)
## Not run:
## load human genes
org.Hs.eg <- xRDataLoader(RData='org.Hs.eg',
RData.location=RData.location)
data <- data.frame(symbols=org.Hs.eg$gene_info$Symbol[1:500], sig)

# b) provide the network
g <- xRDataLoader(RData.customised='org.Hs.PCommons_UN',
RData.location=RData.location)

# c) perform priority analysis
pNode <- xPier(seeds=data, g=g, restart=0.75)

## End(Not run)
```

---

xPierABF *Function to prioritise genes based on seed eGenes identified through ABF integrating GWAS and eQTL summary data*

Description

xPierABF is supposed to prioritise genes based on seed eGenes identified through ABF integrating GWAS and eQTL summary data. To prioritise genes, it first conducts colocalisation analysis through Wakefield's Approximate Bayes Factor (ABF) integrating GWAS and eQTL summary data to identify and score seed genes (that is, eGenes weighted by posterior probability of the SNP being causal for both GWAS and eQTL traits). It implements Random Walk with Restart (RWR) and calculates the affinity score of all nodes in the graph to the seeds. The priority score is the affinity score. Parallel computing is also supported for Linux-like or Windows operating systems. It returns an object of class "pNode".

Usage

```
xPierABF(
data,
eqtl = c("CD14", "LPS2", "LPS24", "IFN", "Bcell", "NK", "Neutrophil",
"CD4", "CD8",
```

```
"Blood", "Monocyte", "shared_CD14", "shared_LPS2", "shared_LPS24",
"shared_IFN"),
prior.eqtl = 1e-04,
prior.gwas = 1e-04,
prior.both = 1e-05,
cutoff.H4 = 0.8,
cutoff.pgwas = 1e-05,
network = c("STRING_highest", "STRING_high", "STRING_medium",
"STRING_low",
"PCommonsUN_high", "PCommonsUN_medium", "PCommonsDN_high",
"PCommonsDN_medium",
"PCommonsDN_Reactome", "PCommonsDN_KEGG", "PCommonsDN_HumanCyc",
"PCommonsDN_PID",
"PCommonsDN_PANTHER", "PCommonsDN_ReconX", "PCommonsDN_TRANSFAC",
"PCommonsDN_PhosphoSite", "PCommonsDN_CTD", "KEGG", "KEGG_metabolism",
"KEGG_genetic", "KEGG_environmental", "KEGG_cellular",
"KEGG_organismal",
"KEGG_disease"),
STRING.only = c(NA, "neighborhood_score", "fusion_score",
"cooccurence_score",
"coexpression_score", "experimental_score", "database_score",
"textmining_score")[1],
weighted = FALSE,
network.customised = NULL,
seeds.inclusive = TRUE,
normalise = c("laplacian", "row", "column", "none"),
restart = 0.7,
normalise.affinity.matrix = c("none", "quantile"),
parallel = TRUE,
multicores = NULL,
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

data	a data frame storing GWAS summary data with following required columns 'snp', 'effect' (the effect allele assessed), 'other' (other allele), 'b' (effect size for the allele assessed; log(odds ratio) for a case-control study), 'se' (standard error), 'p' (p-value)
eqtl	context-specific eQTL summary data. It can be one of "Bcell","Blood","CD14","CD4","CD8","IFN","LPS
prior.eqtl	the prior probability an eQTL associated with the eQTL trait. The default value is 1e-4
prior.gwas	the prior probability an SNP associated with the GWAS trait. The default value is 1e-4
prior.both	the prior probability an eQTL/SNP associated with both eQTL/GWAS traits. The default value is 1e-5

`cutoff.H4` the H4 cutoff used to define eGenes. This cutoff is based on the posterior probabilities of H4 - one shared causal variant. The default value is 0.8

`cutoff.pgwas` the GWAS p-value cutoff that must be met to consider SNPs. The default value is 1e-5

`network` the built-in network. Currently two sources of network information are supported: the STRING database (version 10) and the Pathways Commons database (version 7). STRING is a meta-integration of undirect interactions from the functional aspect, while Pathways Commons mainly contains both undirect and direct interactions from the physical/pathway aspect. Both have scores to control the confidence of interactions. Therefore, the user can choose the different quality of the interactions. In STRING, "STRING_highest" indicates interactions with highest confidence (confidence scores>=900), "STRING_high" for interactions with high confidence (confidence scores>=700), "STRING_medium" for interactions with medium confidence (confidence scores>=400), and "STRING_low" for interactions with low confidence (confidence scores>=150). For undirect/physical interactions from Pathways Commons, "PCommonsUN_high" indicates undirect interactions with high confidence (supported with the PubMed references plus at least 2 different sources), "PCommonsUN_medium" for undirect interactions with medium confidence (supported with the PubMed references). For direct (pathway-merged) interactions from Pathways Commons, "PCommonsDN_high" indicates direct interactions with high confidence (supported with the PubMed references plus at least 2 different sources), and "PCommonsUN_medium" for direct interactions with medium confidence (supported with the PubMed references). In addtion to pooled version of pathways from all data sources, the user can also choose the pathway-merged network from individual sources, that is, "PCommonsDN_Reactome" for those from Reactome, "PCommonsDN_KEGG" for those from KEGG, "PCommonsDN_HumanCyc" for those from HumanCyc, "PCommonsDN_PID" for those froom PID, "PCommonsDN_PANTHER" for those from PANTHER, "PCommonsDN_ReconX" for those from ReconX, "PCommonsDN_TRANSFAC" for those from TRANSFAC, "PCommonsDN_PhosphoSite" for those from PhosphoSite, and "PCommonsDN_CTD" for those from CTD. For direct (pathway-merged) interactions sourced from KEGG, it can be 'KEGG' for all, 'KEGG_metabolism' for pathways grouped into 'Metabolism', 'KEGG_genetic' for 'Genetic Information Processing' pathways, 'KEGG_environmental' for 'Environmental Information Processing' pathways, 'KEGG_cellular' for 'Cellular Processes' pathways, 'KEGG_organismal' for 'Organismal Systems' pathways, and 'KEGG_disease' for 'Human Diseases' pathways

`STRING.only` the further restriction of STRING by interaction type. If NA, no such restriction. Otherwide, it can be one or more of "neighborhood_score","fusion_score","cooccurence_score","coexpres Useful options are c("experimental_score","database_score"): only experimental data (extracted from BIND, DIP, GRID, HPRD, IntAct, MINT, and PID) and curated data (extracted from Biocarta, BioCyc, GO, KEGG, and Reactome) are used

`weighted` logical to indicate whether edge weights should be considered. By default, it sets to false. If true, it only works for the network from the STRING database

`network.customised` an object of class "igraph". By default, it is NULL. It is designed to allow the

user analysing their customised network data that are not listed in the above argument 'network'. This customisation (if provided) has the high priority over built-in network. If the user provides the "igraph" object with the "weight" edge attribute, RWR will assume to walk on the weighted network

`seeds.inclusive`
: logical to indicate whether non-network seed genes are included for prioritisation. If TRUE (by default), these genes will be added to the netowrk

`normalise`
: the way to normalise the adjacency matrix of the input graph. It can be 'laplacian' for laplacian normalisation, 'row' for row-wise normalisation, 'column' for column-wise normalisation, or 'none'

`restart`
: the restart probability used for Random Walk with Restart (RWR). The restart probability takes the value from 0 to 1, controlling the range from the starting nodes/seeds that the walker will explore. The higher the value, the more likely the walker is to visit the nodes centered on the starting nodes. At the extreme when the restart probability is zero, the walker moves freely to the neighbors at each step without restarting from seeds, i.e., following a random walk (RW)

`normalise.affinity.matrix`
: the way to normalise the output affinity matrix. It can be 'none' for no normalisation, 'quantile' for quantile normalisation to ensure that columns (if multiple) of the output affinity matrix have the same quantiles

`parallel`
: logical to indicate whether parallel computation with multicores is used. By default, it sets to true, but not necessarily does so. Partly because parallel backends available will be system-specific (now only Linux or Mac OS). Also, it will depend on whether these two packages "foreach" and "doMC" have been installed

`multicores`
: an integer to specify how many cores will be registered as the multicore parallel backend to the 'foreach' package. If NULL, it will use a half of cores available in a user's computer. This option only works when parallel computation is enabled

`verbose`
: logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

`RData.location`
: the characters to tell the location of built-in RData files. See `xRDataLoader` for details

`guid`
: a valid (5-character) Global Unique IDentifier for an OSF project. See `xRDataLoader` for details

Value

an object of class "pNode", a list with following components:

- `priority`: a matrix of nNode X 6 containing node priority information, where nNode is the number of nodes in the input graph, and the 5 columns are "name" (node names), "node" (1 for network genes, 0 for non-network seed genes), "seed" (1 for seeds, 0 for non-seeds), "weight" (weight values), "priority" (the priority scores that are rescaled to the range [0,1]), "rank" (ranks of the priority scores), "description" (node description)
- g: an input "igraph" object
- `evidence`: a data frame storing evidence
- `call`: the call that produced this result

Note

The input graph will treat as an unweighted graph if there is no 'weight' edge attribute associated with

See Also

xRDataLoader, xMEabf, xPierGenes

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
data <- utils::read.delim(file="summary_gwas.RA.txt", header=TRUE,
row.names=NULL, stringsAsFactors=FALSE)
pNode_abf <- xPierABF(data, eqtl="Blood", network="STRING_high",
restart=0.7, RData.location=RData.location)
write.table(pNode_abf$priority, file="Genes_priority.ABF.txt",
sep="\t", row.names=FALSE)

## End(Not run)
```

---

xPierABFheatmap *Function to visualise ABF evidence using heatmap*

Description

xPierABFheatmap is supposed to visualise ABF evidence using heatmap. It returns an object of class "ggplot".

Usage

```
xPierABFheatmap(
data,
xTarget,
type = c("Gene", "Gene_SNP"),
colormap = "steelblue-lightyellow-orange",
zlim = c(-0.5, 0.5)
)
```

Arguments

data	an input vector containing gene symbols
xTarget	a "dTarget" or "sTarget" object with the componet 'list_pNode' related to 'eGene' predictors. Alternatively, it can be a data frame with columns ('context','mode','probeID','Symbol','gene
type	the type of the heatmap. It can be "Gene" (gene-centric heatmap) or "Gene_SNP" (heatmap for the gene-snp pair)

colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
zlim	the minimum and maximum z values for which colors should be plotted

Value

an object of class "ggplot" appended with 'mat' (the matrix colored by 'b_ABF') and 'df' (a data frame with columns 'priority','code','context','mode','ProbeID','Symbol','gene_cse','snps','snp_cse','A1','A2','b_GWAS',

Note

none

See Also

xSparseMatrix, xHeatmap, xSparseMatrix

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
gp <- xPierABFheatmap(data, dTarget)

## End(Not run)
```

---

xPierAnno *Function to prioritise seed genes only from a list of pNode objects using annotation data*

Description

xPierAnno is supposed to prioritise seed genes only from a list of pNode objects using annotation data. To prioritise genes, it first extracts seed genes from a list of pNode objects and then scores seed genes using annotation data (or something similar). It implements Random Walk with Restart (RWR) and calculates the affinity score of all nodes in the graph to the seeds. The priority score is the affinity score. Parallel computing is also supported for Linux-like or Windows operating systems. It returns an object of class "pNode".

Usage

```
xPierAnno(
data,
list_pNode,
network = c("STRING_highest", "STRING_high", "STRING_medium",
"STRING_low",
"PCommonsUN_high", "PCommonsUN_medium", "PCommonsDN_high",
"PCommonsDN_medium",
"PCommonsDN_Reactome", "PCommonsDN_KEGG", "PCommonsDN_HumanCyc",
"PCommonsDN_PID",
"PCommonsDN_PANTHER", "PCommonsDN_ReconX", "PCommonsDN_TRANSFAC",
"PCommonsDN_PhosphoSite", "PCommonsDN_CTD", "KEGG", "KEGG_metabolism",
"KEGG_genetic", "KEGG_environmental", "KEGG_cellular",
"KEGG_organismal",
"KEGG_disease"),
STRING.only = c(NA, "neighborhood_score", "fusion_score",
"cooccurence_score",
"coexpression_score", "experimental_score", "database_score",
"textmining_score")[1],
weighted = FALSE,
network.customised = NULL,
seeds.inclusive = TRUE,
normalise = c("laplacian", "row", "column", "none"),
restart = 0.7,
normalise.affinity.matrix = c("none", "quantile"),
parallel = TRUE,
multicores = NULL,
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

data	a data frame with two columns: 1st column for seed/node names, 2nd column for the weight values. It intends to store annotation data or something similar
list_pNode	a list of "pNode" objects or a "pNode" object. Alternatively, it is NULL, meaning no restrictioin
network	the built-in network. Currently two sources of network information are supported: the STRING database (version 10) and the Pathways Commons database (version 7). STRING is a meta-integration of undirect interactions from the functional aspect, while Pathways Commons mainly contains both undirect and direct interactions from the physical/pathway aspect. Both have scores to control the confidence of interactions. Therefore, the user can choose the different quality of the interactions. In STRING, "STRING_highest" indicates interactions with highest confidence (confidence scores>=900), "STRING_high" for interactions with high confidence (confidence scores>=700), "STRING_medium" for interactions with medium confidence (confidence scores>=400), and "STRING_low"

for interactions with low confidence (confidence scores>=150). For undirect/physical interactions from Pathways Commons, "PCommonsUN_high" indicates undirect interactions with high confidence (supported with the PubMed references plus at least 2 different sources), "PCommonsUN_medium" for undirect interactions with medium confidence (supported with the PubMed references). For direct (pathway-merged) interactions from Pathways Commons, "PCommonsDN_high" indicates direct interactions with high confidence (supported with the PubMed references plus at least 2 different sources), and "PCommonsUN_medium" for direct interactions with medium confidence (supported with the PubMed references). In addtion to pooled version of pathways from all data sources, the user can also choose the pathway-merged network from individual sources, that is, "PCommonsDN_Reactome" for those from Reactome, "PCommonsDN_KEGG" for those from KEGG, "PCommonsDN_HumanCyc" for those from HumanCyc, "PCommonsDN_PID" for those froom PID, "PCommonsDN_PANTHER" for those from PANTHER, "PCommonsDN_ReconX" for those from ReconX, "PCommonsDN_TRANSFAC" for those from TRANSFAC, "PCommonsDN_PhosphoSite" for those from PhosphoSite, and "PCommonsDN_CTD" for those from CTD. For direct (pathway-merged) interactions sourced from KEGG, it can be 'KEGG' for all, 'KEGG_metabolism' for pathways grouped into 'Metabolism', 'KEGG_genetic' for 'Genetic Information Processing' pathways, 'KEGG_environmental' for 'Environmental Information Processing' pathways, 'KEGG_cellular' for 'Cellular Processes' pathways, 'KEGG_organismal' for 'Organismal Systems' pathways, and 'KEGG_disease' for 'Human Diseases' pathways

`STRING.only` the further restriction of STRING by interaction type. If NA, no such restriction. Otherwide, it can be one or more of "neighborhood_score","fusion_score","cooccurence_score","coexpres Useful options are c("experimental_score","database_score"): only experimental data (extracted from BIND, DIP, GRID, HPRD, IntAct, MINT, and PID) and curated data (extracted from Biocarta, BioCyc, GO, KEGG, and Reactome) are used

`weighted` logical to indicate whether edge weights should be considered. By default, it sets to false. If true, it only works for the network from the STRING database

`network.customised`
an object of class "igraph". By default, it is NULL. It is designed to allow the user analysing their customised network data that are not listed in the above argument 'network'. This customisation (if provided) has the high priority over built-in network. If the user provides the "igraph" object with the "weight" edge attribute, RWR will assume to walk on the weighted network

`seeds.inclusive`
logical to indicate whether non-network seed genes are included for prioritisation. If TRUE (by default), these genes will be added to the netowrk

`normalise` the way to normalise the adjacency matrix of the input graph. It can be 'laplacian' for laplacian normalisation, 'row' for row-wise normalisation, 'column' for column-wise normalisation, or 'none'

`restart` the restart probability used for Random Walk with Restart (RWR). The restart probability takes the value from 0 to 1, controlling the range from the starting nodes/seeds that the walker will explore. The higher the value, the more likely

the walker is to visit the nodes centered on the starting nodes. At the extreme when the restart probability is zero, the walker moves freely to the neighbors at each step without restarting from seeds, i.e., following a random walk (RW)

`normalise.affinity.matrix`
: the way to normalise the output affinity matrix. It can be 'none' for no normalisation, 'quantile' for quantile normalisation to ensure that columns (if multiple) of the output affinity matrix have the same quantiles

`parallel`
: logical to indicate whether parallel computation with multicores is used. By default, it sets to true, but not necessarily does so. Partly because parallel backends available will be system-specific (now only Linux or Mac OS). Also, it will depend on whether these two packages "foreach" and "doMC" have been installed

`multicores`
: an integer to specify how many cores will be registered as the multicore parallel backend to the 'foreach' package. If NULL, it will use a half of cores available in a user's computer. This option only works when parallel computation is enabled

`verbose`
: logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

`RData.location`
: the characters to tell the location of built-in RData files. See `xRDataLoader` for details

`guid`
: a valid (5-character) Global Unique IDentifier for an OSF project. See `xRDataLoader` for details

Value

an object of class "pNode", a list with following components:

- `priority`: a matrix of nNode X 6 containing node priority information, where nNode is the number of nodes in the input graph, and the 5 columns are "name" (node names), "node" (1 for network genes, 0 for non-network seed genes), "seed" (1 for seeds, 0 for non-seeds), "weight" (weight values), "priority" (the priority scores that are rescaled to the range [0,1]), "rank" (ranks of the priority scores), "description" (node description)
- g: an input "igraph" object
- `call`: the call that produced this result

Note

The input graph will treat as an unweighted graph if there is no 'weight' edge attribute associated with

See Also

`xPierMatrix`, `xPierGenes`

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# a) provide the seed nodes/genes with the weight info
```

```
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
## get genes within 500kb away from AS GWAS lead SNPs
seeds.genes <- ImmunoBase$AS$genes_variants
## seeds weighted according to distance away from lead SNPs
data <- 1- seeds.genes/500000

# b) perform priority analysis
pNode <- xPierGenes(data=data, network="PCommonsDN_medium",restart=0.7,
RData.location=RData.location)

# c) get annotation data
data.file <- file.path(RData.location, "iAnno.txt")
iA <- read.delim(data.file, header=TRUE, stringsAsFactors=FALSE)[,
1:14]
data_anno <- subset(iA, OMIM>0, select=c("Symbol","OMIM"))

# d) perform priority analysis using annotation data
pNode_anno <- xPierAnno(data_anno, list_pNode=pNode,
network="PCommonsDN_medium", restart=0.7,
RData.location=RData.location)

# c) save to the file called 'Genes_priority.Anno.txt'
write.table(pNode_anno$priority, file="Genes_priority.Anno.txt",
sep="\t", row.names=FALSE)

## End(Not run)
```

---

xPierCor *Function to calculate correlation between prioritised genes and user-defined external data*

Description

xPierCor is supposed to calculate correlation between prioritised genes and user-defined external data.

Usage

```
xPierCor(
pNode,
list_vec,
method = c("pearson", "spearman"),
pvalue.type = c("nominal", "empirical"),
seed = 825,
nperm = 2000,
p.adjust.method = c("BH", "BY", "bonferroni", "holm", "hochberg",
"hommel"),
```

```
plot = FALSE
)
```

Arguments

pNode	an object of class "pNode" (or "sTarget" or "dTarget"). Alternatively, it can be a data frame with two columns ('name' and 'priority')
list_vec	a named vector containing numeric values for genes (gene symbols). Alternatively it can be a list of named vectors
method	the method used to calcualte correlation. It can be 'pearson' for Pearson's correlation or 'spearman' for Spearman rank correlation
pvalue.type	the type of the p-value calcualted. It can be 'nominal' for nominal p-value or 'empirical' for empirical p-value
seed	an integer specifying the seed
nperm	the number of random permutations
p.adjust.method	the method used to adjust p-values. It can be one of "BH", "BY", "bonferroni", "holm", "hochberg" and "hommel". The first two methods "BH" (widely used) and "BY" control the false discovery rate (FDR: the expected proportion of false discoveries amongst the rejected hypotheses); the last four methods "bonferroni", "holm", "hochberg" and "hommel" are designed to give strong control of the family-wise error rate (FWER). Notes: FDR is a less stringent condition than FWER
plot	logical to indicate whether scatter plot is drawn

Value

a list with two componets:

- df_summary: a data frame of n x 4, where n is the number of named vectors, and the 4 columns are "name", "cor" (i.e. "correlation"), "pval" (i.e. p-value), "fdr"
- ls_gp: NULL if the plot is not drawn; otherwise, a list of 'ggplot' objects

Note

none

See Also

xCorrelation

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# a) provide the seed nodes/genes with the weight info
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
```

```
RData.location=RData.location)
## get genes within 500kb away from AS GWAS lead SNPs
seeds.genes <- ImmunoBase$AS$genes_variants
## seeds weighted according to distance away from lead SNPs
data <- 1- seeds.genes/500000

# b) perform priority analysis
pNode <- xPierGenes(data=data, network="PCommonsDN_medium",restart=0.7,
RData.location=RData.location)

# c) do correlation
data <- pNode$priority$priority[1:100]
name(data) <- pNode$priority$name[1:100]
ls_res <- xPierCor(pNode, data, method="pearson",
pvalue.type="empirical", nperm=2000, plot=TRUE)

## End(Not run)
```

`xPierCross` *Function to extract priority matrix from a list of dTarget/sTarget objects*

Description

`xPierCross` is supposed to extract priority matrix from a list of dTarget objects. Also supported is the aggregation of priority matrix (similar to the meta-analysis) generating the priority results; we view this functionality as the cross mode of the prioritisation.

Usage

```
xPierCross(
list_xTarget,
displayBy = c("rating", "rank", "pvalue", "fdr"),
combineBy = c("intersect", "union"),
aggregateBy = c("none", "fishers", "logistic", "Ztransform",
"orderStatistic"),
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

`list_xTarget`	a list of "dTarget"/"sTarget" objects or a "dTarget"/"sTarget" object
`displayBy`	which priority will be extracted. It can be "rating" for priority score/rating (by default), "rank" for priority rank, "pvalue" for priority p-value, "fdr" for priority fdr

combineBy	how to resolve nodes/targets from a list of "dTarget"/"sTarget" objects. It can be "intersect" for intersecting nodes (by default), "union" for unionising nodes
aggregateBy	the aggregate method used. It can be either "none" for no aggregation, or "orderStatistic" for the method based on the order statistics of p-values, "fishers" for Fisher's method, "Ztransform" for Z-transform method, "logistic" for the logistic method. Without loss of generality, the Z-transform method does well in problems where evidence against the combined null is spread widely (equal footings) or when the total evidence is weak; Fisher's method does best in problems where the evidence is concentrated in a relatively small fraction of the individual tests or when the evidence is at least moderately strong; the logistic method provides a compromise between these two. Notably, the aggregate methods 'fishers' and 'logistic' are preferred here
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details
guid	a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details

Value

If aggregateBy is 'none' (by default), a data frame containing priority matrix, with each column/disease for either priority score/rating, or priorty rank or priority p-value. If aggregateBy is not 'none', an object of the class "cTarget", a list with following components:

- priority: a data frame of nGene X 6 containing gene priority (aggregated) information, where nGene is the number of genes, and the 6 columns are "name" (gene names), "rank" (ranks of the priority scores), "pvalue" (the aggregated p-value, converted from empirical cumulative distribution of the probability of being GSP), "fdr" (fdr adjusted from the aggregated p-value), "priority" (-log10(pvalue) but rescaled into the 5-star ratings), "description" (gene description)
- disease: a data frame containing disease matrix, with each column/disease for either priority score, or priorty rank or priority p-value

Note

none

See Also

xSymbol2GeneID

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
df_score <- xPierCross(ls_xTarget)

## End(Not run)
```

`xPierEvidence` *Function to extract evidence from a list of pNode objects*

Description

`xPierEvidence` is supposed to extract evidence from a list of pNode objects, in terms of seed genes under genetic influence.

Usage

```
xPierEvidence(list_pNode, target.query = NULL, verbose = TRUE)
```

Arguments

`list_pNode`	a list of "pNode" objects or a "pNode" object
`target.query`	which gene is in query. If NULL, all genes will be queried
`verbose`	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

a data frame of nPair X 5 containing Gene-SNP pair info per context, where the 6 columns are "Gene" (seed genes), "SNP" (dbSNP), "Score" (an SNP's genetic influential score on a seed gene), "Context" (predictors), "Flag" (indicative of Lead SNPs or LD SNPs), and "Pval" (the SNP p-value)

Note

none

See Also

`xPierEvidence`

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
df_Gene2SNP <- xPierEvidence(ls_pNode)

## End(Not run)
```

xPierGenes *Function to prioritise genes from an input network and the weight info imposed on its nodes*

Description

xPierGenes is supposed to prioritise genes given an input graph and a list of seed nodes. It implements Random Walk with Restart (RWR) and calculates the affinity score of all nodes in the graph to the seeds. The priority score is the affinity score. Parallel computing is also supported for Linux-like or Windows operating systems. It returns an object of class "pNode".

