Package ‘Pi’

April 16, 2019

Type Package

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

Version 1.10.0

Date 2018-9-11

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

Maintainer Hai Fang <`hfang@well.ox.ac.uk`>

Depends XGR, igraph, dnet, ggplot2, graphics

Imports Matrix, MASS, ggbio, GenomicRanges, GenomeInfoDb, supraHex, scales, grDevices, ggrepel, ROCR, randomForest, glmnet, Gviz, lattice, caret, plot3D, stats

Suggests foreach, doParallel, BiocStyle, knitr, rmarkdown, png, GGally, gridExtra, ExpressionAtlas, ggforce, fgsea, pathview, tidyr

Description Priority index or Pi is developed as a genomic-led target prioritisation system, with the focus on leveraging human genetic data to prioritise potential drug targets at the gene, pathway and network level. The long term goal is to use such information to enhance early-stage target validation. Based on evidence of disease association from genome-wide association studies (GWAS), this prioritisation system is able to generate evidence to support identification of the specific modulated genes (seed genes) that are responsible for the genetic association signal by utilising knowledge of linkage disequilibrium (co-inherited genetic variants), distance of associated variants from the gene, evidence of independent genetic association with gene expression in disease-relevant tissues, cell types and states, and evidence of physical interactions between disease-associated genetic variants and gene promoters based on genome-wide capture HiC-generated promoter interactomes in primary blood cell types. Seed genes are scored in an integrative way, quantifying the genetic influence. Scored seed genes are subsequently used as baits to rank seed genes plus additional (non-seed) genes; this is achieved by iteratively exploring the global connectivity of a gene interaction network. Genes with the highest priority are further used to identify/prioritise pathways that are significantly enriched with highly prioritised genes. Prioritised genes are also used to identify a gene network interconnecting highly prioritised genes and a minimal number of less prioritised genes (which act as linkers bringing together highly prioritised genes).

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' 'xSNP2eGenes.r' 'xPierSNPsConsensus.r' 'xPredictROCR.r' 'xPredictCompare.r' 'xContour.r' 'xSNP2cGenes.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' 'xPierKEGG.r' 'xVisEvidenceAdv.r' 'xPieplot.r' 'xPierCor.r' 'xPierGRs.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_8

git_last_commit 4801867

git_last_commit_date 2018-10-30

Date/Publication 2019-04-15

R topics documented:

`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)
```

`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)
```

`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)
```

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)
```

`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

`xPierGSEA`

Examples

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

## End(Not run)
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

`xPierGSEA`

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 = NULL,
subtitle = c("leading", "enrichment", "both", "none"),
clab = "5-star\nrating", 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

`xPierGSEA`

Examples

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

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

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

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

`xRDataLoader`, `xPredictROCR`, `xMLrandomforest`

Examples

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

## End(Not run)
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)
```

`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")
```

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

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 nGene X 6 containing gene priority information, where nGene is the number of genes in the input data frame, and the 6 columns are "GS" (either 'GSP', or 'GSN', or 'Putative'), "name" (gene names), "rank" (ranks of the priority scores), "priority" (5-star priority score), 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"

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","randomFor`

Examples

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

```
## End(Not run)
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

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

## End(Not run)
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

`xMLrandomforest`

Examples

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

## End(Not run)
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"),
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)
`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

`xMLrandomforest`

Examples

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

## End(Not run)
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

Examples

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

## End(Not run)
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", ...)
```

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
`...`	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