Usage

```
xPierGenes(
data,
network = c("STRING_highest", "STRING_high", "STRING_medium",
"STRING_low",
"PCommonsUN_high", "PCommonsUN_medium", "PCommonsDN_high",
"PCommonsDN_medium",
"PCommonsDN_Reactome", "PCommonsDN_KEGG", "PCommonsDN_HumanCyc",
"PCommonsDN_PID",
"PCommonsDN_PANTHER", "PCommonsDN_ReconX", "PCommonsDN_TRANSFAC",
"PCommonsDN_PhosphoSite", "PCommonsDN_CTD", "KEGG", "KEGG_metabolism",
"KEGG_genetic", "KEGG_environmental", "KEGG_cellular",
"KEGG_organismal",
"KEGG_disease", "REACTOME"),
STRING.only = c(NA, "neighborhood_score", "fusion_score",
"cooccurence_score",
"coexpression_score", "experimental_score", "database_score",
"textmining_score")[1],
weighted = FALSE,
network.customised = NULL,
seeds.inclusive = TRUE,
normalise = c("laplacian", "row", "column", "none"),
restart = 0.7,
normalise.affinity.matrix = c("none", "quantile"),
parallel = TRUE,
multicores = NULL,
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

data	a named input vector containing a list of seed nodes (ie gene symbols). For this named vector, the element names are seed/node names (e.g. gene symbols),

the element (non-zero) values used to weight the relative importance of seeds. Alternatively, it can be a matrix or data frame with two columns: 1st column for seed/node names, 2nd column for the weight values

`network` the built-in network. Currently two sources of network information are supported: the STRING database (version 10) and the Pathway Commons database (version 7). STRING is a meta-integration of undirect interactions from the functional aspect, while Pathways Commons mainly contains both undirect and direct interactions from the physical/pathway aspect. Both have scores to control the confidence of interactions. Therefore, the user can choose the different quality of the interactions. In STRING, "STRING_highest" indicates interactions with highest confidence (confidence scores>=900), "STRING_high" for interactions with high confidence (confidence scores>=700), "STRING_medium" for interactions with medium confidence (confidence scores>=400), and "STRING_low" for interactions with low confidence (confidence scores>=150). For undirect/physical interactions from Pathways Commons, "PCommonsUN_high" indicates undirect interactions with high confidence (supported with the PubMed references plus at least 2 different sources), "PCommonsUN_medium" for undirect interactions with medium confidence (supported with the PubMed references). For direct (pathway-merged) interactions from Pathways Commons, "PCommonsDN_high" indicates direct interactions with high confidence (supported with the PubMed references plus at least 2 different sources), and "PCommonsUN_medium" for direct interactions with medium confidence (supported with the PubMed references). In addition to pooled version of pathways from all data sources, the user can also choose the pathway-merged network from individual sources, that is, "PCommonsDN_Reactome" for those from Reactome, "PCommonsDN_KEGG" for those from KEGG, "PCommonsDN_HumanCyc" for those from HumanCyc, "PCommonsDN_PID" for those froom PID, "PCommonsDN_PANTHER" for those from PANTHER, "PCommonsDN_ReconX" for those from ReconX, "PCommonsDN_TRANSFAC" for those from TRANSFAC, "PCommonsDN_PhosphoSite" for those from PhosphoSite, and "PCommonsDN_CTD" for those from CTD. For direct (pathway-merged) interactions sourced from KEGG, it can be 'KEGG' for all, 'KEGG_metabolism' for pathways grouped into 'Metabolism', 'KEGG_genetic' for 'Genetic Information Processing' pathways, 'KEGG_environmental' for 'Environmental Information Processing' pathways, 'KEGG_cellular' for 'Cellular Processes' pathways, 'KEGG_organismal' for 'Organismal Systems' pathways, and 'KEGG_disease' for 'Human Diseases' pathways. 'REACTOME' for protein-protein interactions derived from Reactome pathways

`STRING.only` the further restriction of STRING by interaction type. If NA, no such restriction. Otherwide, it can be one or more of "neighborhood_score","fusion_score","cooccurence_score","coexpres
Useful options are c("experimental_score","database_score"): only experimental data (extracted from BIND, DIP, GRID, HPRD, IntAct, MINT, and PID) and curated data (extracted from Biocarta, BioCyc, GO, KEGG, and Reactome) are used

`weighted` logical to indicate whether edge weights should be considered. By default, it sets to false. If true, it only works for the network from the STRING database

`network.customised`
an object of class "igraph". By default, it is NULL. It is designed to allow the

user analysing their customised network data that are not listed in the above argument 'network'. This customisation (if provided) has the high priority over built-in network. If the user provides the "igraph" object with the "weight" edge attribute, RWR will assume to walk on the weighted network

`seeds.inclusive` logical to indicate whether non-network seed genes are included for prioritisation. If TRUE (by default), these genes will be added to the netowrk

`normalise` the way to normalise the adjacency matrix of the input graph. It can be 'laplacian' for laplacian normalisation, 'row' for row-wise normalisation, 'column' for column-wise normalisation, or 'none'

`restart` the restart probability used for Random Walk with Restart (RWR). The restart probability takes the value from 0 to 1, controlling the range from the starting nodes/seeds that the walker will explore. The higher the value, the more likely the walker is to visit the nodes centered on the starting nodes. At the extreme when the restart probability is zero, the walker moves freely to the neighbors at each step without restarting from seeds, i.e., following a random walk (RW)

`normalise.affinity.matrix` the way to normalise the output affinity matrix. It can be 'none' for no normalisation, 'quantile' for quantile normalisation to ensure that columns (if multiple) of the output affinity matrix have the same quantiles

`parallel` logical to indicate whether parallel computation with multicores is used. By default, it sets to true, but not necessarily does so. Partly because parallel backends available will be system-specific (now only Linux or Mac OS). Also, it will depend on whether these two packages "foreach" and "doMC" have been installed

`multicores` an integer to specify how many cores will be registered as the multicore parallel backend to the 'foreach' package. If NULL, it will use a half of cores available in a user's computer. This option only works when parallel computation is enabled

`verbose` logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

`RData.location` the characters to tell the location of built-in RData files. See `xRDataLoader` for details

`guid` a valid (5-character) Global Unique IDentifier for an OSF project. See `xRDataLoader` for details

Value

an object of class "pNode", a list with following components:

- `priority`: a matrix of nNode X 6 containing node priority information, where nNode is the number of nodes in the input graph, and the 5 columns are "name" (node names), "node" (1 for network genes, 0 for non-network seed genes), "seed" (1 for seeds, 0 for non-seeds), "weight" (weight values), "priority" (the priority scores that are rescaled to the range [0,1]), "rank" (ranks of the priority scores), "description" (node description)
- g: an input "igraph" object

Note

The input graph will treat as an unweighted graph if there is no 'weight' edge attribute associated with

See Also

xDefineNet, xPier

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# a) provide the seed nodes/genes with the weight info
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
## get genes within 500kb away from AS GWAS lead SNPs
seeds.genes <- ImmunoBase$AS$genes_variants
## seeds weighted according to distance away from lead SNPs
data <- 1- seeds.genes/500000

# b) perform priority analysis
pNode <- xPierGenes(data=data, network="PCommonsDN_medium",restart=0.7,
RData.location=RData.location)

# c) save to the file called 'Genes_priority.txt'
write.table(pNode$priority, file="Genes_priority.txt", sep="\t",
row.names=FALSE)

## End(Not run)
```

---

xPierGRs *Function to prioritise genes given a list of genomic regions*

Description

xPierGRs is supposed to prioritise genes given a list of genomic regions with or without the significance level. To prioritise genes, it first defines and scores genes crosslinking to an input list of genomic regions (GR). With seed genes and their scores, it then uses Random Walk with Restart (RWR) to calculate the affinity score of all nodes in the input graph to the seed genes. The priority score is the affinity score. Parallel computing is also supported for Linux-like or Windows operating systems. It returns an object of class "pNode".

Usage

```
xPierGRs(
data,
significance.threshold = NULL,
```

```
score.cap = NULL,
build.conversion = c(NA, "hg38.to.hg19", "hg18.to.hg19"),
crosslink = c("genehancer", "PCHiC_combined", "GTEx_V6p_combined",
"nearby"),
crosslink.customised = NULL,
cdf.function = c("original", "empirical"),
scoring.scheme = c("max", "sum", "sequential"),
nearby.distance.max = 50000,
nearby.decay.kernel = c("rapid", "slow", "linear", "constant"),
nearby.decay.exponent = 2,
network = c("STRING_highest", "STRING_high", "STRING_medium",
"STRING_low",
"PCommonsUN_high", "PCommonsUN_medium", "PCommonsDN_high",
"PCommonsDN_medium",
"PCommonsDN_Reactome", "PCommonsDN_KEGG", "PCommonsDN_HumanCyc",
"PCommonsDN_PID",
"PCommonsDN_PANTHER", "PCommonsDN_ReconX", "PCommonsDN_TRANSFAC",
"PCommonsDN_PhosphoSite", "PCommonsDN_CTD", "KEGG", "KEGG_metabolism",
"KEGG_genetic", "KEGG_environmental", "KEGG_cellular",
"KEGG_organismal",
"KEGG_disease", "REACTOME"),
STRING.only = c(NA, "neighborhood_score", "fusion_score",
"cooccurence_score",
"coexpression_score", "experimental_score", "database_score",
"textmining_score")[1],
weighted = FALSE,
network.customised = NULL,
seeds.inclusive = TRUE,
normalise = c("laplacian", "row", "column", "none"),
restart = 0.7,
normalise.affinity.matrix = c("none", "quantile"),
parallel = TRUE,
multicores = NULL,
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

data	a named input vector containing the significance level for genomic regions (GR). For this named vector, the element names are GR, in the format of 'chrN:start-end', where N is either 1-22 or X, start (or end) is genomic positional number; for example, 'chr1:13-20', the element values for the significance level (measured as p-value or fdr). Alternatively, it can be a matrix or data frame with two columns: 1st column for GR, 2nd column for the significance level. Also supported is the input with GR only (without the significance level)
significance.threshold	the given significance threshold. By default, it is set to NULL, meaning there is

no constraint on the significance level when transforming the significance level of GR into scores. If given, those GR below this are considered significant and thus scored positively. Instead, those above this are considered insignificant and thus receive no score

`score.cap` the maximum score being capped. By default, it is set to NULL, meaning that no capping is applied

`build.conversion`
the conversion from one genome build to another. The conversions supported are "hg38.to.hg19" and "hg18.to.hg19". By default it is NA (no need to do so)

`crosslink` the built-in crosslink info with a score quantifying the link of a GR to a gene. See `xGR2xGenes` for details

`crosslink.customised`
the crosslink info with a score quantifying the link of a GR to a gene. A user-input matrix or data frame with 4 columns: 1st column for genomic regions (formatted as "chr:start-end", genome build 19), 2nd column for Genes, 3rd for crosslink score (crosslinking a genomic region to a gene, such as -log10 significance level), and 4th for contexts (optional; if nor provided, it will be added as 'C'). Alternatively, it can be a file containing these 4 columns. Required, otherwise it will return NULL

`cdf.function` a character specifying how to transform the input crosslink score. It can be one of 'original' (no such transformation), and 'empirical' for looking at empirical Cumulative Distribution Function (cdf; as such it is converted into pvalue-like values [0,1])

`scoring.scheme` the method used to calculate seed gene scores under a set of GR (also over Contexts if many). It can be one of "sum" for adding up, "max" for the maximum, and "sequential" for the sequential weighting. The sequential weighting is done via: $\sum_{i=1} \frac{R_i}{i}$, where R_i is the i^{th} rank (in a descreasing order)

`nearby.distance.max`
the maximum distance between genes and GR. Only those genes no far way from this distance will be considered as seed genes. This parameter will influence the distance-component weights calculated for nearby GR per gene

`nearby.decay.kernel`
a character specifying a decay kernel function. It can be one of 'slow' for slow decay, 'linear' for linear decay, and 'rapid' for rapid decay. If no distance weight is used, please select 'constant'

`nearby.decay.exponent`
a numeric specifying a decay exponent. By default, it sets to 2

`network` the built-in network. Currently two sources of network information are supported: the STRING database (version 10) and the Pathway Commons database (version 7). STRING is a meta-integration of undirect interactions from the functional aspect, while Pathways Commons mainly contains both undirect and direct interactions from the physical/pathway aspect. Both have scores to control the confidence of interactions. Therefore, the user can choose the different quality of the interactions. In STRING, "STRING_highest" indicates interactions with highest confidence (confidence scores>=900), "STRING_high" for interactions with high confidence (confidence scores>=700), "STRING_medium" for

interactions with medium confidence (confidence scores>=400), and "STRING_low" for interactions with low confidence (confidence scores>=150). For undirect/physical interactions from Pathways Commons, "PCommonsUN_high" indicates undirect interactions with high confidence (supported with the PubMed references plus at least 2 different sources), "PCommonsUN_medium" for undirect interactions with medium confidence (supported with the PubMed references). For direct (pathway-merged) interactions from Pathways Commons, "PCommonsDN_high" indicates direct interactions with high confidence (supported with the PubMed references plus at least 2 different sources), and "PCommonsUN_medium" for direct interactions with medium confidence (supported with the PubMed references). In addition to pooled version of pathways from all data sources, the user can also choose the pathway-merged network from individual sources, that is, "PCommonsDN_Reactome" for those from Reactome, "PCommonsDN_KEGG" for those from KEGG, "PCommonsDN_HumanCyc" for those from HumanCyc, "PCommonsDN_PID" for those froom PID, "PCommonsDN_PANTHER" for those from PANTHER, "PCommonsDN_ReconX" for those from ReconX, "PCommonsDN_TRANSFAC" for those from TRANSFAC, "PCommonsDN_PhosphoSite" for those from PhosphoSite, and "PCommonsDN_CTD" for those from CTD. For direct (pathway-merged) interactions sourced from KEGG, it can be 'KEGG' for all, 'KEGG_metabolism' for pathways grouped into 'Metabolism', 'KEGG_genetic' for 'Genetic Information Processing' pathways, 'KEGG_environmental' for 'Environmental Information Processing' pathways, 'KEGG_cellular' for 'Cellular Processes' pathways, 'KEGG_organismal' for 'Organismal Systems' pathways, and 'KEGG_disease' for 'Human Diseases' pathways. 'REACTOME' for protein-protein interactions derived from Reactome pathways

`STRING.only` the further restriction of STRING by interaction type. If NA, no such restriction. Otherwide, it can be one or more of "neighborhood_score","fusion_score","cooccurence_score","coexpres Useful options are c("experimental_score","database_score"): only experimental data (extracted from BIND, DIP, GRID, HPRD, IntAct, MINT, and PID) and curated data (extracted from Biocarta, BioCyc, GO, KEGG, and Reactome) are used

`weighted` logical to indicate whether edge weights should be considered. By default, it sets to false. If true, it only works for the network from the STRING database

`network.customised`
an object of class "igraph". By default, it is NULL. It is designed to allow the user analysing their customised network data that are not listed in the above argument 'network'. This customisation (if provided) has the high priority over built-in network. If the user provides the "igraph" object with the "weight" edge attribute, RWR will assume to walk on the weighted network

`seeds.inclusive`
logical to indicate whether non-network seed genes are included for prioritisation. If TRUE (by default), these genes will be added to the netowrk

`normalise` the way to normalise the adjacency matrix of the input graph. It can be 'laplacian' for laplacian normalisation, 'row' for row-wise normalisation, 'column' for column-wise normalisation, or 'none'

`restart` the restart probability used for Random Walk with Restart (RWR). The restart probability takes the value from 0 to 1, controlling the range from the starting

nodes/seeds that the walker will explore. The higher the value, the more likely the walker is to visit the nodes centered on the starting nodes. At the extreme when the restart probability is zero, the walker moves freely to the neighbors at each step without restarting from seeds, i.e., following a random walk (RW)

`normalise.affinity.matrix`
: the way to normalise the output affinity matrix. It can be 'none' for no normalisation, 'quantile' for quantile normalisation to ensure that columns (if multiple) of the output affinity matrix have the same quantiles

`parallel`
: logical to indicate whether parallel computation with multicores is used. By default, it sets to true, but not necessarily does so. Partly because parallel backends available will be system-specific (now only Linux or Mac OS). Also, it will depend on whether these two packages "foreach" and "doMC" have been installed

`multicores`
: an integer to specify how many cores will be registered as the multicore parallel backend to the 'foreach' package. If NULL, it will use a half of cores available in a user's computer. This option only works when parallel computation is enabled

`verbose`
: logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

`RData.location`
: the characters to tell the location of built-in RData files. See `xRDataLoader` for details

`guid`
: a valid (5-character) Global Unique IDentifier for an OSF project. See `xRDataLoader` for details

Value

an object of class "pNode", a list with following components:

- `priority`: a matrix of nNode X 6 containing node priority information, where nNode is the number of nodes in the input graph, and the 6 columns are "name" (node names), "node" (1 for network genes, 0 for non-network seed genes), "seed" (1 for seeds, 0 for non-seeds), "weight" (weight values), "priority" (the priority scores that are rescaled to the range [0,1]), "rank" (ranks of the priority scores), "description" (node description)
- g: an input "igraph" object
- `mSeed`: a list with following components 'GR', 'Gene' and 'Link'

See Also

`xGR2xGeneScores`, `xPierGenes`

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

## Not run:
# a) provide the seed SNPs with the significance info
data(ImmunoBase)
## only AS GWAS SNPs and their significance info (p-values)
df <- as.data.frame(ImmunoBase$AS$variant, row.names=NULL)
```

```
GR <- paste0(df$seqnames,':',df$start,'-',df$end)
data <- cbind(GR=GR, Sig=df$Pvalue)

# b) perform priority analysis
pNode <- xPierGRs(data=data, crosslink="PCHiC_combined",
network="STRING_highest", restart=0.7, RData.location=RData.location)

# c) save to the file called 'GRs_priority.txt'
write.table(pNode$priority, file="GRs_priority.txt", sep="\t",
row.names=FALSE)

# d) manhattan plot
mp <- xPierManhattan(pNode, top=20, top.label.size=1.5, y.scale="sqrt",
RData.location=RData.location)
#pdf(file="Gene_manhattan.pdf", height=6, width=12, compress=TRUE)
print(mp)
#dev.off()

## End(Not run)
```

---

xPierGSEA *Function to prioritise pathways based on GSEA analysis of prioritised genes*

Description

xPierGSEA is supposed to prioritise pathways given prioritised genes and the ontology in query. It is done via gene set enrichment analysis (GSEA). It returns an object of class "eGSEA".

Usage

```
xPierGSEA(
pNode,
priority.top = NULL,
ontology = c("GOBP", "GOMF", "GOCC", "PS", "PS2", "SF", "Pfam", "DO",
"HPPA", "HPMI",
"HPCM", "HPMA", "MP", "EF", "MsigdbH", "MsigdbC1", "MsigdbC2CGP",
"MsigdbC2CPall",
"MsigdbC2CP", "MsigdbC2KEGG", "MsigdbC2REACTOME", "MsigdbC2BIOCARTA",
"MsigdbC3TFT",
"MsigdbC3MIR", "MsigdbC4CGN", "MsigdbC4CM", "MsigdbC5BP", "MsigdbC5MF",
"MsigdbC5CC",
"MsigdbC6", "MsigdbC7", "DGIdb", "GTExV4", "GTExV6p", "GTExV7",
"CreedsDisease",
"CreedsDiseaseUP", "CreedsDiseaseDN", "CreedsDrug", "CreedsDrugUP",
"CreedsDrugDN",
"CreedsGene", "CreedsGeneUP", "CreedsGeneDN", "KEGG", "KEGGmetabolism",
"KEGGgenetic", "KEGGenvironmental", "KEGGcellular", "KEGGorganismal",
```

```
"KEGGdisease"),
customised.genesets = NULL,
size.range = c(10, 500),
p.adjust.method = c("BH", "BY", "bonferroni", "holm", "hochberg",
"hommel"),
path.mode = c("all_paths", "shortest_paths", "all_shortest_paths"),
weight = 1,
seed = 825,
nperm = 2000,
fast = TRUE,
verbose = TRUE,
silent = FALSE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

pNode	an object of class "pNode" (or "sTarget" or "dTarget"). Alternatively, it can be a data frame with two columns ('priority' and 'rank')
priority.top	the number of the top targets used for GSEA. By default, it is NULL meaning all targets are used
ontology	the ontology supported currently. It can be "GOBP" for Gene Ontology Biological Process, "GOMF" for Gene Ontology Molecular Function, "GOCC" for Gene Ontology Cellular Component, "PS" for phylostratific age information, "PS2" for the collapsed PS version (inferred ancestors being collapsed into one with the known taxonomy information), "SF" for SCOP domain superfamilies, "Pfam" for Pfam domain families, "DO" for Disease Ontology, "HPPA" for Human Phenotype Phenotypic Abnormality, "HPMI" for Human Phenotype Mode of Inheritance, "HPCM" for Human Phenotype Clinical Modifier, "HPMA" for Human Phenotype Mortality Aging, "MP" for Mammalian Phenotype, "EF" for Experimental Factor Ontology (used to annotate GWAS Catalog genes), Drug-Gene Interaction database ("DGIdb") for drugable categories, tissue-specific eQTL-containing genes from GTEx ("GTExV4", "GTExV6p" and "GTExV7"), crowd extracted expression of differential signatures from CREEDS ("CreedsDisease", "CreedsDiseaseUP", "CreedsDiseaseDN", "CreedsDrug", "CreedsDrugUP", "CreedsDrugDN", "CreedsGene", "CreedsGeneUP" and "CreedsGeneDN"), KEGG pathways (including 'KEGG' for all, 'KEGGmetabolism' for 'Metabolism' pathways, 'KEGGgenetic' for 'Genetic Information Processing' pathways, 'KEGGenvironmental' for 'Environmental Information Processing' pathways, 'KEGGcellular' for 'Cellular Processes' pathways, 'KEGGorganismal' for 'Organismal Systems' pathways, and 'KEGGdisease' for 'Human Diseases' pathways), and the molecular signatures database (Msigdb, including "MsigdbH", "MsigdbC1", "MsigdbC2CGP", "MsigdbC2CPall", "MsigdbC2CP", "MsigdbC2KEGG", "MsigdbC2REACTOME", "MsigdbC2BIOCARTA", "MsigdbC3TFT", "MsigdbC3MIR", "MsigdbC4CGN", "MsigdbC4CM", "MsigdbC5BP", "MsigdbC5MF", "MsigdbC5CC", "MsigdbC6", "MsigdbC7")

customised.genesets
: a list each containing gene symbols. By default, it is NULL. If the list provided, it will overtake the previous parameter "ontology"

size.range
: the minimum and maximum size of members of each term in consideration. By default, it sets to a minimum of 10 but no more than 500

p.adjust.method
: the method used to adjust p-values. It can be one of "BH", "BY", "bonferroni", "holm", "hochberg" and "hommel". The first two methods "BH" (widely used) and "BY" control the false discovery rate (FDR: the expected proportion of false discoveries amongst the rejected hypotheses); the last four methods "bonferroni", "holm", "hochberg" and "hommel" are designed to give strong control of the family-wise error rate (FWER). Notes: FDR is a less stringent condition than FWER

path.mode
: the mode of paths induced by vertices/nodes with input annotation data. It can be "all_paths" for all possible paths to the root, "shortest_paths" for only one path to the root (for each node in query), "all_shortest_paths" for all shortest paths to the root (i.e. for each node, find all shortest paths with the equal lengths)

weight
: an integer specifying score weight. It can be "0" for unweighted (an equivalent to Kolmogorov-Smirnov, only considering the rank), "1" for weighted by input gene score (by default), and "2" for over-weighted, and so on

seed
: an integer specifying the seed

nperm
: the number of random permutations. For each permutation, gene-score associations will be permutated so that permutation of gene-term associations is realised

fast
: logical to indicate whether to fast calculate GSEA resulting. By default, it sets to true, but not necessarily does so. It will depend on whether the package "fgsea" has been installed

verbose
: logical to indicate whether the messages will be displayed in the screen. By default, it sets to true

silent
: logical to indicate whether the messages will be silent completely. By default, it sets to false. If true, verbose will be forced to be false

RData.location
: the characters to tell the location of built-in RData files. See xRDataLoader for details

guid
: a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details

Value

an object of class "eGSEA", a list with following components:

- df_summary: a data frame of nTerm x 9 containing gene set enrichment analysis result, where nTerm is the number of terms/genesets, and the 9 columns are "setID" (i.e. "Term ID"), "name" (i.e. "Term Name"), "nAnno" (i.e. number in members annotated by a term), "nLead" (i.e. number in members as leading genes), "peak" (i.e. the rank at peak), "total" (i.e. the total number of genes analysed), "es" (i.e. enrichment score), "nes" (i.e. normalised enrichment score; enrichment score but after being normalised by gene set size), "pvalue" (i.e. nominal p value), "adjp" (i.e. adjusted p value; p value but after being adjusted for multiple comparisons), "distance" (i.e. term distance or metadata)

- `leading`: a list of gene sets, each storing leading gene info (i.e. the named vector with names for gene symbols and elements for priority rank). Always, gene sets are identified by "setID"
- `full`: a list of gene sets, each storing full info on gene set enrichment analysis result (i.e. a data frame of nGene x 6, where nGene is the number of genes, and the 6 columns are "GeneID", "Rank" for priority rank, "Score" for priority score, "RES" for running enrichment score, "Hits" for gene set hits info with 1 for gene hit, 2 for leading gene hit, 3 for the point defining leading genes, 0 for no hit), and "Symbol" for gene symbols. Always, gene sets are identified by "setID"
- `cross`: a matrix of nTerm X nTerm, with an on-diagnal cell for the leading genes observed in an individaul term, and off-diagnal cell for the overlapped leading genes shared between two terms

Note

none

See Also

xSymbol2GeneID, xRDataLoader, xDAGanno

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# a) provide the seed nodes/genes with the weight info
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
## get genes within 500kb away from AS GWAS lead SNPs
seeds.genes <- ImmunoBase$AS$genes_variants
## seeds weighted according to distance away from lead SNPs
data <- 1- seeds.genes/500000

# b) perform priority analysis
pNode <- xPierGenes(data=data, network="PCommonsDN_medium",restart=0.7,
RData.location=RData.location)

# c) do pathway-level priority using GSEA
eGSEA <- xPierGSEA(pNode=pNode, ontology="DGIdb", nperm=2000,
RData.location=RData.location)
bp <- xGSEAbarplot(eGSEA, top_num="auto", displayBy="nes")
gp <- xGSEAdotplot(eGSEA, top=1)

## End(Not run)
```

xPierManhattan *Function to visualise prioritised genes using manhattan plot*

Description

xPierManhattan is supposed to visualise prioritised genes using manhattan plot. Genes with the top priority are highlighed. It returns an object of class "ggplot".

Usage

```
xPierManhattan(
pNode,
color = c("darkred", "darkgreen"),
top = 50,
top.label.type = c("box", "text"),
top.label.size = 2,
top.label.col = "darkblue",
top.label.query = NULL,
label.query.only = FALSE,
chromosome.only = TRUE,
y.scale = c("normal", "sqrt", "log"),
y.lab = NULL,
GR.Gene = c("UCSC_knownGene", "UCSC_knownCanonical"),
font.family = "sans",
signature = TRUE,
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL,
...
)
```

Arguments

pNode	an object of class "pNode" (or "sTarget" or "dTarget")
color	a character vector for colors to alternate chromosome colorings. If NULL, ggplot2 default colors will be used. If a single character is provided, it can be "jet" (jet colormap) or "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta)
top	the number of the top targets to be labelled/highlighted
top.label.type	how to label the top targets. It can be "box" drawing a box around the labels , and "text" for the text only
top.label.size	the highlight label size
top.label.col	the highlight label color

`top.label.query`
: which top genes in query will be labelled. By default, it sets to NULL meaning all top genes will be displayed. If labels in query can not be found, then all will be displayed

`label.query.only`
: logical to indicate whether only genes in query will be displayed. By default, it sets to FALSE. It only works when labels in query are enabled/found

`chromosome.only`
: logical to indicate whether only genes from input data will be displayed. By default, it sets to TRUE

`y.scale`
: how to transform the y scale. It can be "normal" for no transformation, "sqrt" for square root transformation, and "log" for log-based transformation

`y.lab`
: the y labelling. If NULL (by default), it shows the column of input data

`GR.Gene`
: the genomic regions of genes. By default, it is 'UCSC_knownGene', that is, UCSC known genes (together with genomic locations) based on human genome assembly hg19. It can be 'UCSC_knownCanonical', that is, UCSC known canonical genes (together with genomic locations) based on human genome assembly hg19. Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to Gene Symbols. Then, tell "GR.Gene" with your RData file name (with or without extension), plus specify your file RData path in "RData.location"

`font.family`
: the font family for texts

`signature`
: logical to indicate whether the signature is assigned to the plot caption. By default, it sets TRUE

`verbose`
: logical to indicate whether the messages will be displayed in the screen. By default, it sets to false for no display

`RData.location`
: the characters to tell the location of built-in RData files. See `xRDataLoader` for details

`guid`
: a valid (5-character) Global Unique IDentifier for an OSF project. See `xRDataLoader` for details

`...`
: additional paramters associated with ggrepel::geom_text_repel

Value

an object of class "ggplot", appended by an GR object called 'gr'

Note

none

See Also

`xRDataLoader`, `xColormap`

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# a) provide the SNPs with the significance info
## get lead SNPs reported in AS GWAS and their significance info (p-values)
#data.file <- "http://galahad.well.ox.ac.uk/bigdata/AS.txt"
#AS <- read.delim(data.file, header=TRUE, stringsAsFactors=FALSE)
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
gr <- ImmunoBase$AS$variants
AS <- as.data.frame(GenomicRanges::mcols(gr)[, c('Variant','Pvalue')])

# b) perform priority analysis
pNode <- xPierSNPs(data=AS, include.eQTL="JKng_mono",
include.HiC='Monocytes', network="PCommonsUN_medium", restart=0.7,
RData.location=RData.location)

# c) manhattan plot
## default plot
mp <- xPierManhattan(pNode, RData.location=RData.location)
#pdf(file="Gene_manhattan.pdf", height=6, width=12, compress=TRUE)
print(mp)
#dev.off()
mp$gr
## control visuals
mp <- xPierManhattan(pNode, color='ggplot2', top=50,
top.label.col="black", y.scale="sqrt", RData.location=RData.location)
mp
## control labels
# only IL genes will be labelled
ind <- grep('^IL', rownames(pNode$priority))
top.label.query <- rownames(pNode$priority)[ind]
mp <- xPierManhattan(pNode, top.label.query=top.label.query,
RData.location=RData.location)
mp
# only IL genes will be displayed
mp <- xPierManhattan(pNode, top.label.query=top.label.query,
label.query.only=TRUE, RData.location=RData.location)
mp

## End(Not run)
```

`xPierMatrix` *Function to extract priority or evidence matrix from a list of pNode objects*

Description

`xPierMatrix` is supposed to extract priority or evidence matrix from a list of pNode objects. Also supported is the aggregation of priority matrix (similar to the meta-analysis) generating the priority results; we view this functionality as the discovery mode of the prioritisation.

Usage

```
xPierMatrix(
list_pNode,
displayBy = c("score", "rank", "weight", "pvalue", "evidence"),
combineBy = c("union", "intersect"),
aggregateBy = c("none", "fishers", "logistic", "Ztransform",
"orderStatistic"),
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

list_pNode	a list of "pNode" objects or a "pNode" object
displayBy	which priority will be extracted. It can be "score" for priority score/rating (by default), "rank" for priority rank, "weight" for seed weight, "pvalue" for priority p-value, "evidence" for the evidence (seed info)
combineBy	how to resolve nodes/targets from a list of "pNode" objects. It can be "intersect" for intersecting nodes (by default), "union" for unionising nodes
aggregateBy	the aggregate method used. It can be either "none" for no aggregation, or "orderStatistic" for the method based on the order statistics of p-values, "fishers" for Fisher's method, "Ztransform" for Z-transform method, "logistic" for the logistic method. Without loss of generality, the Z-transform method does well in problems where evidence against the combined null is spread widely (equal footings) or when the total evidence is weak; Fisher's method does best in problems where the evidence is concentrated in a relatively small fraction of the individual tests or when the evidence is at least moderately strong; the logistic method provides a compromise between these two. Notably, the aggregate methods 'fishers' and 'logistic' are preferred here
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details
guid	a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details

Value

If displayBy is 'evidence', an object of the class "eTarget", a list with following components:

- evidence: a data frame of nGene X 6 containing gene evidence information, where nGene is the number of genes, and the 7 columns are seed info including "Overall" for the total number of different types of seeds, followed by details on individual type of seeds (that is, "dGene", "pGene", "fGene", "nGene", "eGene", "cGene")
- metag: an "igraph" object

Otherwise (if displayBy is not 'evidence'), if aggregateBy is 'none' (by default), a data frame containing priority matrix, with each column/predictor for either priority score, or priorty rank or priority p-value. If aggregateBy is not 'none', an object of the class "dTarget", a list with following components:

- `priority`: a data frame of n X 4+7 containing gene priority (aggregated) information, where n is the number of genes, and the 4 columns are "name" (gene names), "rank" (ranks of the priority scores), "rating" (the 5-star score/rating), "description" (gene description), and 7 seed info columns including "seed" (whether or not seed genes), "nGene" (nearby genes), "cGene" (conformation genes), "eGene" (eQTL gens), "dGene" (disease genes), "pGene" (phenotype genes), and "fGene" (function genes)
- `predictor`: a data frame containing predictor matrix, with each column/predictor for either priority score/rating, or priorty rank or priority p-value
- `metag`: an "igraph" object
- `list_pNode`: a list of "pNode" objects

Note

none

See Also

`xSparseMatrix`, `xSymbol2GeneID`

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# get predictor matrix for targets
df_score <- xPierMatrix(ls_pNode)
# get evidence for targets
eTarget <- xPierMatrix(ls_pNode, displayBy="evidence")
# get target priority in a discovery mode
dTarget <- xPierMatrix(ls_pNode, displayBy="pvalue",
aggregateBy="fishers")

## End(Not run)
```

---

xPierMRS *Function to calculate multi-trait rating score from a list of dTarget/sTarget objects*

Description

`xPierMRS` is supposed to calculate multi-trait rating score (MRS) from a list of dTarget/sTarget objects.

Usage

```
xPierMRS(list_xTarget, cutoff.rank = 150, verbose = TRUE)
```

Arguments

list_xTarget	a list of "dTarget"/"sTarget" objects
cutoff.rank	the rank cutoff. By default it is 150
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

a data frame containing columns 'Target', 'MRS', 'rating' (in the form of "rating.trait_names") and 'rank' (in the form of "rank.trait_names").

Note

none

See Also

xPierCross

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
df_MRS <- xPierMRS(ls_xTarget)

## End(Not run)
```

---

xPierPathways *Function to prioritise pathways based on enrichment analysis of top prioritised genes*

Description

xPierPathways is supposed to prioritise pathways given prioritised genes and the ontology in query. It returns an object of class "eTerm". It is done via enrichment analysis.

Usage

```
xPierPathways(
pNode,
priority.top = 100,
background = NULL,
ontology = NA,
size.range = c(10, 2000),
min.overlap = 3,
which.distance = NULL,
test = c("hypergeo", "fisher", "binomial"),
background.annotatable.only = NULL,
p.tail = c("one-tail", "two-tails"),
p.adjust.method = c("BH", "BY", "bonferroni", "holm", "hochberg",
"hommel"),
ontology.algorithm = c("none", "pc", "elim", "lea"),
elim.pvalue = 0.01,
lea.depth = 2,
path.mode = c("all_paths", "shortest_paths", "all_shortest_paths"),
true.path.rule = FALSE,
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

pNode	an object of class "pNode" (or "sTarget" or "dTarget")
priority.top	the number of the top targets used for enrichment analysis. By default, it sets to 100
background	a background vector. It contains a list of Gene Symbols as the test background. If NULL, by default all annotatable are used as background
ontology	the ontology supported currently. By default, it is 'NA' to disable this option. Pre-built ontology and annotation data are detailed in xDefineOntology.
size.range	the minimum and maximum size of members of each term in consideration. By default, it sets to a minimum of 10 but no more than 2000
min.overlap	the minimum number of overlaps. Only those terms with members that overlap with input data at least min.overlap (3 by default) will be processed
which.distance	which terms with the distance away from the ontology root (if any) is used to restrict terms in consideration. By default, it sets to 'NULL' to consider all distances
test	the statistic test used. It can be "fisher" for using fisher's exact test, "hypergeo" for using hypergeometric test, or "binomial" for using binomial test. Fisher's exact test is to test the independence between gene group (genes belonging to a group or not) and gene annotation (genes annotated by a term or not), and thus compare sampling to the left part of background (after sampling without

replacement). Hypergeometric test is to sample at random (without replacement) from the background containing annotated and non-annotated genes, and thus compare sampling to background. Unlike hypergeometric test, binomial test is to sample at random (with replacement) from the background with the constant probability. In terms of the ease of finding the significance, they are in order: hypergeometric test > binomial test > fisher's exact test. In other words, in terms of the calculated p-value, hypergeometric test < binomial test < fisher's exact test

`background.annotatable.only`
: logical to indicate whether the background is further restricted to the annotatable. By default, it is NULL: if ontology.algorithm is not 'none', it is always TRUE; otherwise, it depends on the background (if not provided, it will be TRUE; otherwise FALSE). Surely, it can be explicitly stated

`p.tail`
: the tail used to calculate p-values. It can be either "two-tails" for the significance based on two-tails (ie both over- and under-overrepresentation) or "one-tail" (by default) for the significance based on one tail (ie only over-representation)

`p.adjust.method`
: the method used to adjust p-values. It can be one of "BH", "BY", "bonferroni", "holm", "hochberg" and "hommel". The first two methods "BH" (widely used) and "BY" control the false discovery rate (FDR: the expected proportion of false discoveries amongst the rejected hypotheses); the last four methods "bonferroni", "holm", "hochberg" and "hommel" are designed to give strong control of the family-wise error rate (FWER). Notes: FDR is a less stringent condition than FWER

`ontology.algorithm`
: the algorithm used to account for the hierarchy of the ontology. It can be one of "none", "pc", "elim" and "lea". For details, please see 'Note' below

`elim.pvalue`
: the parameter only used when "ontology.algorithm" is "elim". It is used to control how to declare a signficantly enriched term (and subsequently all genes in this term are eliminated from all its ancestors)

`lea.depth`
: the parameter only used when "ontology.algorithm" is "lea". It is used to control how many maximum depth is used to consider the children of a term (and subsequently all genes in these children term are eliminated from the use for the recalculation of the signifance at this term)

`path.mode`
: the mode of paths induced by vertices/nodes with input annotation data. It can be "all_paths" for all possible paths to the root, "shortest_paths" for only one path to the root (for each node in query), "all_shortest_paths" for all shortest paths to the root (i.e. for each node, find all shortest paths with the equal lengths)

`true.path.rule`
: logical to indicate whether the true-path rule should be applied to propagate annotations. By default, it sets to false

`verbose`
: logical to indicate whether the messages will be displayed in the screen. By default, it sets to false for no display

`RData.location`
: the characters to tell the location of built-in RData files. See `xRDataLoader` for details

`guid`
: a valid (5-character) Global Unique IDentifier for an OSF project. See `xRDataLoader` for details

Value

an object of class "eTerm", a list with following components:

- `term_info`: a matrix of nTerm X 4 containing snp/gene set information, where nTerm is the number of terms, and the 4 columns are "id" (i.e. "Term ID"), "name" (i.e. "Term Name"), "namespace" and "distance"
- `annotation`: a list of terms containing annotations, each term storing its annotations. Always, terms are identified by "id"
- `data`: a vector containing input data in consideration. It is not always the same as the input data as only those mappable are retained
- `background`: a vector containing the background data. It is not always the same as the input data as only those mappable are retained
- `overlap`: a list of overlapped snp/gene sets, each storing snps overlapped between a snp/gene set and the given input data (i.e. the snps of interest). Always, gene sets are identified by "id"
- `zscore`: a vector containing z-scores
- `pvalue`: a vector containing p-values
- `adjp`: a vector containing adjusted p-values. It is the p value but after being adjusted for multiple comparisons
- `call`: the call that produced this result

Note

The interpretation of the algorithms used to account for the hierarchy of the ontology is:

- "none": does not consider the ontology hierarchy at all.
- "lea": computers the significance of a term in terms of the significance of its children at the maximum depth (e.g. 2). Precisely, once snps are already annotated to any children terms with a more signficance than itself, then all these snps are eliminated from the use for the recalculation of the signifance at that term. The final p-values takes the maximum of the original p-value and the recalculated p-value.
- "elim": computers the significance of a term in terms of the significance of its all children. Precisely, once snps are already annotated to a signficantly enriched term under the cutoff of e.g. pvalue<1e-2, all these snps are eliminated from the ancestors of that term).
- "pc": requires the significance of a term not only using the whole snps as background but also using snps annotated to all its direct parents/ancestors as background. The final p-value takes the maximum of both p-values in these two calculations.
- "Notes": the order of the number of significant terms is: "none" > "lea" > "elim" > "pc".

See Also

xEnricherGenes

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# a) provide the SNPs with the significance info
## get lead SNPs reported in AS GWAS and their significance info (p-values)
#data.file <- "http://galahad.well.ox.ac.uk/bigdata/AS.txt"
#AS <- read.delim(data.file, header=TRUE, stringsAsFactors=FALSE)
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase')
gr <- ImmunoBase$AS$variants
AS <- as.data.frame(GenomicRanges::mcols(gr)[, c('Variant','Pvalue')])

# b) perform priority analysis
pNode <- xPierSNPs(data=AS, include.eQTL="JKng_mono",
include.HiC='Monocytes', network="PCommonsUN_medium", restart=0.7,
RData.location=RData.location)

# c) derive pathway-level priority
eTerm <- xPierPathways(pNode=pNode, priority.top=100,
ontology="MsigdbC2CP", RData.location=RData.location)

# d) view enrichment results for the top significant terms
xEnrichViewer(eTerm)

# e) save enrichment results to the file called 'Pathways_priority.txt'
res <- xEnrichViewer(eTerm, top_num=length(eTerm$adjp), sortBy="adjp",
details=TRUE)
output <- data.frame(term=rownames(res), res)
utils::write.table(output, file="Pathways_priority.txt", sep="\t",
row.names=FALSE)

## End(Not run)
```

`xPierROCR` *Function to assess the dTarget performance via ROC and Precision-Recall (PR) analysis*

Description

`xPierROCR` is supposed to assess the dTarget performance via Receiver Operating Characteristic (ROC) and Precision-Recall (PR) analysis. It requires three inputs: 1) Gold Standard Positive (GSP) targets; 2) Gold Standard Negative (GSN) targets; 3) dTarget containing predicted targets and predictive scores.