Examples

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

## End(Not run)
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

`xMLparameters`

Examples

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

## End(Not run)
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", ...)
```

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
`...`	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 7 containing gene priority information, where nGene is the number of genes in the input data frame, and the 7 columns are "GS" (either 'GSP', or 'GSN', or 'NEW'), "name" (gene names), "rank" (ranks of the priority scores), "pvalue" (the cross-fold aggregated p-value of being GSP, per-fold 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), 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"

Note

none

Examples

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

## End(Not run)
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

`xMLrandomforest`

Examples

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

## End(Not run)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
old_names <- colnames(sTarget$predictor)[-c(1,2)]
new_names <- c('Annotation_disease genes (dGenes)',
'Annotation_phenotype genes (pGenes)', 'Annotation_function genes
(fGenes)', 'nearbyGenes_nearby genes (nGenes)', 'eQTL_eQTL: resting
state (CD14+)', 'eQTL_eQTL: activating state (CD14+ by LPS2h)',
'eQTL_eQTL: activating state (CD14+ by LPS24h)', 'eQTL_eQTL: activating
state (CD14+ by IFN24h)', 'eQTL_eQTL: B cells', 'eQTL_eQTL: monocytes',
'eQTL_eQTL: CD4+ T cells', 'eQTL_eQTL: CD8+ T cells', 'eQTL_eQTL:
neutrophils', 'eQTL_eQTL: NK cells','eQTL_eQTL: peripheral blood',
'HiC_Hi-C: monocytes', 'HiC_Hi-C: macrophages (M0)', 'HiC_Hi-C:
macrophages (M1)', 'HiC_Hi-C: macrophages (M2)', 'HiC_Hi-C:
neutrophils', 'HiC_Hi-C: CD4+ T cells (naive)', 'HiC_Hi-C: CD4+ T cells
(total)', 'HiC_Hi-C: CD8+ T cells (naive)', 'HiC_Hi-C: CD8+ T cells
(total)', 'HiC_Hi-C: B cells (naive)', 'HiC_Hi-C: B cells (total)')
sTarget_rename <- xMLrename(sTarget, old_names, 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

`xMLrandomforest`

Examples

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

## End(Not run)
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 = c("ggplot2", "bwr", "jet", "gbr",
"wyr",
"br", "yr", "rainbow", "wb", "heat", "terrain", "topo", "cm"),
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

`xPieplot`

Examples

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

## End(Not run)
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

`xRDataLoader`, `xRWR`, `xPierSNPs`, `xPierGenes`, `xPierPathways`

Examples

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

## End(Not run)

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

---

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")
```

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","coex 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 normali-

	sation, '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

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

`xRDataLoader`, `xPierSNPs`, `xPier`, `xPierPathways`

Examples

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

## End(Not run)

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

`xPierCor`

Examples

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

## End(Not run)

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("priority", "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")
```

Arguments

`list_xTarget`	a list of "dTarget"/"sTarget" objects or a "dTarget"/"sTarget" object
`displayBy`	which priority will be extracted. It can be "priority" for priority score (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

Value

If aggregateBy is 'none' (by default), a data frame containing priority matrix, with each column/disease for either priority score, 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

`xPierMatrix`

Examples

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

## End(Not run)
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

xPierSNPsAdv

Examples

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

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

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","coex 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

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

`xRDataLoader`, `xPierSNPs`, `xPier`, `xPierPathways`

Examples

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

## End(Not run)

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")
```

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","coe 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

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

xPierGenes

Examples

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

## End(Not run)

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")
```

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

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

xGSEAbarplot, xGSEAdotplot

Examples

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

## End(Not run)

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)
```

---

xPierKEGG *Function to visualise prioritised genes in terms of a KEGG pathway*

Description

xPierKEGG is supposed to visualise prioritised genes in terms of a KEGG pathway. It returns an object of class "igraph".

Usage

```
xPierKEGG(xTarget, vis = c("net", "evidence", "pathview"),
hsa = "hsa04621", priority.top = NULL, incoming.neighbor.order = 1,
nodes_query = NULL, largest.comp = TRUE, pathview.filename = NULL,
pathview.filetype = c("png", "pdf"), verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata", ...)
```

Arguments

`xTarget`	an object of class "dTarget" or "sTarget"
`vis`	the type of visualisation for a KEGG pathway. It can be one of "net" (visualising the network with nodes colored by priority score; by default), "evidence" for visualising the network with nodes as pie charts, and "pathview" for using the package "pathview"
`hsa`	the identity of KEGG pathway in query. The full list of pathways in human can be found at `http://www.genome.jp/kegg-bin/show_organism?menu_type=pathway_maps&org=hsa`. For example, 'hsa04621' for 'NOD-like receptor signaling pathway', where the prefix 'hsa' can be ignored
`priority.top`	the number of the top targets. By default, it is NULL meaning no such restriction
`incoming.neighbor.order`	an integer giving the order of the incoming neighborhood. By default, it is 1-order incoming neighborhood
`nodes_query`	which gene in query will be visualised. It (if not null) has the high priority over nodes selected by 'priority.top' and 'incoming.neighbor.order' above
`largest.comp`	logical to indicate whether the largest component is only retained. By default, it sets to true for the largest component being left
`pathview.filename`	the file name saved using the package "pathview". By default, it is NULL meaning "hsa.Pi"
`pathview.filetype`	the file format saved using the package "pathview". It can be "png" or "pdf"
`verbose`	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true
`RData.location`	the characters to tell the location of built-in RData files. See `xRDataLoader` for details
`...`	additional graphic parameters. If the type of visualisation is 'net', see `xVisNet`; if the visualisation type is 'evidence', see `xVisEvidence`

Value

a subgraph, an object of class "igraph".

Note

If vis is 'pathview', it will depend on whether a package "pathview" has been installed. It can be installed via: `BiocManager::install("pathview")`.