Usage

```
xPierROCR(dTarget, GSP, GSN, verbose = TRUE)
```

Arguments

dTarget	a data frame containing dTargets along with predictive scores. It has two columns: 1st column for target, 2nd column for predictive scores (the higher the better). Alternatively, it can be an object of class "pNode" (or "sTarget" or "dTarget") from which a data frame is extracted
GSP	a vector containing Gold Standard Positives (GSP)
GSN	a vector containing Gold Standard Negatives (GSN)
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to TRUE for display

Value

an object of the class "dTarget", a list with following components:

- priority: a data frame of nGene X 7 containing gene priority (aggregated) information, where nGene is the number of genes, and the 7 columns are "GS" (either 'GSP', or 'GSN', or 'NEW'), "name" (gene names), "rank" (ranks of the priority scores), "pvalue" (the aggregated p-value, converted from empirical cumulative distribution of the probability of being GSP), "fdr" (fdr adjusted from the aggregated p-value), "priority" (-log10(pvalue) but rescaled into the 5-star ratings), "description" (gene description) and seed info including "Overall" for the number of different types of seeds, followed by details on individual type of seeds (that is, "OMIM", "Phenotype", "Function", "nearbyGenes", "eQTL", "HiC")
- predictor: a data frame containing predictor matrix, with each column/predictor for either priority score, or priorty rank or priority p-value
- metag: an "igraph" object
- pPerf: a "pPerf" object, with components "PRS", "AUROC", "Fmax", "ROC_perf", "PR_perf", "Pred_obj"

Note

AUC: the area under ROC F-measure: the maximum of a harmonic mean between precision and recall along PR curve

See Also

xPierROCR

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
dTarget <- xPierROCR(dTarget, GSP, GSN)
gp <- xPredictCompare(dTarget$pPerf)

## End(Not run)
```

xPierSNPs *Function to prioritise genes given a list of seed SNPs together with the significance level (e.g. GWAS reported p-values)*

Description

xPierSNPs is supposed to prioritise genes given a list of seed SNPs together with the significance level. To prioritise genes, it first defines and scores seed genes: nearby genes, eQTL genes and Hi-C genes. With seed genes and their scores, it then uses Random Walk with Restart (RWR) to calculate the affinity score of all nodes in the input graph to the seed genes. The priority score is the affinity score. Parallel computing is also supported for Linux-like or Windows operating systems. It returns an object of class "pNode".

Usage

```
xPierSNPs(
data,
include.LD = NA,
LD.customised = NULL,
LD.r2 = 0.8,
significance.threshold = 5e-05,
score.cap = 10,
distance.max = 2000,
decay.kernel = c("slow", "constant", "linear", "rapid"),
decay.exponent = 2,
GR.SNP = c("dbSNP_GWAS", "dbSNP_Common", "dbSNP_Single"),
GR.Gene = c("UCSC_knownGene", "UCSC_knownCanonical"),
include.TAD = c("none", "GM12878", "IMR90", "MSC", "TRO", "H1", "MES",
"NPC"),
include.eQTL = NA,
eQTL.customised = NULL,
include.HiC = NA,
cdf.function = c("empirical", "exponential"),
relative.importance = c(1/3, 1/3, 1/3),
scoring.scheme = c("max", "sum", "sequential"),
network = c("STRING_highest", "STRING_high", "STRING_medium",
"STRING_low",
"PCommonsUN_high", "PCommonsUN_medium", "PCommonsDN_high",
"PCommonsDN_medium",
"PCommonsDN_Reactome", "PCommonsDN_KEGG", "PCommonsDN_HumanCyc",
"PCommonsDN_PID",
"PCommonsDN_PANTHER", "PCommonsDN_ReconX", "PCommonsDN_TRANSFAC",
"PCommonsDN_PhosphoSite", "PCommonsDN_CTD", "KEGG", "KEGG_metabolism",
"KEGG_genetic", "KEGG_environmental", "KEGG_cellular",
"KEGG_organismal",
"KEGG_disease", "REACTOME"),
STRING.only = c(NA, "neighborhood_score", "fusion_score",
```

```
"cooccurence_score",
"coexpression_score", "experimental_score", "database_score",
"textmining_score")[1],
weighted = FALSE,
network.customised = NULL,
seeds.inclusive = TRUE,
normalise = c("laplacian", "row", "column", "none"),
restart = 0.7,
normalise.affinity.matrix = c("none", "quantile"),
parallel = TRUE,
multicores = NULL,
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

data	a named input vector containing the sinificance level for nodes (dbSNP). For this named vector, the element names are dbSNP ID (or in the format such as 'chr16:28525386'), the element values for the significance level (measured as p-value or fdr). Alternatively, it can be a matrix or data frame with two columns: 1st column for dbSNP, 2nd column for the significance level
include.LD	additional SNPs in LD with Lead SNPs are also included. By default, it is 'NA' to disable this option. Otherwise, LD SNPs will be included based on one or more of 26 populations and 5 super populations from 1000 Genomics Project data (phase 3). The population can be one of 5 super populations ("AFR", "AMR", "EAS", "EUR", "SAS"), or one of 26 populations ("ACB", "ASW", "BEB", "CDX", "CEU", "CHB", "CHS", "CLM", "ESN", "FIN", "GBR", "GIH", "GWD", "IBS", "ITU", "JPT", "KHV", "LWK", "MSL", "MXL", "PEL", "PJL", "PUR", "STU", "TSI", "YRI"). Explanations for population code can be found at http://www.1000genomes.org/faq/which-populations-are-part-your-study
LD.customised	a user-input matrix or data frame with 3 columns: 1st column for Lead SNPs, 2nd column for LD SNPs, and 3rd for LD r2 value. It is designed to allow the user analysing their pre-calculated LD info. This customisation (if provided) has the high priority over built-in LD SNPs
LD.r2	the LD r2 value. By default, it is 0.8, meaning that SNPs in LD (r2>=0.8) with input SNPs will be considered as LD SNPs. It can be any value from 0.8 to 1
significance.threshold	the given significance threshold. By default, it is set to NULL, meaning there is no constraint on the significance level when transforming the significance level of SNPs into scores. If given, those SNPs below this are considered significant and thus scored positively. Instead, those above this are considered insigificant and thus receive no score
score.cap	the maximum score being capped. By default, it is set to 10. If NULL, no capping is applied

`distance.max` the maximum distance between genes and SNPs. Only those genes no far way from this distance will be considered as seed genes. This parameter will influence the distance-component weights calculated for nearby SNPs per gene

`decay.kernel` a character specifying a decay kernel function. It can be one of 'slow' for slow decay, 'linear' for linear decay, and 'rapid' for rapid decay. If no distance weight is used, please select 'constant'

`decay.exponent` an integer specifying a decay exponent. By default, it sets to 2

`GR.SNP` the genomic regions of SNPs. By default, it is 'dbSNP_GWAS', that is, SNPs from dbSNP (version 146) restricted to GWAS SNPs and their LD SNPs (hg19). It can be 'dbSNP_Common', that is, Common SNPs from dbSNP (version 146) plus GWAS SNPs and their LD SNPs (hg19). Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to dbSNP IDs. Then, tell "GR.SNP" with your RData file name (with or without extension), plus specify your file RData path in "RData.location". Note: you can also load your customised GR object directly

`GR.Gene` the genomic regions of genes. By default, it is 'UCSC_knownGene', that is, UCSC known genes (together with genomic locations) based on human genome assembly hg19. It can be 'UCSC_knownCanonical', that is, UCSC known canonical genes (together with genomic locations) based on human genome assembly hg19. Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to Gene Symbols. Then, tell "GR.Gene" with your RData file name (with or without extension), plus specify your file RData path in "RData.location". Note: you can also load your customised GR object directly

`include.TAD` TAD boundary regions are also included. By default, it is 'none' to disable this option. Otherwise, inclusion of a TAD dataset to pre-filter SNP-nGene pairs (i.e. only those within a TAD region will be kept). TAD datasets can be one of "GM12878" (lymphoblast), "IMR90" (fibroblast), "MSC" (mesenchymal stem cell) ,"TRO" (trophoblasts-like cell), "H1" (embryonic stem cell), "MES" (mesendoderm) and "NPC" (neural progenitor cell). Explanations can be found at http://dx.doi.org/10.1016/j.celrep.2016.10.061

`include.eQTL` the eQTL supported currently. By default, it is 'NA' to disable this option. Pre-built eQTL datasets are detailed in `xDefineEQTL`

`eQTL.customised`
a user-input matrix or data frame with 4 columns: 1st column for SNPs/eQTLs, 2nd column for Genes, 3rd for eQTL mapping significance level (p-values or FDR), and 4th for contexts (required even though only one context is input). Alternatively, it can be a file containing these 4 columns. It is designed to allow the user analysing their eQTL data. This customisation (if provided) will populate built-in eQTL data

`include.HiC` genes linked to input SNPs are also included. By default, it is 'NA' to disable this option. Otherwise, those genes linked to SNPs will be included according to Promoter Capture HiC (PCHiC) datasets. Pre-built HiC datasets are detailed in `xDefineHIC`

`cdf.function` a character specifying a Cumulative Distribution Function (cdf). It can be one of 'exponential' based on exponential cdf, 'empirical' for empirical cdf

`relative.importance`
a vector specifying the relative importance of nearby genes, eQTL genes and HiC genes. By default, it sets c(1/3, 1/3, 1/3)

`scoring.scheme` the method used to calculate seed gene scores under a set of SNPs. It can be one of "sum" for adding up, "max" for the maximum, and "sequential" for the sequential weighting. The sequential weighting is done via: $\sum_{i=1} \frac{R_i}{i}$, where R_i is the i^{th} rank (in a descreasing order)

`network` the built-in network. Currently two sources of network information are supported: the STRING database (version 10) and the Pathway Commons database (version 7). STRING is a meta-integration of undirect interactions from the functional aspect, while Pathways Commons mainly contains both undirect and direct interactions from the physical/pathway aspect. Both have scores to control the confidence of interactions. Therefore, the user can choose the different quality of the interactions. In STRING, "STRING_highest" indicates interactions with highest confidence (confidence scores>=900), "STRING_high" for interactions with high confidence (confidence scores>=700), "STRING_medium" for interactions with medium confidence (confidence scores>=400), and "STRING_low" for interactions with low confidence (confidence scores>=150). For undirect/physical interactions from Pathways Commons, "PCommonsUN_high" indicates undirect interactions with high confidence (supported with the PubMed references plus at least 2 different sources), "PCommonsUN_medium" for undirect interactions with medium confidence (supported with the PubMed references). For direct (pathway-merged) interactions from Pathways Commons, "PCommonsDN_high" indicates direct interactions with high confidence (supported with the PubMed references plus at least 2 different sources), and "PCommonsUN_medium" for direct interactions with medium confidence (supported with the PubMed references). In addition to pooled version of pathways from all data sources, the user can also choose the pathway-merged network from individual sources, that is, "PCommonsDN_Reactome" for those from Reactome, "PCommonsDN_KEGG" for those from KEGG, "PCommonsDN_HumanCyc" for those from HumanCyc, "PCommonsDN_PID" for those froom PID, "PCommonsDN_PANTHER" for those from PANTHER, "PCommonsDN_ReconX" for those from ReconX, "PCommonsDN_TRANSFAC" for those from TRANSFAC, "PCommonsDN_PhosphoSite" for those from PhosphoSite, and "PCommonsDN_CTD" for those from CTD. For direct (pathway-merged) interactions sourced from KEGG, it can be 'KEGG' for all, 'KEGG_metabolism' for pathways grouped into 'Metabolism', 'KEGG_genetic' for 'Genetic Information Processing' pathways, 'KEGG_environmental' for 'Environmental Information Processing' pathways, 'KEGG_cellular' for 'Cellular Processes' pathways, 'KEGG_organismal' for 'Organismal Systems' pathways, and 'KEGG_disease' for 'Human Diseases' pathways. 'REACTOME' for protein-protein interactions derived from Reactome pathways

`STRING.only` the further restriction of STRING by interaction type. If NA, no such restriction. Otherwide, it can be one or more of "neighborhood_score","fusion_score","cooccurence_score","coexpres
Useful options are c("experimental_score","database_score"): only experimental data (extracted from BIND, DIP, GRID, HPRD, IntAct, MINT, and PID) and

curated data (extracted from Biocarta, BioCyc, GO, KEGG, and Reactome) are used

`weighted` logical to indicate whether edge weights should be considered. By default, it sets to false. If true, it only works for the network from the STRING database

`network.customised`
an object of class "igraph". By default, it is NULL. It is designed to allow the user analysing their customised network data that are not listed in the above argument 'network'. This customisation (if provided) has the high priority over built-in network. If the user provides the "igraph" object with the "weight" edge attribute, RWR will assume to walk on the weighted network

`seeds.inclusive`
logical to indicate whether non-network seed genes are included for prioritisation. If TRUE (by default), these genes will be added to the netowrk

`normalise` the way to normalise the adjacency matrix of the input graph. It can be 'laplacian' for laplacian normalisation, 'row' for row-wise normalisation, 'column' for column-wise normalisation, or 'none'

`restart` the restart probability used for Random Walk with Restart (RWR). The restart probability takes the value from 0 to 1, controlling the range from the starting nodes/seeds that the walker will explore. The higher the value, the more likely the walker is to visit the nodes centered on the starting nodes. At the extreme when the restart probability is zero, the walker moves freely to the neighbors at each step without restarting from seeds, i.e., following a random walk (RW)

`normalise.affinity.matrix`
the way to normalise the output affinity matrix. It can be 'none' for no normalisation, 'quantile' for quantile normalisation to ensure that columns (if multiple) of the output affinity matrix have the same quantiles

`parallel` logical to indicate whether parallel computation with multicores is used. By default, it sets to true, but not necessarily does so. Partly because parallel backends available will be system-specific (now only Linux or Mac OS). Also, it will depend on whether these two packages "foreach" and "doMC" have been installed

`multicores` an integer to specify how many cores will be registered as the multicore parallel backend to the 'foreach' package. If NULL, it will use a half of cores available in a user's computer. This option only works when parallel computation is enabled

`verbose` logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

`RData.location` the characters to tell the location of built-in RData files. See `xRDataLoader` for details

`guid` a valid (5-character) Global Unique IDentifier for an OSF project. See `xRDataLoader` for details

Value

an object of class "pNode", a list with following components:

- priority: a matrix of nNode X 6 containing node priority information, where nNode is the number of nodes in the input graph, and the 6 columns are "name" (node names), "node" (1 for network genes, 0 for non-network seed genes), "seed" (1 for seeds, 0 for non-seeds), "weight" (weight values), "priority" (the priority scores that are rescaled to the range [0,1]), "rank" (ranks of the priority scores), "description" (node description)
- g: an input "igraph" object
- SNP: a data frame of nSNP X 4 containing input SNPs and/or LD SNPs info, where nSNP is the number of input SNPs and/or LD SNPs, and the 4 columns are "SNP" (dbSNP), "Score" (the SNP score), "Pval" (the SNP p-value), "Flag" (indicative of Lead SNPs or LD SNPs)
- Gene2SNP: a data frame of nPair X 3 containing Gene-SNP pair info, where nPair is the number of Gene-SNP pairs, and the 3 columns are "Gene" (seed genes), "SNP" (dbSNP), "Score" (an SNP's genetic influential score on a seed gene)
- nGenes: if not NULL, it is a data frame containing nGene-SNP pair info
- eGenes: if not NULL, it is a data frame containing eGene-SNP pair info per context
- cGenes: if not NULL, it is a data frame containing cGene-SNP pair info per context

Note

The prioritisation procedure (from SNPs to target genes) consists of following steps:

- i) xSNPscores used to calculate the SNP score.
- ii) xSNP2nGenes used to define and score the nearby genes.
- iii) xSNP2eGenes used to define and score the eQTL genes.
- iv) xSNP2cGenes used to define and score the HiC genes.
- v) define seed genes as the nearby genes in ii) and the eQTL genes in iii) and the HiC genes in iv), which are then scored in an integrative manner.
- vi) xPierGenes used to prioritise genes using an input graph and a list of seed genes and their scores from v). The priority score is the affinity score estimated by Random Walk with Restart (RWR), measured as the affinity of all nodes in the graph to the seeds.

See Also

xSNPscores, xSNP2nGenes, xSNP2eGenes, xSNP2cGenes, xSparseMatrix, xSM2DF, xPierGenes, xSM2DF, xDefineEQTL, xDefineHIC

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

## Not run:
# a) provide the SNPs with the significance info
data(ImmunoBase)
gr <- ImmunoBase$AS$variants
AS <- as.data.frame(GenomicRanges::mcols(gr)[, c('Variant','Pvalue')])

# b) perform priority analysis
pNode <- xPierSNPs(data=AS, include.TAD='GM12878',
```

```
include.eQTL="JKng_mono", include.HiC='Monocytes',
network="PCommonsUN_medium", restart=0.7,
RData.location=RData.location)

# c) save to the file called 'SNPs_priority.txt'
write.table(pNode$priority, file="SNPs_priority.txt", sep="\t",
row.names=FALSE)

# d) manhattan plot
mp <- xPierManhattan(pNode, top=20, top.label.size=1.5, y.scale="sqrt",
RData.location=RData.location)
#pdf(file="Gene_manhattan.pdf", height=6, width=12, compress=TRUE)
print(mp)
#dev.off()

## End(Not run)
```

---

xPierSNPsAdv *Function to prepare genetic predictors given a list of seed SNPs together with the significance level (e.g. GWAS reported p-values)*

Description

xPierSNPsAdv is supposed to prepare genetic predictors given a list of seed SNPs together with the significance level (e.g. GWAS reported p-values). Internally it calls xPierSNPs to prepare the distance predictor, the eQTL predictors (if required) and the HiC predictors (if required). It returns a list of class "pNode" objects.

Usage

```
xPierSNPsAdv(
data,
include.LD = NA,
LD.customised = NULL,
LD.r2 = 0.8,
significance.threshold = 5e-05,
score.cap = 10,
distance.max = 2000,
decay.kernel = c("slow", "constant", "linear", "rapid"),
decay.exponent = 2,
GR.SNP = c("dbSNP_GWAS", "dbSNP_Common", "dbSNP_Single"),
GR.Gene = c("UCSC_knownGene", "UCSC_knownCanonical"),
include.TAD = c("none", "GM12878", "IMR90", "MSC", "TRO", "H1", "MES",
"NPC"),
include.eQTL = NA,
eQTL.customised = NULL,
include.HiC = NA,
cdf.function = c("empirical", "exponential"),
```

```
scoring.scheme = c("max", "sum", "sequential"),
network = c("STRING_highest", "STRING_high", "STRING_medium",
"STRING_low",
"PCommonsUN_high", "PCommonsUN_medium", "PCommonsDN_high",
"PCommonsDN_medium",
"PCommonsDN_Reactome", "PCommonsDN_KEGG", "PCommonsDN_HumanCyc",
"PCommonsDN_PID",
"PCommonsDN_PANTHER", "PCommonsDN_ReconX", "PCommonsDN_TRANSFAC",
"PCommonsDN_PhosphoSite", "PCommonsDN_CTD", "KEGG", "KEGG_metabolism",
"KEGG_genetic", "KEGG_environmental", "KEGG_cellular",
"KEGG_organismal",
"KEGG_disease", "REACTOME"),
STRING.only = c(NA, "neighborhood_score", "fusion_score",
"cooccurence_score",
"coexpression_score", "experimental_score", "database_score",
"textmining_score")[1],
weighted = FALSE,
network.customised = NULL,
seeds.inclusive = TRUE,
normalise = c("laplacian", "row", "column", "none"),
restart = 0.7,
normalise.affinity.matrix = c("none", "quantile"),
parallel = TRUE,
multicores = NULL,
verbose = TRUE,
verbose.details = FALSE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

data	a named input vector containing the sinificance level for nodes (dbSNP). For this named vector, the element names are dbSNP ID (or in the format such as 'chr16:28525386'), the element values for the significance level (measured as p-value or fdr). Alternatively, it can be a matrix or data frame with two columns: 1st column for dbSNP, 2nd column for the significance level
include.LD	additional SNPs in LD with Lead SNPs are also included. By default, it is 'NA' to disable this option. Otherwise, LD SNPs will be included based on one or more of 5 super-populations from 1000 Genomics Project data (phase 3). They are "AFR", "AMR", "EAS", "EUR", and "SAS". Explanations for population code can be found at http://www.1000genomes.org/faq/which-populations-are-part-your-stuc
LD.customised	a user-input matrix or data frame with 3 columns: 1st column for Lead SNPs, 2nd column for LD SNPs, and 3rd for LD r2 value. It is designed to allow the user analysing their pre-calculated LD info. This customisation (if provided) has the high priority over built-in LD SNPs
LD.r2	the LD r2 value. By default, it is 0.8, meaning that SNPs in LD (r2>=0.8) with input SNPs will be considered as LD SNPs. It can be any value from 0.8 to 1

significance.threshold
: the given significance threshold. By default, it is set to NULL, meaning there is no constraint on the significance level when transforming the significance level of SNPs into scores. If given, those SNPs below this are considered significant and thus scored positively. Instead, those above this are considered insigificant and thus receive no score

score.cap
: the maximum score being capped. By default, it is set to 10. If NULL, no capping is applied

distance.max
: the maximum distance between genes and SNPs. Only those genes no far way from this distance will be considered as seed genes. This parameter will influence the distance-component weights calculated for nearby SNPs per gene

decay.kernel
: a character specifying a decay kernel function. It can be one of 'slow' for slow decay, 'linear' for linear decay, and 'rapid' for rapid decay. If no distance weight is used, please select 'constant'

decay.exponent
: an integer specifying a decay exponent. By default, it sets to 2

GR.SNP
: the genomic regions of SNPs. By default, it is 'dbSNP_GWAS', that is, SNPs from dbSNP (version 146) restricted to GWAS SNPs and their LD SNPs (hg19). It can be 'dbSNP_Common', that is, Common SNPs from dbSNP (version 146) plus GWAS SNPs and their LD SNPs (hg19). Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to dbSNP IDs. Then, tell "GR.SNP" with your RData file name (with or without extension), plus specify your file RData path in "RData.location". Note: you can also load your customised GR object directly

GR.Gene
: the genomic regions of genes. By default, it is 'UCSC_knownGene', that is, UCSC known genes (together with genomic locations) based on human genome assembly hg19. It can be 'UCSC_knownCanonical', that is, UCSC known canonical genes (together with genomic locations) based on human genome assembly hg19. Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to Gene Symbols. Then, tell "GR.Gene" with your RData file name (with or without extension), plus specify your file RData path in "RData.location". Note: you can also load your customised GR object directly

include.TAD
: TAD boundary regions are also included. By default, it is 'none' to disable this option. Otherwise, inclusion of a TAD dataset to pre-filter SNP-nGene pairs (i.e. only those within a TAD region will be kept). TAD datasets can be one of "GM12878" (lymphoblast), "IMR90" (fibroblast), "MSC" (mesenchymal stem cell) ,"TRO" (trophoblasts-like cell), "H1" (embryonic stem cell), "MES" (mesendoderm) and "NPC" (neural progenitor cell). Explanations can be found at http://dx.doi.org/10.1016/j.celrep.2016.10.061

include.eQTL
: the eQTL supported currently. By default, it is 'NA' to disable this option. Prebuilt eQTL datasets are detailed in xDefineEQTL

eQTL.customised
: a user-input matrix or data frame with 4 columns: 1st column for SNPs/eQTLs, 2nd column for Genes, 3rd for eQTL mapping significance level (p-values or

FDR), and 4th for contexts (required even though only one context is input). Alternatively, it can be a file containing these 4 columns. It is designed to allow the user analysing their eQTL data. This customisation (if provided) will populate built-in eQTL data

`include.HiC` genes linked to input SNPs are also included. By default, it is 'NA' to disable this option. Otherwise, those genes linked to SNPs will be included according to Promoter Capture HiC (PCHiC) datasets. Pre-built HiC datasets are detailed in `xDefineHIC`

`cdf.function` a character specifying a Cumulative Distribution Function (cdf). It can be one of 'exponential' based on exponential cdf, 'empirical' for empirical cdf

`scoring.scheme` the method used to calculate seed gene scores under a set of SNPs. It can be one of "sum" for adding up, "max" for the maximum, and "sequential" for the sequential weighting. The sequential weighting is done via: $\sum_{i=1} \frac{R_i}{i}$, where R_i is the i^{th} rank (in a descreasing order)

`network` the built-in network. Currently two sources of network information are supported: the STRING database (version 10) and the Pathway Commons database (version 7). STRING is a meta-integration of undirect interactions from the functional aspect, while Pathways Commons mainly contains both undirect and direct interactions from the physical/pathway aspect. Both have scores to control the confidence of interactions. Therefore, the user can choose the different quality of the interactions. In STRING, "STRING_highest" indicates interactions with highest confidence (confidence scores>=900), "STRING_high" for interactions with high confidence (confidence scores>=700), "STRING_medium" for interactions with medium confidence (confidence scores>=400), and "STRING_low" for interactions with low confidence (confidence scores>=150). For undirect/physical interactions from Pathways Commons, "PCommonsUN_high" indicates undirect interactions with high confidence (supported with the PubMed references plus at least 2 different sources), "PCommonsUN_medium" for undirect interactions with medium confidence (supported with the PubMed references). For direct (pathway-merged) interactions from Pathways Commons, "PCommonsDN_high" indicates direct interactions with high confidence (supported with the PubMed references plus at least 2 different sources), and "PCommonsUN_medium" for direct interactions with medium confidence (supported with the PubMed references). In addition to pooled version of pathways from all data sources, the user can also choose the pathway-merged network from individual sources, that is, "PCommonsDN_Reactome" for those from Reactome, "PCommonsDN_KEGG" for those from KEGG, "PCommonsDN_HumanCyc" for those from HumanCyc, "PCommonsDN_PID" for those froom PID, "PCommonsDN_PANTHER" for those from PANTHER, "PCommonsDN_ReconX" for those from ReconX, "PCommonsDN_TRANSFAC" for those from TRANSFAC, "PCommonsDN_PhosphoSite" for those from PhosphoSite, and "PCommonsDN_CTD" for those from CTD. For direct (pathway-merged) interactions sourced from KEGG, it can be 'KEGG' for all, 'KEGG_metabolism' for pathways grouped into 'Metabolism', 'KEGG_genetic' for 'Genetic Information Processing' pathways, 'KEGG_environmental' for 'Environmental Information Processing' pathways, 'KEGG_cellular' for 'Cellular Processes' pathways, 'KEGG_organismal' for 'Organismal Systems' pathways, and 'KEGG_disease' for 'Human Dis-

eases' pathways. 'REACTOME' for protein-protein interactions derived from Reactome pathways

`STRING.only` the further restriction of STRING by interaction type. If NA, no such restriction. Otherwide, it can be one or more of "neighborhood_score","fusion_score","cooccurence_score","coexpres Useful options are c("experimental_score","database_score"): only experimental data (extracted from BIND, DIP, GRID, HPRD, IntAct, MINT, and PID) and curated data (extracted from Biocarta, BioCyc, GO, KEGG, and Reactome) are used

`weighted` logical to indicate whether edge weights should be considered. By default, it sets to false. If true, it only works for the network from the STRING database

`network.customised`
an object of class "igraph". By default, it is NULL. It is designed to allow the user analysing their customised network data that are not listed in the above argument 'network'. This customisation (if provided) has the high priority over built-in network. If the user provides the "igraph" object with the "weight" edge attribute, RWR will assume to walk on the weighted network

`seeds.inclusive`
logical to indicate whether non-network seed genes are included for prioritisation. If TRUE (by default), these genes will be added to the netowrk

`normalise` the way to normalise the adjacency matrix of the input graph. It can be 'laplacian' for laplacian normalisation, 'row' for row-wise normalisation, 'column' for column-wise normalisation, or 'none'

`restart` the restart probability used for Random Walk with Restart (RWR). The restart probability takes the value from 0 to 1, controlling the range from the starting nodes/seeds that the walker will explore. The higher the value, the more likely the walker is to visit the nodes centered on the starting nodes. At the extreme when the restart probability is zero, the walker moves freely to the neighbors at each step without restarting from seeds, i.e., following a random walk (RW)

`normalise.affinity.matrix`
the way to normalise the output affinity matrix. It can be 'none' for no normalisation, 'quantile' for quantile normalisation to ensure that columns (if multiple) of the output affinity matrix have the same quantiles

`parallel` logical to indicate whether parallel computation with multicores is used. By default, it sets to true, but not necessarily does so. Partly because parallel backends available will be system-specific (now only Linux or Mac OS). Also, it will depend on whether these two packages "foreach" and "doMC" have been installed

`multicores` an integer to specify how many cores will be registered as the multicore parallel backend to the 'foreach' package. If NULL, it will use a half of cores available in a user's computer. This option only works when parallel computation is enabled

`verbose` logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

`verbose.details`
logical to indicate whether the detailed messages from being-called functions will be displayed in the screen. By default, it sets to FALSE enabling messages

RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details
guid	a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details

Value

A list of class "pNode" objects, each object having a list with following components:

- priority: a matrix of nNode X 6 containing node priority information, where nNode is the number of nodes in the input graph, and the 6 columns are "name" (node names), "node" (1 for network genes, 0 for non-network seed genes), "seed" (1 for seeds, 0 for non-seeds), "weight" (weight values), "priority" (the priority scores that are rescaled to the range [0,1]), "rank" (ranks of the priority scores), "description" (node description)
- g: an input "igraph" object
- SNP: a data frame of nSNP X 4 containing input SNPs and/or LD SNPs info, where nSNP is the number of input SNPs and/or LD SNPs, and the 4 columns are "SNP" (dbSNP), "Score" (the SNP score), "Pval" (the SNP p-value), "Flag" (indicative of Lead SNPs or LD SNPs)
- Gene2SNP: a data frame of nPair X 3 containing Gene-SNP pair info, where nPair is the number of Gene-SNP pairs, and the 3 columns are "Gene" (seed genes), "SNP" (dbSNP), "Score" (an SNP's genetic influential score on a seed gene)
- nGenes: if not NULL, it is a data frame containing nGene-SNP pair info
- eGenes: if not NULL, it is a data frame containing eGene-SNP pair info per context
- cGenes: if not NULL, it is a data frame containing cGene-SNP pair info per context

Note

This function calls xPierSNPs in a loop way generating the distance predictor, the eQTL predictors (if required) and the HiC predictors (if required).

See Also

xPierSNPs

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# a) provide the SNPs with the significance info
data(ImmunoBase)
gr <- ImmunoBase$AS$variants
AS <- as.data.frame(GenomicRanges::mcols(gr)[, c('Variant','Pvalue')])

# b) perform priority analysis
ls_pNode <- xPierSNPsAdv(data=AS, include.TAD='GM12878',
include.eQTL="JKng_mono", include.HiC='Monocytes',
network="PCommonsUN_medium", restart=0.7,
RData.location=RData.location)
#ls_pNode <- xPierSNPsAdv(data=AS, include.TAD='GM12878', include.eQTL="JKng_mono", include.HiC='Monocytes', net
```

```
## End(Not run)
```

---

xPierSNPsAdvABF *Function to prepare genetic predictors given GWAS summary data with eGenes identified through ABF*

Description

xPierSNPsAdvABF is supposed to prepare genetic predictors given GWAS summary data with eGenes identified through ABF. Internally it calls xPierSNPs to prepare the distance predictor and the HiC predictors (if required), and xPierABF to prepare the eQTL predictors (if required). It returns a list of class "pNode" objects.

Usage

```
xPierSNPsAdvABF(
data,
include.LD = NA,
LD.customised = NULL,
LD.r2 = 0.8,
significance.threshold = 5e-05,
score.cap = 10,
distance.max = 2000,
decay.kernel = c("slow", "constant", "linear", "rapid"),
decay.exponent = 2,
GR.SNP = c("dbSNP_GWAS", "dbSNP_Common", "dbSNP_Single"),
GR.Gene = c("UCSC_knownGene", "UCSC_knownCanonical"),
include.TAD = c("none", "GM12878", "IMR90", "MSC", "TRO", "H1", "MES",
"NPC"),
include.eQTL = c("CD14", "LPS2", "LPS24", "IFN", "Bcell", "NK",
"Neutrophil", "CD4",
"CD8", "Blood", "Monocyte", "shared_CD14", "shared_LPS2",
"shared_LPS24",
"shared_IFN"),
include.HiC = NA,
cdf.function = c("empirical", "exponential"),
scoring.scheme = c("max", "sum", "sequential"),
network = c("STRING_highest", "STRING_high", "STRING_medium",
"STRING_low",
"PCommonsUN_high", "PCommonsUN_medium", "PCommonsDN_high",
"PCommonsDN_medium",
"PCommonsDN_Reactome", "PCommonsDN_KEGG", "PCommonsDN_HumanCyc",
"PCommonsDN_PID",
"PCommonsDN_PANTHER", "PCommonsDN_ReconX", "PCommonsDN_TRANSFAC",
"PCommonsDN_PhosphoSite", "PCommonsDN_CTD", "KEGG", "KEGG_metabolism",
"KEGG_genetic", "KEGG_environmental", "KEGG_cellular",
```

```
"KEGG_organismal",
"KEGG_disease", "REACTOME"),
STRING.only = c(NA, "neighborhood_score", "fusion_score",
"cooccurence_score",
"coexpression_score", "experimental_score", "database_score",
"textmining_score")[1],
weighted = FALSE,
network.customised = NULL,
seeds.inclusive = TRUE,
normalise = c("laplacian", "row", "column", "none"),
restart = 0.7,
normalise.affinity.matrix = c("none", "quantile"),
parallel = TRUE,
multicores = NULL,
verbose = TRUE,
verbose.details = FALSE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL,
...
)
```

Arguments

data	a data frame storing GWAS summary data with following required columns 'snp', 'p' (p-value), 'effect' (the effect allele assessed), 'other' (other allele), 'b' (effect size for the allele assessed; log(odds ratio) for a case-control study), 'se' (standard error), 'suggestive' (logical, and those false only to define nGene/cGene; all used to define eGene)
include.LD	additional SNPs in LD with Lead SNPs are also included. By default, it is 'NA' to disable this option. Otherwise, LD SNPs will be included based on one or more of 5 super-populations from 1000 Genomics Project data (phase 3). They are "AFR", "AMR", "EAS", "EUR", and "SAS". Explanations for population code can be found at http://www.1000genomes.org/faq/which-populations-are-part-your-stuc
LD.customised	a user-input matrix or data frame with 3 columns: 1st column for Lead SNPs, 2nd column for LD SNPs, and 3rd for LD r2 value. It is designed to allow the user analysing their pre-calculated LD info. This customisation (if provided) has the high priority over built-in LD SNPs
LD.r2	the LD r2 value. By default, it is 0.8, meaning that SNPs in LD (r2>=0.8) with input SNPs will be considered as LD SNPs. It can be any value from 0.8 to 1
significance.threshold	the given significance threshold. By default, it is set to NULL, meaning there is no constraint on the significance level when transforming the significance level of SNPs into scores. If given, those SNPs below this are considered significant and thus scored positively. Instead, those above this are considered insigificant and thus receive no score
score.cap	the maximum score being capped. By default, it is set to 10. If NULL, no capping is applied

`distance.max` the maximum distance between genes and SNPs. Only those genes no far way from this distance will be considered as seed genes. This parameter will influence the distance-component weights calculated for nearby SNPs per gene

`decay.kernel` a character specifying a decay kernel function. It can be one of 'slow' for slow decay, 'linear' for linear decay, and 'rapid' for rapid decay. If no distance weight is used, please select 'constant'

`decay.exponent` an integer specifying a decay exponent. By default, it sets to 2

`GR.SNP` the genomic regions of SNPs. By default, it is 'dbSNP_GWAS', that is, SNPs from dbSNP (version 146) restricted to GWAS SNPs and their LD SNPs (hg19). It can be 'dbSNP_Common', that is, Common SNPs from dbSNP (version 146) plus GWAS SNPs and their LD SNPs (hg19). Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to dbSNP IDs. Then, tell "GR.SNP" with your RData file name (with or without extension), plus specify your file RData path in "RData.location". Note: you can also load your customised GR object directly

`GR.Gene` the genomic regions of genes. By default, it is 'UCSC_knownGene', that is, UCSC known genes (together with genomic locations) based on human genome assembly hg19. It can be 'UCSC_knownCanonical', that is, UCSC known canonical genes (together with genomic locations) based on human genome assembly hg19. Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to Gene Symbols. Then, tell "GR.Gene" with your RData file name (with or without extension), plus specify your file RData path in "RData.location". Note: you can also load your customised GR object directly

`include.TAD` TAD boundary regions are also included. By default, it is 'none' to disable this option. Otherwise, inclusion of a TAD dataset to pre-filter SNP-nGene pairs (i.e. only those within a TAD region will be kept). TAD datasets can be one of "GM12878" (lymphoblast), "IMR90" (fibroblast), "MSC" (mesenchymal stem cell) ,"TRO" (trophoblasts-like cell), "H1" (embryonic stem cell), "MES" (mesendoderm) and "NPC" (neural progenitor cell). Explanations can be found at http://dx.doi.org/10.1016/j.celrep.2016.10.061

`include.eQTL` the context-specific eQTL summary data supported currently. Contexts include "Bcell", "Blood", "CD14", "CD4", "CD8", "IFN", "LPS24", "LPS2", "Monocyte", "Neutrophil", "NK", "shared_CD14", "shared_IFN", "shared_LPS24", "shared_LPS2"

`include.HiC` genes linked to input SNPs are also included. By default, it is 'NA' to disable this option. Otherwise, those genes linked to SNPs will be included according to Promoter Capture HiC (PCHiC) datasets. Pre-built HiC datasets are detailed in `xDefineHIC`

`cdf.function` a character specifying a Cumulative Distribution Function (cdf). It can be one of 'exponential' based on exponential cdf, 'empirical' for empirical cdf

`scoring.scheme` the method used to calculate seed gene scores under a set of SNPs. It can be one of "sum" for adding up, "max" for the maximum, and "sequential" for the sequential weighting. The sequential weighting is done via: $\sum_{i=1} \frac{R_i}{i}$, where R_i is the i^{th} rank (in a descreasing order)

network the built-in network. Currently two sources of network information are supported: the STRING database (version 10) and the Pathway Commons database (version 7). STRING is a meta-integration of undirect interactions from the functional aspect, while Pathways Commons mainly contains both undirect and direct interactions from the physical/pathway aspect. Both have scores to control the confidence of interactions. Therefore, the user can choose the different quality of the interactions. In STRING, "STRING_highest" indicates interactions with highest confidence (confidence scores>=900), "STRING_high" for interactions with high confidence (confidence scores>=700), "STRING_medium" for interactions with medium confidence (confidence scores>=400), and "STRING_low" for interactions with low confidence (confidence scores>=150). For undirect/physical interactions from Pathways Commons, "PCommonsUN_high" indicates undirect interactions with high confidence (supported with the PubMed references plus at least 2 different sources), "PCommonsUN_medium" for undirect interactions with medium confidence (supported with the PubMed references). For direct (pathway-merged) interactions from Pathways Commons, "PCommonsDN_high" indicates direct interactions with high confidence (supported with the PubMed references plus at least 2 different sources), and "PCommonsUN_medium" for direct interactions with medium confidence (supported with the PubMed references). In addition to pooled version of pathways from all data sources, the user can also choose the pathway-merged network from individual sources, that is, "PCommonsDN_Reactome" for those from Reactome, "PCommonsDN_KEGG" for those from KEGG, "PCommonsDN_HumanCyc" for those from HumanCyc, "PCommonsDN_PID" for those froom PID, "PCommonsDN_PANTHER" for those from PANTHER, "PCommonsDN_ReconX" for those from ReconX, "PCommonsDN_TRANSFAC" for those from TRANSFAC, "PCommonsDN_PhosphoSite" for those from PhosphoSite, and "PCommonsDN_CTD" for those from CTD. For direct (pathway-merged) interactions sourced from KEGG, it can be 'KEGG' for all, 'KEGG_metabolism' for pathways grouped into 'Metabolism', 'KEGG_genetic' for 'Genetic Information Processing' pathways, 'KEGG_environmental' for 'Environmental Information Processing' pathways, 'KEGG_cellular' for 'Cellular Processes' pathways, 'KEGG_organismal' for 'Organismal Systems' pathways, and 'KEGG_disease' for 'Human Diseases' pathways. 'REACTOME' for protein-protein interactions derived from Reactome pathways

STRING.only the further restriction of STRING by interaction type. If NA, no such restriction. Otherwide, it can be one or more of "neighborhood_score","fusion_score","cooccurence_score","coexpres
Useful options are c("experimental_score","database_score"): only experimental data (extracted from BIND, DIP, GRID, HPRD, IntAct, MINT, and PID) and curated data (extracted from Biocarta, BioCyc, GO, KEGG, and Reactome) are used

weighted logical to indicate whether edge weights should be considered. By default, it sets to false. If true, it only works for the network from the STRING database

network.customised
an object of class "igraph". By default, it is NULL. It is designed to allow the user analysing their customised network data that are not listed in the above argument 'network'. This customisation (if provided) has the high priority over built-in network. If the user provides the "igraph" object with the "weight" edge

attribute, RWR will assume to walk on the weighted network

`seeds.inclusive` logical to indicate whether non-network seed genes are included for prioritisation. If TRUE (by default), these genes will be added to the netowrk

`normalise` the way to normalise the adjacency matrix of the input graph. It can be 'laplacian' for laplacian normalisation, 'row' for row-wise normalisation, 'column' for column-wise normalisation, or 'none'

`restart` the restart probability used for Random Walk with Restart (RWR). The restart probability takes the value from 0 to 1, controlling the range from the starting nodes/seeds that the walker will explore. The higher the value, the more likely the walker is to visit the nodes centered on the starting nodes. At the extreme when the restart probability is zero, the walker moves freely to the neighbors at each step without restarting from seeds, i.e., following a random walk (RW)

`normalise.affinity.matrix` the way to normalise the output affinity matrix. It can be 'none' for no normalisation, 'quantile' for quantile normalisation to ensure that columns (if multiple) of the output affinity matrix have the same quantiles

`parallel` logical to indicate whether parallel computation with multicores is used. By default, it sets to true, but not necessarily does so. Partly because parallel backends available will be system-specific (now only Linux or Mac OS). Also, it will depend on whether these two packages "foreach" and "doMC" have been installed

`multicores` an integer to specify how many cores will be registered as the multicore parallel backend to the 'foreach' package. If NULL, it will use a half of cores available in a user's computer. This option only works when parallel computation is enabled

`verbose` logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

`verbose.details` logical to indicate whether the detailed messages from being-called functions will be displayed in the screen. By default, it sets to FALSE enabling messages

`RData.location` the characters to tell the location of built-in RData files. See `xRDataLoader` for details

`guid` a valid (5-character) Global Unique IDentifier for an OSF project. See `xRDataLoader` for details

`...` additional parameters used in xPierABF ("prior.eqtl", "prior.gwas", "prior.both", "cutoff.H4", "cutoff.pgwas")

Value

A list of class "pNode" objects, each object having a list with following components:

- `priority`: a matrix of nNode X 6 containing node priority information, where nNode is the number of nodes in the input graph, and the 6 columns are "name" (node names), "node" (1 for network genes, 0 for non-network seed genes), "seed" (1 for seeds, 0 for non-seeds), "weight" (weight values), "priority" (the priority scores that are rescaled to the range [0,1]), "rank" (ranks of the priority scores), "description" (node description)

- g: an input "igraph" object
- SNP: a data frame of nSNP X 4 containing input SNPs and/or LD SNPs info, where nSNP is the number of input SNPs and/or LD SNPs, and the 4 columns are "SNP" (dbSNP), "Score" (the SNP score), "Pval" (the SNP p-value), "Flag" (indicative of Lead SNPs or LD SNPs)
- Gene2SNP: a data frame of nPair X 3 containing Gene-SNP pair info, where nPair is the number of Gene-SNP pairs, and the 3 columns are "Gene" (seed genes), "SNP" (dbSNP), "Score" (an SNP's genetic influential score on a seed gene)
- nGenes: if not NULL, it is a data frame containing nGene-SNP pair info
- eGenes: if not NULL, it is a data frame containing eGene-SNP pair info per context
- cGenes: if not NULL, it is a data frame containing cGene-SNP pair info per context

Note

This function calls xPierSNPs in a loop way generating the distance predictor, the eQTL predictors (if required) and the HiC predictors (if required).

See Also

xPierSNPs, xPierABF, xPierSNPs

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
data <- utils::read.delim(file="summary_gwas.RA.txt", header=T,
row.names=NULL, stringsAsFactors=F)

# b) perform priority analysis
ls_pNode <- xPierSNPsAdvABF(data=AS, include.TAD='GM12878',
include.eQTL="Blood", include.HiC='Monocytes',
network="PCommonsUN_medium", restart=0.7,
RData.location=RData.location)

## End(Not run)
```

---

xPierSubnet *Function to identify a gene network from top prioritised genes*

Description

xPierSubnet is supposed to identify maximum-scoring gene subnetwork from a graph with the node information on priority scores, both are part of an object of class "pNode". It returns an object of class "igraph".

Usage

```
xPierSubnet(
pNode,
priority.quantile = 0.1,
network = c(NA, "STRING_highest", "STRING_high", "STRING_medium",
"STRING_low",
"PCommonsUN_high", "PCommonsUN_medium", "PCommonsDN_high",
"PCommonsDN_medium",
"PCommonsDN_Reactome", "PCommonsDN_KEGG", "PCommonsDN_HumanCyc",
"PCommonsDN_PID",
"PCommonsDN_PANTHER", "PCommonsDN_ReconX", "PCommonsDN_TRANSFAC",
"PCommonsDN_PhosphoSite", "PCommonsDN_CTD", "KEGG", "KEGG_metabolism",
"KEGG_genetic", "KEGG_environmental", "KEGG_cellular",
"KEGG_organismal",
"KEGG_disease", "REACTOME"),
STRING.only = c(NA, "neighborhood_score", "fusion_score",
"cooccurence_score",
"coexpression_score", "experimental_score", "database_score",
"textmining_score")[1],
network.customised = NULL,
subnet.significance = 0.01,
subnet.size = NULL,
test.permutation = FALSE,
num.permutation = 100,
respect = c("none", "degree"),
aggregateBy = c("Ztransform", "fishers", "logistic", "orderStatistic"),
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

pNode	an object of class "pNode" (or "sTarget" or "dTarget")
priority.quantile	the quantile of the top priority genes. By default, 10 analysis. If NULL or NA, all prioritised genes will be used
network	the built-in network. Currently two sources of network information are supported: the STRING database (version 10) and the Pathway Commons database (version 7). STRING is a meta-integration of undirect interactions from the functional aspect, while Pathways Commons mainly contains both undirect and direct interactions from the physical/pathway aspect. Both have scores to control the confidence of interactions. Therefore, the user can choose the different quality of the interactions. In STRING, "STRING_highest" indicates interactions with highest confidence (confidence scores>=900), "STRING_high" for interactions with high confidence (confidence scores>=700), "STRING_medium" for interactions with medium confidence (confidence scores>=400), and "STRING_low" for interactions with low confidence (confidence scores>=150). For undirect/physical

interactions from Pathways Commons, "PCommonsUN_high" indicates undirect interactions with high confidence (supported with the PubMed references plus at least 2 different sources), "PCommonsUN_medium" for undirect interactions with medium confidence (supported with the PubMed references). For direct (pathway-merged) interactions from Pathways Commons, "PCommonsDN_high" indicates direct interactions with high confidence (supported with the PubMed references plus at least 2 different sources), and "PCommonsUN_medium" for direct interactions with medium confidence (supported with the PubMed references). In addition to pooled version of pathways from all data sources, the user can also choose the pathway-merged network from individual sources, that is, "PCommonsDN_Reactome" for those from Reactome, "PCommonsDN_KEGG" for those from KEGG, "PCommonsDN_HumanCyc" for those from HumanCyc, "PCommonsDN_PID" for those froom PID, "PCommonsDN_PANTHER" for those from PANTHER, "PCommonsDN_ReconX" for those from ReconX, "PCommonsDN_TRANSFAC" for those from TRANSFAC, "PCommonsDN_PhosphoSite" for those from PhosphoSite, and "PCommonsDN_CTD" for those from CTD. For direct (pathway-merged) interactions sourced from KEGG, it can be 'KEGG' for all, 'KEGG_metabolism' for pathways grouped into 'Metabolism', 'KEGG_genetic' for 'Genetic Information Processing' pathways, 'KEGG_environmental' for 'Environmental Information Processing' pathways, 'KEGG_cellular' for 'Cellular Processes' pathways, 'KEGG_organismal' for 'Organismal Systems' pathways, and 'KEGG_disease' for 'Human Diseases' pathways. 'REACTOME' for protein-protein interactions derived from Reactome pathways

`STRING.only` the further restriction of STRING by interaction type. If NA, no such restriction. Otherwide, it can be one or more of "neighborhood_score","fusion_score","cooccurence_score","coexpres Useful options are c("experimental_score","database_score"): only experimental data (extracted from BIND, DIP, GRID, HPRD, IntAct, MINT, and PID) and curated data (extracted from Biocarta, BioCyc, GO, KEGG, and Reactome) are used

`network.customised`
an object of class "igraph". By default, it is NULL. It is designed to allow the user analysing their customised network data that are not listed in the above argument 'network'. This customisation (if provided) has the high priority over built-in network

`subnet.significance`
the given significance threshold. By default, it is set to NULL, meaning there is no constraint on nodes/genes. If given, those nodes/genes with p-values below this are considered significant and thus scored positively. Instead, those p-values above this given significance threshold are considered insigificant and thus scored negatively

`subnet.size` the desired number of nodes constrained to the resulting subnet. It is not nulll, a wide range of significance thresholds will be scanned to find the optimal significance threshold leading to the desired number of nodes in the resulting subnet. Notably, the given significance threshold will be overwritten by this option

`test.permutation`
logical to indicate whether the permutation test is perform to estimate the significance of identified network with the same number of nodes. By default, it sets

	to false
num.permutation	the number of permutations generating the null distribution of the identified network
respect	how to respect nodes to be sampled. It can be one of 'none' (randomly sampling) and 'degree' (degree-preserving sampling)
aggregateBy	the aggregate method used to aggregate edge confidence p-values. It can be either "orderStatistic" for the method based on the order statistics of p-values, or "fishers" for Fisher's method, "Ztransform" for Z-transform method, "logistic" for the logistic method. Without loss of generality, the Z-transform method does well in problems where evidence against the combined null is spread widely (equal footings) or when the total evidence is weak; Fisher's method does best in problems where the evidence is concentrated in a relatively small fraction of the individual tests or when the evidence is at least moderately strong; the logistic method provides a compromise between these two. Notably, the aggregate methods 'Ztransform' and 'logistic' are preferred here
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details
guid	a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details

Value

a subgraph with a maximum score, an object of class "igraph". It has ndoe attributes: signficance, score, type, priority (part of the "pNode" object). If permutation test is enabled, it also has a graph attribute (combinedP) and an edge attribute (edgeConfidence)

Note

The priority score will be first scaled to the range x=[0 100] and then is converted to pvalue-like significant level: 10^(-x). Next, xSubneterGenes is used to identify a maximum-scoring gene subnetwork that contains as many highly prioritised genes as possible but a few lowly prioritised genes as linkers. An iterative procedure of scanning different priority thresholds is also used to identify the network with a desired number of nodes/genes. Notably, the preferential use of the same network as used in gene-level prioritisation is due to the fact that gene-level affinity/priority scores are smoothly distributed over the network after being walked. In other words, the chance of identifying such a gene network enriched with top prioritised genes is much higher.

See Also

xSubneterGenes

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
```

```
# a) provide the SNPs with the significance info
## get lead SNPs reported in AS GWAS and their significance info (p-values)
#data.file <- "http://galahad.well.ox.ac.uk/bigdata/AS.txt"
#AS <- read.delim(data.file, header=TRUE, stringsAsFactors=FALSE)
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
gr <- ImmunoBase$AS$variants
AS <- as.data.frame(GenomicRanges::mcols(gr)[, c('Variant','Pvalue')])

# b) perform priority analysis
pNode <- xPierSNPs(data=AS, include.eQTL="JKng_mono",
include.HiC='Monocytes', network="PCommonsUN_medium", restart=0.7,
RData.location=RData.location)

# c) perform network analysis
# find maximum-scoring subnet with the desired node number=50
subnet <- xPierSubnet(pNode, priority.quantile=0.1, subnet.size=50,
RData.location=RData.location)

# d) save subnet results to the files called 'subnet_edges.txt' and 'subnet_nodes.txt'
output <- igraph::get.data.frame(subnet, what="edges")
utils::write.table(output, file="subnet_edges.txt", sep="\t",
row.names=FALSE)
output <- igraph::get.data.frame(subnet, what="vertices")
utils::write.table(output, file="subnet_nodes.txt", sep="\t",
row.names=FALSE)

# e) visualise the identified subnet
## do visualisation with nodes colored according to the priority
xVisNet(g=subnet, pattern=V(subnet)$priority, vertex.shape="sphere")
## do visualisation with nodes colored according to pvalue-like signficance
xVisNet(g=subnet, pattern=-log10(as.numeric(V(subnet)$significance)),
vertex.shape="sphere", colormap="wyr")

# f) visualise the identified subnet as a circos plot
library(RCircos)
xCircos(g=subnet, entity="Gene", RData.location=RData.location)

## End(Not run)
```

`xPierTrack` *Function to visualise a prioritised gene using track plot*

Description

`xPierTrack` is supposed to visualise a prioritised gene using track plot. Priority for the gene in query is displayed on the data track and nearby genes on the annotation track. Genomic locations on the X-axis are indicated on the X-axis, and the gene in query is highlighted. If SNPs are also provided, SNP annotation track will be also displayed at the bottom.

Usage

```
xPierTrack(
pNode,
priority.top = NULL,
target.query = NULL,
window = 1e+06,
nearby = NULL,
query.highlight = TRUE,
track.ideogram = TRUE,
track.genomeaxis = TRUE,
name.datatrack = "5-star rating\n(Priority index)",
name.annotrack = "Targets",
GR.Gene = c("UCSC_knownGene", "UCSC_knownCanonical"),
SNPs = NULL,
max.num.SNPs = 50,
GR.SNP = c("dbSNP_GWAS", "dbSNP_Common", "dbSNP_Single"),
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL,
...
)
```

Arguments

pNode	an object of class "pNode" (or "sTarget" or "dTarget")
priority.top	the number of the top targets used for track plot. By default, it is NULL meaning all targets are used
target.query	which gene in query will be visualised. If NULL, the target gene with the top priority will be displayed
window	the maximum distance defining nearby genes around the target gene in query. By default it is 1e6
nearby	the maximum number defining nearby genes around the target gene in query. By default it is NULL. If not NULL, it will overwrite the parameter 'window'
query.highlight	logical to indicate whether the gene in query will be highlighted
track.ideogram	logical to indicate whether ideogram track is shown. By default, it is TRUE
track.genomeaxis	logical to indicate whether genome axis track is shown. By default, it is TRUE
name.datatrack	the name for the data track. By default, it is "Priority index"
name.annotrack	the name for the annotation track. By default, it is "Genes". If NULL, the title for annotation track will be hided
GR.Gene	the genomic regions of genes. By default, it is 'UCSC_knownGene', that is, UCSC known genes (together with genomic locations) based on human genome assembly hg19. It can be 'UCSC_knownCanonical', that is, UCSC known canonical genes (together with genomic locations) based on human genome assembly hg19. Alternatively, the user can specify the customised input. To do so,

	first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to Gene Symbols. Then, tell "GR.Gene" with your RData file name (with or without extension), plus specify your file RData path in "RData.location"
SNPs	a input vector containing SNPs. SNPs should be provided as dbSNP ID (ie starting with rs). Alternatively, they can be in the format of 'chrN:xxx', where N is either 1-22 or X, xxx is genomic positional number; for example, 'chr16:28525386'. By default, it is NULL meaning the SNP annotation track will be not displayed
max.num.SNPs	the maximum number (50 by default) of SNPs to be shown. If NULL, no such restriction. Also this parameter only works when the SNP annotation track is enabled
GR.SNP	the genomic regions of SNPs. By default, it is 'dbSNP_GWAS', that is, SNPs from dbSNP (version 146) restricted to GWAS SNPs and their LD SNPs (hg19). It can be 'dbSNP_Common', that is, Common SNPs from dbSNP (version 146) plus GWAS SNPs and their LD SNPs (hg19). Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to dbSNP IDs. Then, tell "GR.SNP" with your RData file name (with or without extension), plus specify your file RData path in "RData.location". Note: you can also load your customised GR object directly
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to false for no display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details
guid	a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details
...	additional graphic parameters. For example, the parameter "add" allows the plot added to an existing plotting canvas without re-initialising. See http://www.rdocumentation.org/packages/Gviz/topics/plotTracks for the complete list.

Value

a list of GenomeGraph tracks, each one augmented by the computed image map coordinates in the 'imageMap' slot, along with the additional 'ImageMap' object 'titles' containing information about the title panels.

Note

none

See Also

xRDataLoader, xSNPlocations, xGR

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# a) provide the SNPs with the significance info
## get lead SNPs reported in AS GWAS and their significance info (p-values)
#data.file <- "http://galahad.well.ox.ac.uk/bigdata/AS.txt"
#AS <- read.delim(data.file, header=TRUE, stringsAsFactors=FALSE)
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
gr <- ImmunoBase$AS$variants
AS <- as.data.frame(GenomicRanges::mcols(gr)[, c('Variant','Pvalue')])

# b) perform priority analysis
pNode <- xPierSNPs(data=AS, include.eQTL="JKng_mono",
include.HiC='Monocytes', network="PCommonsUN_medium", restart=0.7,
RData.location=RData.location)

# c) track plot
library(Gviz)
#pdf(file="Gene_tracks.pdf", height=4, width=10, compress=TRUE)
xPierTrack(pNode, RData.location=RData.location)
#dev.off()
xPierTrack(pNode, priority.top=1000, nearby=20,
RData.location=RData.location)

## End(Not run)
```

---

xPierTrackAdv *Function to visualise a list of prioritised genes using advanced track plot*

Description

xPierTrackAdv is supposed to visualise prioritised genes using advanced track plot. Internally, it calls the function 'xPierTrack' per gene.

Usage

```
xPierTrackAdv(
pNode,
priority.top = NULL,
targets.query = NULL,
window = 1e+06,
nearby = NULL,
query.highlight = TRUE,
track.ideogram = TRUE,
track.genomeaxis = TRUE,
name.datatrack = "Priority index",
```

```
name.annotrack = "Genes",
GR.Gene = c("UCSC_knownGene", "UCSC_knownCanonical"),
SNPs = NULL,
GR.SNP = c("dbSNP_GWAS", "dbSNP_Common", "dbSNP_Single"),
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL,
...
)
```

Arguments

pNode	an object of class "pNode" (or "sTarget" or "dTarget")
priority.top	the number of the top targets used for track plot. By default, it is NULL meaning all targets are used
targets.query	which genes in query will be visualised. If NULL, the target gene with the top priority will be displayed
window	the maximum distance defining nearby genes around the target gene in query. By default it is 1e6
nearby	the maximum number defining nearby genes around the target gene in query. By default it is NULL. If not NULL, it will overwrite the parameter 'window'
query.highlight	logical to indicate whether the gene in query will be highlighted
track.ideogram	logical to indicate whether ideogram track is shown. By default, it is TRUE
track.genomeaxis	logical to indicate whether genome axis track is shown. By default, it is TRUE
name.datatrack	the name for the data track. By default, it is "Priority index"
name.annotrack	the name for the annotation track. By default, it is "Target genes"
GR.Gene	the genomic regions of genes. By default, it is 'UCSC_knownGene', that is, UCSC known genes (together with genomic locations) based on human genome assembly hg19. It can be 'UCSC_knownCanonical', that is, UCSC known canonical genes (together with genomic locations) based on human genome assembly hg19. Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to Gene Symbols. Then, tell "GR.Gene" with your RData file name (with or without extension), plus specify your file RData path in "RData.location"
SNPs	a input vector containing SNPs. SNPs should be provided as dbSNP ID (ie starting with rs). Alternatively, they can be in the format of 'chrN:xxx', where N is either 1-22 or X, xxx is genomic positional number; for example, 'chr16:28525386'. By default, it is NLL meaning the SNP annotation track will be not displayed
GR.SNP	the genomic regions of SNPs. By default, it is 'dbSNP_GWAS', that is, SNPs from dbSNP (version 146) restricted to GWAS SNPs and their LD SNPs (hg19). It can be 'dbSNP_Common', that is, Common SNPs from dbSNP (version 146) plus GWAS SNPs and their LD SNPs (hg19). Alternatively, the user can specify

	the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to dbSNP IDs. Then, tell "GR.SNP" with your RData file name (with or without extension), plus specify your file RData path in "RData.location". Note: you can also load your customised GR object directly
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to false for no display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details
guid	a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details
...	additional graphic parameters. For example, the parameter "strip" allows the panel title is hided (FALSE), shown (TRUE) or without the background (lattice::strip.custom(bg="transparent")); the parameter "layout" allows specification of the layout (the first element for the columns and the second element for the rows). See http://www.rdocumentation.org/packages/lattice/topics/xyplot for the complete list.

Value

an object of class "trellis"

Note

none

See Also

xRDataLoader

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# a) provide the SNPs with the significance info
## get lead SNPs reported in AS GWAS and their significance info (p-values)
#data.file <- "http://galahad.well.ox.ac.uk/bigdata/AS.txt"
#AS <- read.delim(data.file, header=TRUE, stringsAsFactors=FALSE)
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
gr <- ImmunoBase$AS$variants
AS <- as.data.frame(GenomicRanges::mcols(gr)[, c('Variant','Pvalue')])

# b) perform priority analysis
pNode <- xPierSNPs(data=AS, include.eQTL="JKng_mono",
include.HiC='Monocytes', network="PCommonsUN_medium", restart=0.7,
RData.location=RData.location)

# c) track plot
library(Gviz)
```

```
#pdf(file="Gene_tracks.pdf", height=4, width=10, compress=TRUE)
xPierTrackAdv(pNode, RData.location=RData.location)
#dev.off()
xPierTrackAdv(pNode, priority.top=1000, nearby=20,
RData.location=RData.location)

## End(Not run)
```

`xPredictCompare` *Function to compare prediction performance results*

Description

`xPredictCompare` is supposed to compare prediction performance results. It returns an object of class "ggplot".

Usage

```
xPredictCompare(
list_pPerf,
displayBy = c("ROC", "PR"),
type = c("curve", "bar"),
sort = TRUE,
detail = TRUE,
facet = FALSE,
font.family = "sans",
signature = TRUE
)
```

Arguments

`list_pPerf`	a list of "pPerf" objects
`displayBy`	which performance will be used for comparison. It can be "ROC" for ROC curve (by default), "PR" for PR curve
`type`	the type of plot to draw. It can be "curve" for curve plot (by default), "bar" for bar plot
`sort`	logical to indicate whether to sort methods according to performance. By default, it sets TRUE
`detail`	logical to indicate whether to label methods along with performance. By default, it sets TRUE
`facet`	logical to indicate whether to facet/wrap a 1d of panels into 2d. By default, it sets FALSE
`font.family`	the font family for texts
`signature`	a logical to indicate whether the signature is assigned to the plot caption. By default, it sets TRUE showing which function is used to draw this graph

Value

an object of class "ggplot" or NULL (if all input pPerf objects are NULL)

Note

none

See Also

xPredictCompare

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
bp <- xPredictCompare(ls_pPerf, displayBy="ROC")
print(bp)
## modify legend position
bp + theme(legend.position=c(0.75,0.25))

## End(Not run)
```

---

xPredictROCR *Function to assess the prediction performance via ROC and Precision-Recall (PR) analysis*

Description

xPredictROCR is supposed to assess the prediction performance via Receiver Operating Characteristic (ROC) and Precision-Recall (PR) analysis. It requires three inputs: 1) Gold Standard Positive (GSP) targets; 2) Gold Standard Negative (GSN) targets; 3) prediction containing predicted targets and predictive scores.

Usage

```
xPredictROCR(
prediction,
GSP,
GSN,
rescale = TRUE,
plot = c("none", "ROC", "PR"),
verbose = TRUE,
font.family = "sans",
signature = TRUE
)
```

Arguments

prediction	a data frame containing predictions along with predictive scores. It has two columns: 1st column for target, 2nd column for predictive scores (the higher the better). Alternatively, it can be an object of class "pNode" (or "sTarget" or "dTarget") from which a data frame is extracted
GSP	a vector containing Gold Standard Positives (GSP)
GSN	a vector containing Gold Standard Negatives (GSN)
rescale	logical to indicate whether to linearly rescale predictive scores for GSP/GSN targets to the range [0,1]. By default, it sets to TRUE
plot	the way to plot performance curve. It can be 'none' for no curve returned, 'ROC' for ROC curve, and 'PR' for PR curve.
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to TRUE for display
font.family	the font family for texts
signature	a logical to indicate whether the signature is assigned to the plot caption. By default, it sets TRUE showing which function is used to draw this graph

Value

If plot is 'none' (by default), an object of class "pPerf", a list with following components:

- PRS: a data frame with 3 columns ('Precision', 'Recall' and 'Specificity')
- AUROC: a scalar value for ROC AUC
- Fmax: a scalar value for maximum F-measure
- ROC_perf: a ROCR performance-class object for ROC curve
- PR_perf: a ROCR performance-class object for PR curve
- Pred_obj: a ROCR prediction-class object (potentially used for calculating other performance measures)

If plot is 'ROC' or 'PR', it will return a ggplot object after being appended with the same components as mentioned above. If no GSP and/or GSN is predicted, it will return NULL

Note

AUC: the area under ROC F-measure: the maximum of a harmonic mean between precision and recall along PR curve

See Also

xPredictROCR

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
pPerf <- xPredictROCR(prediction, GSP, GSN)

## End(Not run)
```

xRDataLoader *Function to load the package built-in RData*

Description

xRDataLoader is supposed to load the package built-in RData.

Usage

```
xRDataLoader(
RData = c(NA, "GWAS2EF", "GWAS_LD", "IlluminaHumanHT",
"IlluminaOmniExpress",
"ig.DO", "ig.EF", "ig.GOBP", "ig.GOCC", "ig.GOMF", "ig.HPCM",
"ig.HPMA", "ig.HPMI",
"ig.HPPA", "ig.MP", "org.Hs.eg", "org.Hs.egDGIdb", "org.Hs.egDO",
"org.Hs.egGOBP",
"org.Hs.egGOCC", "org.Hs.egGOMF", "org.Hs.egHPCM", "org.Hs.egHPMA",
"org.Hs.egHPMI",
"org.Hs.egHPPA", "org.Hs.egMP", "org.Hs.egMsigdbC1",
"org.Hs.egMsigdbC2BIOCARTA",
"org.Hs.egMsigdbC2CGP", "org.Hs.egMsigdbC2CPall",
"org.Hs.egMsigdbC2CP",
"org.Hs.egMsigdbC2KEGG", "org.Hs.egMsigdbC2REACTOME",
"org.Hs.egMsigdbC3MIR",
"org.Hs.egMsigdbC3TFT", "org.Hs.egMsigdbC4CGN", "org.Hs.egMsigdbC4CM",
"org.Hs.egMsigdbC5BP", "org.Hs.egMsigdbC5CC", "org.Hs.egMsigdbC5MF",
"org.Hs.egMsigdbC6", "org.Hs.egMsigdbC7", "org.Hs.egMsigdbH",
"org.Hs.egPS",
"org.Hs.egSF", "org.Hs.egPfam", "org.Hs.string", "org.Hs.PCommons_DN",
"org.Hs.PCommons_UN"),
RData.customised = NULL,
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

RData	which built-in RData to load. It can be one of "GWAS2EF", "GWAS_LD", "IlluminaHumanHT", "IlluminaOmniExpress", "ig.DO", "ig.EF", "ig.GOBP", "ig.GOCC", "ig.GOMF", "ig.HPCM", "ig.HPMA", "ig.HPMI", "ig.HPPA", "ig.MP", "org.Hs.eg", "org.Hs.egDGIdb", "org.Hs.egDO", "org.Hs.egGOBP", "org.Hs.egGOCC", "org.Hs.egGOMF", "org.Hs.egHPCM", "org.Hs.egHPMA", "org.Hs.egHPMI", "org.Hs.egHPPA", "org.Hs.egMP", "org.Hs.egMsigdbC1", "org.Hs.egMsigdbC2BIOCARTA", "org.Hs.egMsigdbC2CGP", "org.Hs.egMsigdbC2CPall", "org.Hs.egMsigdbC2CP", "org.Hs.egMsigdbC2KEGG", "org.Hs.egMsigdbC2REACTOME", "org.Hs.egMsigdbC3MIR", "org.Hs.egMsigdbC3TFT", "org.Hs.egMsigdbC4CGN", "org.Hs.egMsigdbC4CM",

"org.Hs.egMsigdbC5BP", "org.Hs.egMsigdbC5CC", "org.Hs.egMsigdbC5MF", "org.Hs.egMsigdbC6", "org.Hs.egMsigdbC7", "org.Hs.egMsigdbH", "org.Hs.egPS", "org.Hs.egSF", "org.Hs.egPfam", "org.Hs.string", "org.Hs.PCommons_DN", "org.Hs.PCommons_UN", "org.Hs.egGTExV4", "org.Hs.egGTExV6"

`RData.customised` a file name for RData-formatted file. By default, it is NULL. It is designed when the user wants to import customised RData that are not listed in the above argument 'RData'. However, this argument can be always used even for those RData that are listed in the argument 'RData'

`verbose` logical to indicate whether the messages will be displayed in the screen. By default, it sets to TRUE for display

`RData.location` the characters to tell the location of built-in RData files. By default, it remotely locates at http://galahad.well.ox.ac.uk/bigdata; the development version locates at http://galahad.well.ox.ac.uk/bigdata. For the user equipped with fast internet connection, this option can be just left as default. But it is always advisable to download these files locally. Especially when the user needs to run this function many times, there is no need to ask the function to remotely download every time (also it will unnecessarily increase the runtime). For examples, these files (as a whole or part of them) can be first downloaded into your current working directory, and then set this option as: $RData.location = "."$. Surely, the location can be anywhere as long as the user provides the correct path pointing to (otherwise, the script will have to remotely download each time)

`guid` a valid (5-character) Global Unique IDentifier for an OSF project. For example, 'gskpn' (see 'https://osf.io/gskpn'). If a valid provided and the query matched, it has priority over the one specified via RData.location

Value

any use-specified variable that is given on the right side of the assigement sign '<-', which contains the loaded RData. If the data cannot be loaded, it returns NULL.

Note

If there are no use-specified variable that is given on the right side of the assigement sign '<-', then no RData will be loaded onto the working environment. To enable 'guid', please also install a package "osfr" via `BiocManager::install("osfr",dependencies=TRUE)`.

See Also

xRDataLoader

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase')
org.Hs.eg <- xRDataLoader(RData='org.Hs.eg')
ig.HPPA <- xRDataLoader(RData='ig.HPPA')
org.Hs.egHPPA <- xRDataLoader(RData='org.Hs.egHPPA')
```

```
org.Hs.egHPPA <- xRDataLoader(RData.customised='org.Hs.egHPPA')
org.Hs.egHPPA <- xRDataLoader(RData.customised='org.Hs.egHPPA')

# from OSF
org.Mm.egKEGG <- xRDataLoader(RData='org.Mm.egKEGG', guid='gskpn')
org.Mm.string_high <- xRDataLoader(RData='org.Mm.string_high',
guid='gskpn')

## End(Not run)
```

---

xRWR *Function to implement Random Walk with Restart (RWR) on the input graph*

Description

xRWR is supposed to implement Random Walk with Restart (RWR) on the input graph. If the seeds (i.e. a set of starting nodes) are given, it intends to calculate the affinity score of all nodes in the graph to the seeds. If the seeds are not given, it will pre-compute affinity matrix for nodes in the input graph with respect to each starting node (as a seed) by looping over every node in the graph. Parallel computing is also supported.

Usage

```
xRWR(
g,
normalise = c("laplacian", "row", "column", "none"),
setSeeds = NULL,
restart = 0.75,
normalise.affinity.matrix = c("none", "quantile"),
parallel = TRUE,
multicores = NULL,
verbose = TRUE
)
```

Arguments

g	an object of class "igraph" or "graphNEL"
normalise	the way to normalise the adjacency matrix of the input graph. It can be 'laplacian' for laplacian normalisation, 'row' for row-wise normalisation, 'column' for column-wise normalisation, or 'none'
setSeeds	an input matrix used to define sets of starting seeds. One column corresponds to one set of seeds that a walker starts with. The input matrix must have row names, coming from node names of input graph, i.e. V(g)$name, since there is a mapping operation. The non-zero entries mean that the corresonding rows (i.e. the gene/row names) are used as the seeds, and non-zero values can be viewed as how to weight the relative importance of seeds. By default, this option sets

to "NULL", suggesting each node in the graph will be used as a set of the seed to pre-compute affinity matrix for the input graph. This default does not scale for large input graphs since it will loop over every node in the graph; however, the pre-computed affinity matrix can be extensively reused for obtaining affinity scores between any combinations of nodes/seeds, allows for some flexibility in the downstream use, in particular when sampling a large number of random node combinations for statistical testing

`restart` the restart probability used for RWR. The restart probability takes the value from 0 to 1, controlling the range from the starting nodes/seeds that the walker will explore. The higher the value, the more likely the walker is to visit the nodes centered on the starting nodes. At the extreme when the restart probability is zero, the walker moves freely to the neighbors at each step without restarting from seeds, i.e., following a random walk (RW)

`normalise.affinity.matrix`
the way to normalise the output affinity matrix. It can be 'none' for no normalisation, 'quantile' for quantile normalisation to ensure that columns (if multiple) of the output affinity matrix have the same quantiles

`parallel` logical to indicate whether parallel computation with multicores is used. By default, it sets to true, but not necessarily does so. It will depend on whether these two packages "foreach" and "doParallel" have been installed

`multicores` an integer to specify how many cores will be registered as the multicore parallel backend to the 'foreach' package. If NULL, it will use a half of cores available in a user's computer. This option only works when parallel computation is enabled

`verbose` logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

It returns a sparse matrix, called 'PTmatrix':

- When the seeds are NOT given: a pre-computated affinity matrix with the dimension of n X n, where n is the number of nodes in the input graph. Columns stand for starting nodes walking from, and rows for ending nodes walking to. Therefore, a column for a starting node represents a steady-state affinity vector that the starting node will visit all the ending nodes in the graph
- When the seeds are given: an affinity matrix with the dimension of n X nset, where n is the number of nodes in the input graph, and nset for the number of the sets of seeds (i.e. the number of columns in setSeeds). Each column stands for the steady probability vector, storing the affinity score of all nodes in the graph to the starting nodes/seeds. This steady probability vector can be viewed as the "influential impact" over the graph imposed by the starting nodes/seeds.

Note

The input graph will treat as an unweighted graph if there is no 'weight' edge attribute associated with

See Also

xCheckParallel

Examples

```
# 1) generate a random graph according to the ER model
set.seed(123)
g <- erdos.renyi.game(10, 1/10)

## Not run:
# 2) produce the induced subgraph only based on the nodes in query
subg <- dNetInduce(g, V(g), knn=0)
V(subg)$name <- 1:vcount(subg)

# 3) obtain the pre-computated affinity matrix
PTmatrix <- xRWR(g=subg, normalise="laplacian", restart=0.75,
parallel=FALSE)
# visualise affinity matrix
visHeatmapAdv(as.matrix(PTmatrix), Rowv=FALSE, Colv=FALSE,
colormap="wyr", KeyValueName="Affinity")

# 4) obtain affinity matrix given sets of seeds
# define sets of seeds
# each seed with equal weight (i.e. all non-zero entries are '1')
aSeeds <- c(1,0,1,0,1)
bSeeds <- c(0,0,1,0,1)
setSeeds <- data.frame(aSeeds,bSeeds)
rownames(setSeeds) <- 1:5
# calcualte affinity matrix
PTmatrix <- xRWR(g=subg, normalise="laplacian", setSeeds=setSeeds,
restart=0.75, parallel=FALSE)
PTmatrix

## End(Not run)
```

---

xSM2DF *Function to create a data frame (with three columns) from a (sparse) matrix*

Description

xSM2DF is supposed to create a data frame (with three columns) from a (sparse) matrix. Only nonzero/nonna entries from the matrix will be kept in the resulting data frame.

Usage

```
xSM2DF(data, verbose = TRUE)
```

Arguments

data	a matrix or an object of the dgCMatrix class (a sparse matrix)
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to TRUE for display

Value

a data frame containing three columns: 1st column for row names, 2nd for column names, and 3rd for numeric values

Note

none None

See Also

xSM2DF

Examples

```
# create a sparse matrix of 4 X 2
input.file <- rbind(c('R1','C1',1), c('R2','C1',1), c('R2','C2',1),
c('R3','C2',2), c('R4','C1',1))
data <- xSparseMatrix(input.file)
## Not run:
# convert into a data frame
res_df <- xSM2DF(data)
res_df

## End(Not run)
```

---

xSNP2cGenes *Function to define HiC genes given a list of SNPs*

Description

xSNP2cGenes is supposed to define HiC genes given a list of SNPs. The HiC weight is calcualted as Cumulative Distribution Function of HiC interaction scores.

Usage

```
xSNP2cGenes(
data,
entity = c("SNP", "chr:start-end", "data.frame", "bed", "GRanges"),
include.HiC = NA,
GR.SNP = c("dbSNP_GWAS", "dbSNP_Common"),
cdf.function = c("empirical", "exponential"),
plot = FALSE,
```

```
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

data	an input vector containing SNPs. SNPs should be provided as dbSNP ID (ie starting with rs) or in the format of 'chrN:xxx', where N is either 1-22 or X, xxx is number; for example, 'chr16:28525386'. Alternatively, it can be other formats/entities (see the next parameter 'entity')
entity	the data entity. By default, it is "SNP". For general use, it can also be one of "chr:start-end", "data.frame", "bed" or "GRanges"
include.HiC	genes linked to input SNPs are also included. By default, it is 'NA' to disable this option. Otherwise, those genes linked to SNPs will be included according to Promoter Capture HiC (PCHiC) datasets. Pre-built HiC datasets are detailed in xDefineHIC
GR.SNP	the genomic regions of SNPs. By default, it is 'dbSNP_GWAS', that is, SNPs from dbSNP (version 146) restricted to GWAS SNPs and their LD SNPs (hg19). It can be 'dbSNP_Common', that is, Common SNPs from dbSNP (version 146) plus GWAS SNPs and their LD SNPs (hg19). Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to dbSNP IDs. Then, tell "GR.SNP" with your RData file name (with or without extension), plus specify your file RData path in "RData.location". Note: you can also load your customised GR object directly
cdf.function	a character specifying a Cumulative Distribution Function (cdf). It can be one of 'exponential' based on exponential cdf, 'empirical' for empirical cdf
plot	logical to indicate whether the histogram plot (plus density or CDF plot) should be drawn. By default, it sets to false for no plotting
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details
guid	a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details

Value

a data frame with following columns:

- Gene: SNP-interacting genes caputured by HiC
- SNP: SNPs
- Sig: the interaction score (the higher stronger)
- Weight: the HiC weight

Note

none

See Also

xDefineHIC, xSymbol2GeneID

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# a) provide the SNPs with the significance info
data(ImmunoBase)
data <- names(ImmunoBase$AS$variants)

# b) define HiC genes
df_cGenes <- xSNP2cGenes(data, include.HiC="Monocytes",
RData.location=RData.location)

## End(Not run)
```

---

xSNP2eGenes *Function to define eQTL genes given a list of SNPs or a customised eQTL mapping data*

Description

xSNP2eGenes is supposed to define eQTL genes given a list of SNPs or a customised eQTL mapping data. The eQTL weight is calcualted as Cumulative Distribution Function of negative log-transformed eQTL-reported signficance level.

Usage

```
xSNP2eGenes(
data,
include.eQTL = NA,
eQTL.customised = NULL,
cdf.function = c("empirical", "exponential"),
plot = FALSE,
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

data	an input vector containing SNPs. SNPs should be provided as dbSNP ID (ie starting with rs). Alternatively, they can be in the format of 'chrN:xxx', where N is either 1-22 or X, xxx is number; for example, 'chr16:28525386'
include.eQTL	the eQTL supported currently. By default, it is 'NA' to disable this option. Pre-built eQTL datasets are detailed in xDefineEQTL
eQTL.customised	a user-input matrix or data frame with 4 columns: 1st column for SNPs/eQTLs, 2nd column for Genes, 3rd for eQTL mapping significance level (p-values or FDR), and 4th for contexts (required even though only one context is input). Alternatively, it can be a file containing these 4 columns. It is designed to allow the user analysing their eQTL data. This customisation (if provided) will populate built-in eQTL data
cdf.function	a character specifying a Cumulative Distribution Function (cdf). It can be one of 'exponential' based on exponential cdf, 'empirical' for empirical cdf
plot	logical to indicate whether the histogram plot (plus density or CDF plot) should be drawn. By default, it sets to false for no plotting
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details
guid	a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details

Value

a data frame with following columns:

- Gene: eQTL-containing genes
- SNP: eQTLs
- Sig: the eQTL mapping significant level (the best/minimum)
- Weight: the eQTL weight

Note

none

See Also

xDefineEQTL, xSymbol2GeneID

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# a) provide the SNPs with the significance info
```

```
data(ImmunoBase)
gr <- ImmunoBase$AS$variants
AS <- as.data.frame(GenomicRanges::mcols(gr)[, c('Variant','Pvalue')])

# b) define eQTL genes
df_eGenes <- xSNP2eGenes(data=AS[,1], include.eQTL="JKscience_TS2A",
RData.location=RData.location)

## End(Not run)
```

---

xSNP2nGenes *Function to define nearby genes given a list of SNPs*

Description

xSNP2nGenes is supposed to define nearby genes given a list of SNPs within certain distance window. The distance weight is calcualted as a decaying function of the gene-to-SNP distance.

Usage

```
xSNP2nGenes(
data,
distance.max = 2e+05,
decay.kernel = c("rapid", "slow", "linear", "constant"),
decay.exponent = 2,
GR.SNP = c("dbSNP_GWAS", "dbSNP_Common", "dbSNP_Single"),
GR.Gene = c("UCSC_knownGene", "UCSC_knownCanonical"),
include.TAD = c("none", "GM12878", "IMR90", "MSC", "TRO", "H1", "MES",
"NPC"),
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

data	an input vector containing SNPs. SNPs should be provided as dbSNP ID (ie starting with rs). Alternatively, they can be in the format of 'chrN:xxx', where N is either 1-22 or X, xxx is genomic positional number; for example, 'chr16:28525386'
distance.max	the maximum distance between genes and SNPs. Only those genes no far way from this distance will be considered as seed genes. This parameter will influence the distance-component weights calculated for nearby SNPs per gene
decay.kernel	a character specifying a decay kernel function. It can be one of 'slow' for slow decay, 'linear' for linear decay, and 'rapid' for rapid decay. If no distance weight is used, please select 'constant'
decay.exponent	a numeric specifying a decay exponent. By default, it sets to 2

GR.SNP — the genomic regions of SNPs. By default, it is 'dbSNP_GWAS', that is, SNPs from dbSNP (version 146) restricted to GWAS SNPs and their LD SNPs (hg19). It can be 'dbSNP_Common', that is, Common SNPs from dbSNP (version 146) plus GWAS SNPs and their LD SNPs (hg19). Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to dbSNP IDs. Then, tell "GR.SNP" with your RData file name (with or without extension), plus specify your file RData path in "RData.location". Note: you can also load your customised GR object directly

GR.Gene — the genomic regions of genes. By default, it is 'UCSC_knownGene', that is, UCSC known genes (together with genomic locations) based on human genome assembly hg19. It can be 'UCSC_knownCanonical', that is, UCSC known canonical genes (together with genomic locations) based on human genome assembly hg19. Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to Gene Symbols. Then, tell "GR.Gene" with your RData file name (with or without extension), plus specify your file RData path in "RData.location". Note: you can also load your customised GR object directly

include.TAD — TAD boundary regions are also included. By default, it is 'none' to disable this option. Otherwise, inclusion of a TAD dataset to pre-filter SNP-nGene pairs (i.e. only those within a TAD region will be kept). TAD datasets can be one of "GM12878" (lymphoblast), "IMR90" (fibroblast), "MSC" (mesenchymal stem cell) ,"TRO" (trophoblasts-like cell), "H1" (embryonic stem cell), "MES" (mesendoderm) and "NPC" (neural progenitor cell). Explanations can be found at http://dx.doi.org/10.1016/j.celrep.2016.10.061

verbose — logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

RData.location — the characters to tell the location of built-in RData files. See xRDataLoader for details

guid — a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details

Value

a data frame with following columns:

- Gene: nearby genes
- SNP: SNPs
- Dist: the genomic distance between the gene and the SNP
- Weight: the distance weight based on the genomic distance
- Gap: the genomic gap between the gene and the SNP (in the form of 'chr:start-end')
- TAD: if applied, it can be 'Excluded' or the TAD boundary region (in the form of 'chr:start-end') that the genomic interval falls into. Also if SNP within the gene body, Gap and TAD will be SNP location (in the form of 'chr:start-end')

Note

For details on the decay kernels, please refer to xVisKernels

See Also

xSNPlocations, xRDataLoader, xGR, xSymbol2GeneID

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:

# a) provide the seed SNPs with the significance info
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
## get lead SNPs reported in AS GWAS and their significance info (p-values)
gr <- ImmunoBase$AS$variant
data <- names(gr)

# b) define nearby genes
df_nGenes <- xSNP2nGenes(data=data, distance.max=200000,
decay.kernel="slow", decay.exponent=2, RData.location=RData.location)

# c) define nearby genes (considering TAD boundary regions in GM12878)
df_nGenes <- xSNP2nGenes(data=data, distance.max=200000,
decay.kernel="slow", decay.exponent=2, include.TAD='GM12878',
RData.location=RData.location)

## End(Not run)
```

---

xSNPlocations *Function to extract genomic locations given a list of SNPs*

Description

xSNPlocations is supposed to extract genomic locations given a list of SNPs.

Usage

```
xSNPlocations(
data,
GR.SNP = c("dbSNP_GWAS", "dbSNP_Common", "dbSNP_Single"),
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

data	a input vector containing SNPs. SNPs should be provided as dbSNP ID (ie starting with rs). Alternatively, they can be in the format of 'chrN:xxx', where N is either 1-22 or X, xxx is genomic positional number; for example, 'chr16:28525386'
GR.SNP	the genomic regions of SNPs. By default, it is 'dbSNP_GWAS', that is, SNPs from dbSNP (version 146) restricted to GWAS SNPs and their LD SNPs (hg19). It can be 'dbSNP_Common', that is, Common SNPs from dbSNP (version 146) plus GWAS SNPs and their LD SNPs (hg19). Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to dbSNP IDs. Then, tell "GR.SNP" with your RData file name (with or without extension), plus specify your file RData path in "RData.location". Note: you can also load your customised GR object directly
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details
guid	a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details

Value

an GR oject, with an additional metadata column called 'variant_id' storing SNP location in the format of 'chrN:xxx', where N is either 1-22 or X, xxx is genomic positional number.

Note

none

See Also

xRDataLoader

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

## Not run:
# a) provide the SNPs
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
## get lead SNPs reported in AS GWAS and their significance info (p-values)
gr <- ImmunoBase$AS$variant
data <- names(gr)

# b) find the location
snp_gr <- xSNPlocations(data=data, RData.location=RData.location)
```

```
## End(Not run)
```

`xSNPscores` *Function to score lead or LD SNPs based on the given significance level*

Description

`xSNPscores` is supposed to score a list of Lead SNPs together with the significance level. It can consider LD SNPs and the given threshold of the significant level.

Usage

```
xSNPscores(
data,
include.LD = NA,
LD.customised = NULL,
LD.r2 = 0.8,
significance.threshold = 5e-05,
score.cap = 10,
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

`data`	a named input vector containing the significance level for nodes (dbSNP). For this named vector, the element names are dbSNP (starting with rs or in the format of 'chrN:xxx', where N is either 1-22 or X, xxx is number; for example, 'chr16:28525386'), the element values for the significance level (measured as p-value or fdr). Alternatively, it can be a matrix or data frame with two columns: 1st column for dbSNP, 2nd column for the significance level.
`include.LD`	additional SNPs in LD with Lead SNPs are also included. By default, it is 'NA' to disable this option. Otherwise, LD SNPs will be included based on one or more of 26 populations and 5 super populations from 1000 Genomics Project data (phase 3). The population can be one of 5 super populations ("AFR", "AMR", "EAS", "EUR", "SAS"), or one of 26 populations ("ACB", "ASW", "BEB", "CDX", "CEU", "CHB", "CHS", "CLM", "ESN", "FIN", "GBR", "GIH", "GWD", "IBS", "ITU", "JPT", "KHV", "LWK", "MSL", "MXL", "PEL", "PJL", "PUR", "STU", "TSI", "YRI"). Explanations for population code can be found at http://www.1000genomes.org/faq/which-populations-are-part-your-study
`LD.customised`	a user-input matrix or data frame with 3 columns: 1st column for Lead SNPs, 2nd column for LD SNPs, and 3rd for LD r2 value. It is designed to allow the user analysing their precalcuated LD info. This customisation (if provided) has the high priority over built-in LD SNPs

LD.r2	the LD r2 value. By default, it is 0.8, meaning that SNPs in LD (r2>=0.8) with input SNPs will be considered as LD SNPs. It can be any value from 0.8 to 1
significance.threshold	the given significance threshold. By default, it is set to NULL, meaning there is no constraint on the significance level when transforming the significance level of SNPs into scores. If given, those SNPs below this are considered significant and thus scored positively. Instead, those above this are considered insigificant and thus receive no score
score.cap	the maximum score being capped. By default, it is set to 10. If NULL, no capping is applied
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details
guid	a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details

Value

a data frame with following columns:

- SNP: Lead and/or LD SNPs
- Score: the scores for SNPs calculated based on p-values taking into account the given threshold of the significant level
- Pval: the input p-values for Lead SNPs or R2-adjusted p-values for LD SNPs
- Flag: the flag to indicate whether the resulting SNPs are Lead SNPs or LD SNPs

Note

None

See Also

xRDataLoader

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

## Not run:
# a) provide the seed SNPs with the significance info
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
## get lead SNPs reported in AS GWAS and their significance info (p-values)
gr <- ImmunoBase$AS$variant
data <- GenomicRanges::mcols(gr)[,c(1,3)]
```

```
# b) calculate SNP scores (considering significant cutoff 5e-5)
## without inclusion of LD SNPs
df_SNP <- xSNPscores(data=data, significance.threshold=5e-5,
RData.location=RData.location)
## include LD SNPs (calculated based on European populations)
df_SNP <- xSNPscores(data=data, significance.threshold=5e-5,
include.LD="EUR", RData.location=RData.location)

## End(Not run)
```

`xSparseMatrix` *Function to create a sparse matrix for an input file with three columns*

Description

`xSparseMatrix` is supposed to create a sparse matrix for an input file with three columns.

Usage

```
xSparseMatrix(input.file, rows = NULL, columns = NULL, verbose = TRUE)
```

Arguments

`input.file`	an input file containing three columns: 1st column for rows, 2nd for columns, and 3rd for numeric values. Alternatively, the input.file can be a matrix or data frame, assuming that input file has been read. Note: the file should use the tab delimiter as the field separator between columns
`rows`	a vector specifying row names. By default, it is NULL
`columns`	a vector specifying column names. By default, it is NULL
`verbose`	logical to indicate whether the messages will be displayed in the screen. By default, it sets to TRUE for display

Value

an object of the dgCMatrix class (a sparse matrix)

Note

If rows (or columns) are not NULL, the rows (or columns) of resulting sparse matrix will be union of those from input.file and those from rows (or columns). None

See Also

`xSparseMatrix`

Examples

```
# create a sparse matrix of 4 X 2
input.file <- rbind(c('R1','C1',1), c('R2','C1',1), c('R2','C2',1),
c('R3','C2',1), c('R4','C1',1))
res <- xSparseMatrix(input.file)
res
# get a full matrix
as.matrix(res)

res <- xSparseMatrix(input.file, columns=c('C1','C2','C3'))
res
```

---

xSubneterGenes *Function to identify a subnetwork from an input network and the significance level imposed on its nodes*

Description

xSubneterGenes is supposed to identify maximum-scoring subnetwork from an input graph with the node information on the significance (measured as p-values or fdr). It returns an object of class "igraph".

Usage

```
xSubneterGenes(
data,
network = c("STRING_highest", "STRING_high", "STRING_medium",
"STRING_low",
"PCommonsUN_high", "PCommonsUN_medium", "PCommonsDN_high",
"PCommonsDN_medium",
"PCommonsDN_Reactome", "PCommonsDN_KEGG", "PCommonsDN_HumanCyc",
"PCommonsDN_PID",
"PCommonsDN_PANTHER", "PCommonsDN_ReconX", "PCommonsDN_TRANSFAC",
"PCommonsDN_PhosphoSite", "PCommonsDN_CTD", "KEGG", "KEGG_metabolism",
"KEGG_genetic", "KEGG_environmental", "KEGG_cellular",
"KEGG_organismal",
"KEGG_disease", "REACTOME"),
STRING.only = c(NA, "neighborhood_score", "fusion_score",
"cooccurence_score",
"coexpression_score", "experimental_score", "database_score",
"textmining_score")[1],
network.customised = NULL,
seed.genes = TRUE,
subnet.significance = 0.01,
subnet.size = NULL,
test.permutation = FALSE,
num.permutation = 100,
```

```
respect = c("none", "degree"),
aggregateBy = c("Ztransform", "fishers", "logistic", "orderStatistic"),
verbose = TRUE,
silent = FALSE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

data	a named input vector containing the significance level for nodes (gene symbols). For this named vector, the element names are gene symbols, the element values for the significance level (measured as p-value or fdr). Alternatively, it can be a matrix or data frame with two columns: 1st column for gene symbols, 2nd column for the significance level
network	the built-in network. Currently two sources of network information are supported: the STRING database (version 10) and the Pathway Commons database (version 7). STRING is a meta-integration of undirect interactions from the functional aspect, while Pathways Commons mainly contains both undirect and direct interactions from the physical/pathway aspect. Both have scores to control the confidence of interactions. Therefore, the user can choose the different quality of the interactions. In STRING, "STRING_highest" indicates interactions with highest confidence (confidence scores>=900), "STRING_high" for interactions with high confidence (confidence scores>=700), "STRING_medium" for interactions with medium confidence (confidence scores>=400), and "STRING_low" for interactions with low confidence (confidence scores>=150). For undirect/physical interactions from Pathways Commons, "PCommonsUN_high" indicates undirect interactions with high confidence (supported with the PubMed references plus at least 2 different sources), "PCommonsUN_medium" for undirect interactions with medium confidence (supported with the PubMed references). For direct (pathway-merged) interactions from Pathways Commons, "PCommonsDN_high" indicates direct interactions with high confidence (supported with the PubMed references plus at least 2 different sources), and "PCommonsUN_medium" for direct interactions with medium confidence (supported with the PubMed references). In addition to pooled version of pathways from all data sources, the user can also choose the pathway-merged network from individual sources, that is, "PCommonsDN_Reactome" for those from Reactome, "PCommonsDN_KEGG" for those from KEGG, "PCommonsDN_HumanCyc" for those from HumanCyc, "PCommonsDN_PID" for those froom PID, "PCommonsDN_PANTHER" for those from PANTHER, "PCommonsDN_ReconX" for those from ReconX, "PCommonsDN_TRANSFAC" for those from TRANSFAC, "PCommonsDN_PhosphoSite" for those from PhosphoSite, and "PCommonsDN_CTD" for those from CTD. For direct (pathway-merged) interactions sourced from KEGG, it can be 'KEGG' for all, 'KEGG_metabolism' for pathways grouped into 'Metabolism', 'KEGG_genetic' for 'Genetic Information Processing' pathways, 'KEGG_environmental' for 'Environmental Information Processing' pathways, 'KEGG_cellular' for 'Cellular Processes' pathways, 'KEGG_organismal' for 'Organismal Systems' pathways, and 'KEGG_disease' for 'Human Dis-

eases' pathways. 'REACTOME' for protein-protein interactions derived from Reactome pathways

`STRING.only` the further restriction of STRING by interaction type. If NA, no such restriction. Otherwide, it can be one or more of "neighborhood_score","fusion_score","cooccurence_score","coexpres Useful options are c("experimental_score","database_score"): only experimental data (extracted from BIND, DIP, GRID, HPRD, IntAct, MINT, and PID) and curated data (extracted from Biocarta, BioCyc, GO, KEGG, and Reactome) are used

`network.customised`
an object of class "igraph". By default, it is NULL. It is designed to allow the user analysing their customised network data that are not listed in the above argument 'network'. This customisation (if provided) has the high priority over built-in network

`seed.genes` logical to indicate whether the identified network is restricted to seed genes (ie input genes with the signficant level). By default, it sets to true

`subnet.significance`
the given significance threshold. By default, it is set to NULL, meaning there is no constraint on nodes/genes. If given, those nodes/genes with p-values below this are considered significant and thus scored positively. Instead, those p-values above this given significance threshold are considered insigificant and thus scored negatively

`subnet.size` the desired number of nodes constrained to the resulting subnet. It is not nulll, a wide range of significance thresholds will be scanned to find the optimal significance threshold leading to the desired number of nodes in the resulting subnet. Notably, the given significance threshold will be overwritten by this option

`test.permutation`
logical to indicate whether the permutation test is perform to estimate the significance of identified network with the same number of nodes. By default, it sets to false

`num.permutation`
the number of permutations generating the null distribution of the identified network

`respect` how to respect nodes to be sampled. It can be one of 'none' (randomly sampling) and 'degree' (degree-preserving sampling)

`aggregateBy` the aggregate method used to aggregate edge confidence p-values. It can be either "orderStatistic" for the method based on the order statistics of p-values, or "fishers" for Fisher's method, "Ztransform" for Z-transform method, "logistic" for the logistic method. Without loss of generality, the Z-transform method does well in problems where evidence against the combined null is spread widely (equal footings) or when the total evidence is weak; Fisher's method does best in problems where the evidence is concentrated in a relatively small fraction of the individual tests or when the evidence is at least moderately strong; the logistic method provides a compromise between these two. Notably, the aggregate methods 'Ztransform' and 'logistic' are preferred here

`verbose` logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

silent	logical to indicate whether the messages will be silent completely. By default, it sets to false. If true, verbose will be forced to be false
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details
guid	a valid (5-character) Global Unique IDentifier for an OSF project. See xRDataLoader for details

Value

a subgraph with a maximum score, an object of class "igraph". It has node attributes (significance, score, type) and a graph attribute (threshold; determined when scanning 'subnet.size'). If permutation test is enabled, it also has a graph attribute (combinedP) and an edge attribute (edgeConfidence).

Note

The algorithm identifying a subnetwork is implemented in the dnet package (http://genomemedicine.biomedcentral.com/artic
014-0064-8). In brief, from an input network with input node/gene information (the significant level; p-values or FDR), the way of searching for a maximum-scoring subnetwork is done as follows. Given the threshold of tolerable p-value, it gives positive scores for nodes with p-values below the threshold (nodes of interest), and negative scores for nodes with threshold-above p-values (intolerable). After score transformation, the search for a maximum scoring subnetwork is deduced to find the connected subnetwork that is enriched with positive-score nodes, allowing for a few negative-score nodes as linkers. This objective is met through minimum spanning tree finding and post-processing, previously used as a heuristic solver of prize-collecting Steiner tree problem. The solver is deterministic, only determined by the given tolerable p-value threshold. For identification of the subnetwork with a desired number of nodes, an iterative procedure is also developed to fine-tune tolerable thresholds. This explicit control over the node size may be necessary for guiding follow-up experiments.

See Also

xDefineNet

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:

# a) provide the input nodes/genes with the significance info
## load human genes
org.Hs.eg <- xRDataLoader(RData='org.Hs.eg',
RData.location=RData.location)
sig <- rbeta(500, shape1=0.5, shape2=1)
data <- data.frame(symbols=org.Hs.eg$gene_info$Symbol[1:500], sig)

# b) perform network analysis
# b1) find maximum-scoring subnet based on the given significance threshold
subnet <- xSubneterGenes(data=data, network="STRING_high",
subnet.significance=0.01, RData.location=RData.location)
# b2) find maximum-scoring subnet with the desired node number=50
```

```
subnet <- xSubneterGenes(data=data, network="STRING_high",
subnet.size=50, RData.location=RData.location)

# c) save subnet results to the files called 'subnet_edges.txt' and 'subnet_nodes.txt'
output <- igraph::get.data.frame(subnet, what="edges")
utils::write.table(output, file="subnet_edges.txt", sep="\t",
row.names=FALSE)
output <- igraph::get.data.frame(subnet, what="vertices")
utils::write.table(output, file="subnet_nodes.txt", sep="\t",
row.names=FALSE)

# d) visualise the identified subnet
## do visualisation with nodes colored according to the significance (you provide)
xVisNet(g=subnet, pattern=-log10(as.numeric(V(subnet)$significance)),
vertex.shape="sphere", colormap="wyr")
## do visualisation with nodes colored according to transformed scores
xVisNet(g=subnet, pattern=as.numeric(V(subnet)$score),
vertex.shape="sphere")

# e) visualise the identified subnet as a circos plot
library(RCircos)
xCircos(g=subnet, entity="Gene", colormap="white-gray",
RData.location=RData.location)

# g) visualise the subnet using the same layout_with_kk
df_tmp <- df[match(V(subnet)$name,df$Symbol),]
vec_tmp <- colnames(df_tmp)
names(vec_tmp) <- vec_tmp
glayout <- igraph::layout_with_kk(subnet)
V(subnet)$xcoord <- glayout[,1]
V(subnet)$xcoord <- glayout[,2]
# g1) colored according to FDR
ls_ig <- lapply(vec_tmp, function(x){
ig <- subnet
V(ig)$fdr <- -log10(as.numeric(df_tmp[,x]))
ig
})
gp_FDR <- xA2Net(g=ls_g, node.label='name', node.label.size=2,
node.label.color='blue', node.label.alpha=0.8, node.label.padding=0.25,
node.label.arrow=0, node.label.force=0.1, node.shape=19,
node.xcoord='xcoord', node.ycoord='ycoord', node.color='fdr',
node.color.title=expression(-log[10]('FDR')),
colormap='grey-yellow-orange', ncolors=64, zlim=c(0,3),
node.size.range=4,
edge.color="black",edge.color.alpha=0.3,edge.curve=0.1,edge.arrow.gap=0.025)
# g2) colored according to FC
ls_ig <- lapply(vec_tmp, function(x){
ig <- subnet
V(ig)$lfc <- as.numeric(df_tmp[,x])
ig
})
gp_FC <- xA2Net(g=ls_g, node.label='name', node.label.size=2,
node.label.color='blue', node.label.alpha=0.8, node.label.padding=0.25,
```

```
node.label.arrow=0, node.label.force=0.1, node.shape=19,
node.xcoord='xcoord', node.ycoord='ycoord', node.color='lfc',
node.color.title=expression(log[2]('FC')), colormap='cyan1-grey-pink1',
ncolors=64, zlim=c(-3,3), node.size.range=4,
edge.color="black",edge.color.alpha=0.3,edge.curve=0.1,edge.arrow.gap=0.025)
# g3) colored according to FC
gridExtra::grid.arrange(grobs=list(gp_FDR, gp_FC), ncol=2,
as.table=TRUE)

## End(Not run)
```

`xSymbol2GeneID` *Function to convert gene symbols to entrez geneid*

Description

`xSymbol2GeneID` is supposed to convert gene symbols to entrez geneid.

Usage

```
xSymbol2GeneID(
data,
org = c("human", "mouse"),
check.symbol.identity = FALSE,
details = FALSE,
verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata",
guid = NULL
)
```

Arguments

`data`	an input vector containing gene symbols
`org`	a character specifying an organism. Currently supported organisms are 'human' and 'mouse'. It can be an object 'EG'
`check.symbol.identity`	logical to indicate whether to match the input data via Synonyms for those unmatchable by official gene symbols. By default, it sets to false
`details`	logical to indicate whether to result in a data frame (in great details). By default, it sets to false
`verbose`	logical to indicate whether the messages will be displayed in the screen. By default, it sets to false for no display
`RData.location`	the characters to tell the location of built-in RData files. See `xRDataLoader` for details
`guid`	a valid (5-character) Global Unique IDentifier for an OSF project. See `xRDataLoader` for details

Value

a vector containing entrez geneid with 'NA' for the unmatched if (details set to false); otherwise, a data frame is returned

Note

If a symbol mapped many times, the one assiged as the "protein-coding" type of gene is preferred.

See Also

xRDataLoader

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:

# a) provide the input Genes of interest (eg 100 randomly chosen human genes)
## load human genes
org.Hs.eg <- xRDataLoader(RData='org.Hs.eg')
Symbol <- as.character(sample(org.Hs.eg$gene_info$Symbol, 100))
Symbol

# b) convert into GeneID
GeneID <- xSymbol2GeneID(Symbol)

# c) convert into a data frame
df <- xSymbol2GeneID(Symbol, details=TRUE)


# advanced use
df <- xSymbol2GeneID(Symbol, org=org.Hs.eg, details=TRUE)

## End(Not run)
```

---

xVisEvidence *Function to visualise evidence for prioritised genes in a gene network*

Description

xVisEvidence is supposed to visualise evidence for prioritised genes in a gene network. It returns an object of class "igraph".

Usage

```
xVisEvidence(
xTarget,
g = NA,
nodes = NULL,
```

```
node.info = c("smart", "none"),
neighbor.order = 1,
neighbor.seed = TRUE,
neighbor.top = NULL,
largest.comp = TRUE,
show = TRUE,
colormap = "ggplot2",
legend.position = "topleft",
legend.horiz = FALSE,
mtext.side = 3,
verbose = TRUE,
edge.width = NULL,
vertex.size = NULL,
vertex.size.nonseed = NULL,
vertex.label.color = "blue",
vertex.label.color.nonseed = NULL,
...
)
```

Arguments

xTarget	an object of class "dTarget", "sTarget" or "eTarget"
g	an object of class "igraph". If NA, the 'metag' will be used, which is part of the input object "xTarget". If provided, it must have a node attribute called 'priority'
nodes	which node genes are in query. If NULL, the top gene will be queried
node.info	tells the additional information used to label nodes. It can be one of "none" (only gene labeling), "smart" for (by default) using three pieces of information (if any): genes, 5-star ratings, and associated ranks (marked by an @ icon)
neighbor.order	an integer giving the order of the neighborhood. By default, it is 1-order neighborhood
neighbor.seed	logical to indicate whether neighbors are seeds only. By default, it sets to true
neighbor.top	the top number of the neighbors with the highest priority. By default, it sets to NULL to disable this parameter
largest.comp	logical to indicate whether the largest component is only retained. By default, it sets to true for the largest component being left
show	logical to indicate whether to show the graph
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta), and "ggplot2" (emulating ggplot2 default color palette). Alternatively, any hyphen-separated HTML color names, e.g. "lightyellow-orange" (by default), "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names

legend.position
: the legend position. If NA, the legend is not shown

legend.horiz
: logical specifying the legend horizon. If TRUE, set the legend horizontally rather than vertically

mtext.side
: the side of marginal text. If NA, it is not shown

verbose
: logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

edge.width
: the width of the edge. If NULL, the width edge is proportional to the 'weight' edge attribute (if existed)

vertex.size
: the size of each vertex. If null, each vertex has the size proportional to the degree of nodes

vertex.size.nonseed
: the size of each nonseed vertex. If null, each vertex has the size proportional to the degree of nodes

vertex.label.color
: the color of vertex labels

vertex.label.color.nonseed
: the color of nonseed vertex labels

...
: additional graphic parameters. See http://igraph.org/r/doc/plot.common.html for the complete list.

Value

a subgraph, an object of class "igraph".

See Also

xColormap

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
## TNFRSF1A
xVisEvidence(xTarget, nodes="TNFRSF1A", neighbor.order=1,
neighbor.seed=TRUE, neighbor.top=NULL, vertex.label.color="black",
vertex.label.cex=0.7, vertex.label.dist=0.6, vertex.label.font=1,
vertex.label.family="Arial", legend.position="bottomleft",
legend.horiz=TRUE, newpage=FALSE)
## UBA52
xVisEvidence(xTarget, nodes="UBA52", neighbor.order=1,
neighbor.seed=TRUE, neighbor.top=20, vertex.label.color="black",
vertex.label.cex=0.7, vertex.label.dist=0.6, vertex.label.font=1,
legend.position="bottomleft", legend.horiz=TRUE, newpage=FALSE)

## End(Not run)
```

xVisEvidenceAdv *Function to visualise evidence and priority scores for prioritised genes in a gene network*

Description

xVisEvidenceAdv is supposed to visualise evidence and priority scores for prioritised genes in a gene network. It returns an object of class "ggplot".

Usage

```
xVisEvidenceAdv(
xTarget,
g = NA,
nodes = NULL,
node.info = c("smart", "none"),
neighbor.order = 1,
neighbor.seed = TRUE,
neighbor.top = NULL,
largest.comp = TRUE,
node.label.size = 2,
node.label.color = "black",
node.label.alpha = 0.9,
node.label.padding = 0.5,
node.label.arrow = 0,
node.label.force = 0.1,
node.shape = 19,
node.color.title = "Pi\nrating",
colormap = "white-yellow-red",
ncolors = 64,
zlim = c(0, 5),
node.size.range = 5,
title = "",
edge.color = "orange",
edge.color.alpha = 0.5,
edge.curve = 0,
edge.arrow.gap = 0.025,
pie.radius = NULL,
pie.color = "black",
pie.color.alpha = 1,
pie.thick = 0.1,
...
)
```

Arguments

xTarget	an object of class "dTarget", "sTarget" or "eTarget"

g	an object of class "igraph". If NA, the 'metag' will be used, which is part of the input object "xTarget". If provided, it must have a node attribute called 'priority'
nodes	which node genes are in query. If NULL, the top gene will be queried
node.info	tells the additional information used to label nodes. It can be one of "none" (only gene labeling), "smart" for (by default) using three pieces of information (if any): genes, 5-star ratings, and associated ranks (marked by an @ icon)
neighbor.order	an integer giving the order of the neighborhood. By default, it is 1-order neighborhood
neighbor.seed	logical to indicate whether neighbors are seeds only. By default, it sets to true
neighbor.top	the top number of the neighbors with the highest priority. By default, it sets to NULL to disable this parameter
largest.comp	logical to indicate whether the largest component is only retained. By default, it sets to true for the largest component being left
node.label.size	a vector specifying node size or a character specifying which node attribute used for node label size
node.label.color	the node label color
node.label.alpha	the 0-1 value specifying transparency of node labelling
node.label.padding	the padding around the labeled node
node.label.arrow	the arrow pointing to the labeled node
node.label.force	the repelling force between overlapping labels
node.shape	an integer specifying node shape
node.color.title	a character specifying the title for node coloring
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta), and "ggplot2" (emulating ggplot2 default color palette). Alternatively, any hyphen-separated HTML color names, e.g. "lightyellow-orange" (by default), "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
ncolors	the number of colors specified over the colormap
zlim	the minimum and maximum values for which colors should be plotted
node.size.range	the range of actual node size
title	a character specifying the title for the plot
edge.color	a character specifying the edge colors

edge.color.alpha	the 0-1 value specifying transparency of edge colors
edge.curve	a numeric value specifying the edge curve. 0 for the straight line
edge.arrow.gap	a gap between the arrow and the node
pie.radius	the radius of a pie. If NULL, it equals roughly 1/75
pie.color	the border color of a pie
pie.color.alpha	the 0-1 value specifying transparency of pie border colors
pie.thick	the pie border thickness
...	additional graphic parameters for xGGnetwork

Value

a ggplot object.

See Also

xVisEvidence, xGGnetwork, xPieplot

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
## TNFRSF1A
xVisEvidenceAdv(xTarget, nodes="TNFRSF1A", neighbor.order=1,
neighbor.seed=TRUE, neighbor.top=NULL)

## End(Not run)
```

---

xVisKernels *Function to visualise distance kernel functions*

Description

xVisKernels is supposed to visualise distance kernels, each of which is a decaying function of: i) the relative distance d_{gs} between the gene g and the SNP s, and ii) the decay exponent λ.

Usage

```
xVisKernels(exponent = 2, newpage = TRUE)
```

Arguments

exponent	an integer specifying decay exponent. By default, it sets to 2
newpage	logical to indicate whether to open a new page. By default, it sets to true for opening a new page

Value

invisible

Note

There are five kernels that are currently supported:

- For "slow decay" kernel, $h_{ds}(t) = 1 - d_{gs}/D\lambda * (d_{gs} \leq D)$
- For "linear decay" kernel, $h_{ds}(t) = 1 - d_{gs}/D * (d_{gs} \leq D)$
- For "rapid decay" kernel, $h_{ds}(t) = 1 - d_{gs}/D^{\lambda} * (d_{gs} \leq D)$

See Also

xVisKernels

Examples

```
# visualise distance kernels
xVisKernels(exponent=2)
xVisKernels(exponent=3)
```

---

xVisNet *Function to visualise a graph object of class "igraph"*

Description

xVisNet is supposed to visualise a graph object of class "igraph". It also allows vertices/nodes color-coded according to the input pattern.

Usage

```
xVisNet(
g,
pattern = NULL,
colormap = c("yr", "jet", "gbr", "wyr", "br", "bwr", "rainbow", "wb"),
ncolors = 40,
zlim = NULL,
colorbar = TRUE,
newpage = TRUE,
glayout = layout_with_kk,
vertex.frame.color = NA,
vertex.size = NULL,
vertex.color = NULL,
vertex.shape = NULL,
vertex.label = NULL,
vertex.label.cex = NULL,
vertex.label.dist = 0.3,
```

```
vertex.label.color = "blue",
vertex.label.family = "sans",
edge.arrow.size = 0.3,
...
)
```

Arguments

g	an object of class "igraph"
pattern	a numeric vector used to color-code vertices/nodes. Notably, if the input vector contains names, then these names should include all node names of input graph, i.e. V(g)$name, since there is a mapping operation. After mapping, the length of the patern vector should be the same as the number of nodes of input graph; otherwise, this input pattern will be ignored. The way of how to color-code is to map values in the pattern onto the whole colormap (see the next arguments: colormap, ncolors, zlim and colorbar)
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta), and "ggplot2" (emulating ggplot2 default color palette). Alternatively, any hyphen-separated HTML color names, e.g. "lightyellow-orange" (by default), "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
ncolors	the number of colors specified over the colormap
zlim	the minimum and maximum z/patttern values for which colors should be plotted, defaulting to the range of the finite values of z. Each of the given colors will be used to color an equispaced interval of this range. The midpoints of the intervals cover the range, so that values just outside the range will be plotted
colorbar	logical to indicate whether to append a colorbar. If pattern is null, it always sets to false
newpage	logical to indicate whether to open a new page. By default, it sets to true for opening a new page
glayout	either a function or a numeric matrix configuring how the vertices will be placed on the plot. If layout is a function, this function will be called with the graph as the single parameter to determine the actual coordinates. This function can be one of "layout_nicely" (previously "layout.auto"), "layout_randomly" (previously "layout.random"), "layout_in_circle" (previously "layout.circle"), "layout_on_sphere" (previously "layout.sphere"), "layout_with_fr" (previously "layout.fruchterman.reingold"), "layout_with_kk" (previously "layout.kamada.kawai"), "layout_as_tree" (previously "layout.reingold.tilford"), "layout_with_lgl" (previously "layout.lgl"), "layout_with_graphopt" (previously "layout.graphopt"), "layout_with_sugiyama" (previously "layout.kamada.kawai"), "layout_with_dh" (previously "layout.davidson.harel"), "layout_with_drl" (previously "layout.drl"),

"layout_with_gem" (previously "layout.gem"), "layout_with_mds", and "layout_as_bipartite". A full explanation of these layouts can be found in http://igraph.org/r/doc/layout_nicely.html

`vertex.frame.color`
: the color of the frame of the vertices. If it is NA, then there is no frame

`vertex.size`
: the size of each vertex. If it is a vector, each vertex may differ in size

`vertex.color`
: the fill color of the vertices. If it is NA, then there is no fill color. If the pattern is given, this setup will be ignored

`vertex.shape`
: the shape of each vertex. It can be one of "circle", "square", "csquare", "rectangle", "crectangle", "vrectangle", "pie" (http://igraph.org/r/doc/vertex.shape.pie.html), "sphere", and "none". If it sets to NULL, these vertices with negative will be "csquare" and the rest "circle".

`vertex.label`
: the label of the vertices. If it is NA, then there is no label. The default vertex labels are the name attribute of the nodes

`vertex.label.cex`
: the font size of vertex labels.

`vertex.label.dist`
: the distance of the label from the center of the vertex. If it is 0 then the label is centered on the vertex. If it is 1 then the label is displayed beside the vertex

`vertex.label.color`
: the color of vertex labels

`vertex.label.family`
: the font family of vertex labels

`edge.arrow.size`
: the size of the arrows for the directed edge. The default value is 1.

`...`
: additional graphic parameters. See http://igraph.org/r/doc/plot.common.html for the complete list.

Value

invisible

Note

none

See Also

xVisNet

Examples

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# 1) generate a ring graph
g <- make_ring(10, directed=TRUE)

# 2) visualise the graph
```

```
# 2a) visualise in one go
xVisNet(g=g, vertex.shape="sphere", glayout=layout_with_kk)
# 2b) visualise the graph with layout first calculated
glayout <- layout_(g, with_kk(), component_wise())
xVisNet(g=g, vertex.shape="sphere", glayout=glayout)
# 2c) visualise the graph with layout appended to the graph itself
g <- add_layout_(g, with_kk(), component_wise())
xVisNet(g=g, vertex.shape="sphere")

# 4) visualise the graph with vertices being color-coded by the pattern
pattern <- runif(vcount(g))
names(pattern) <- V(g)$name
xVisNet(g=g, pattern=pattern, colormap="bwr", vertex.shape="sphere")

# 5) use font family (Arial)
BiocManager::install("extrafont")
library(extrafont)
font_import()
fonttable()
## creating PDF files with fonts
loadfonts()
pdf("xVisNet_fonts.pdf", family="Arial Black")
xVisNet(g=g, newpage=FALSE, vertex.label.family="Arial Black",
signature=F)
dev.off()

## End(Not run)
```

Index