See Also

`xVisNet`, `xVisEvidence`

Examples

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

## End(Not run)
```

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
## evidence
xPierKEGG(xTarget, hsa="hsa04621", vis="evidence",
RData.location=RData.location)
## network
xPierKEGG(xTarget, hsa="hsa04621", vis="net",
RData.location=RData.location)
## using pathview
pv.out <- xPierKEGG(xTarget, hsa="hsa04621", vis="pathview",
pathview.filetype=c("png","pdf")[2], RData.location=RData.location)

## 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", ...)
```

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
`...`	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`, `xPier`, `xPierSNPs`, `xPierGenes`, `xPierPathways`

Examples

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

## End(Not run)

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")
```

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 (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

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 number of different types of seeds, followed by details on individual type of seeds (that is, "OMIM", "Phenotype", "Function", "nearbyGenes", "eQTL", "HiC")
- `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 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) 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

Note

none

See Also

xPierSNPsAdv

Examples

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

## End(Not run)
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)
```

---

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")
```

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

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

xRDataLoader, xEnricher

Examples

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

## End(Not run)

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

Examples

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

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

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","coe 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

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`, `xPier`, `xPierGenes`, `xPierPathways`

Examples

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

## End(Not run)

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")
```

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-s`
`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
`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","coex 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

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`, `xPierMatrix`

Examples

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

## End(Not run)

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'

## End(Not run)
```

---

xPierSNPsConsensus

Function to resolve relative importance of distance weight and eQTL weight priorising consensus gene ranks given a list of seed SNPs together with the significance level (e.g. GWAS reported p-values)

Description

xPierSNPsConsensus is supposed to priorise genes given a list of seed SNPs together with the significance level. It is a parameter-free version of xPierSNPs identifying the consensus rank (less sensitive to the relative importance of the distance weight and eQTL weight). It returns an object of class "pNode" but appended with components on optimal distance weight and consensus info

Usage

```
xPierSNPsConsensus(data, include.LD = NA, LD.customised = NULL,
LD.r2 = 0.8, significance.threshold = 5e-05, distance.max = 2e+05,
decay.kernel = c("rapid", "slow", "linear"), decay.exponent = 2,
GR.SNP = c("dbSNP_GWAS", "dbSNP_Common"),
GR.Gene = c("UCSC_knownGene", "UCSC_knownCanonical"),
include.eQTL = c(NA, "JKscience_TS2A", "JKscience_TS2B",
"JKscience_TS3A", "JKng_bcell", "JKng_mono", "JKnc_neutro", "JK_nk",
"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", "eQTLdb_NK", "eQTLdb_CD14",
"eQTLdb_LPS2", "eQTLdb_LPS24", "eQTLdb_IFN"),
eQTL.customised = NULL, 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"), weighted = FALSE, network.customised = NULL,
normalise = c("laplacian", "row", "column", "none"), restart = 0.75,
normalise.affinity.matrix = c("none", "quantile"), parallel = TRUE,
multicores = NULL, verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

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
`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
`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"
`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"
`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: immune stimulation in monocytes ('JKscience_TS1A' and 'JKscience_TS2B' for cis-eQTLs or 'JKscience_TS3A' for trans-eQTLs) from Science 2014, 343(6175):1246949; cis- and trans-eQTLs in B cells ('JKng_bcell') and in monocytes ('JKng_mono') from Nature Genetics 2012, 44(5):502-510; cis- and trans-eQTLs in neutrophils ('JKnc_neutro') from Nature Communications 2015, 7(6):7545; cis-eQTLs in NK cells ('JK_nk') which is unpublished. Also supported are GTEx cis-eQTLs from Science 2015, 348(6235):648-60, including 13 tissues: 'GTEx_Adipose_Subcutaneous','GTEx_Artery_Aorta','GT
`eQTL.customised`	a user-input matrix or data frame with 3 columns: 1st column for SNPs/eQTLs, 2nd column for Genes, and 3rd for eQTL mapping significance level (p-values or FDR). It is designed to allow the user analysing their eQTL data. This customisation (if provided) has the high priority over 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
`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 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
`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
`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

Value

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

- `priority`: a matrix of nNode X 4 containing node priority information, where nNode is the number of nodes in the input graph, and the 4 columns are "name" (node names), "seed" (1 for seeds, 0 for non-seeds), "weight" (weight/score values for seed genes), "priority" (the priority scores that are rescaled to the range [0,1]), "rank" (ranks of the priority scores), and two additional columns: 'driver' telling who drives the prioritisation ('nGenes','eGenes' or'both'), and 'consensus_rank'
- `g`: an input "igraph" object
- `SNP`: a data frame of nSNP X 3 containing input SNPs and/or LD SNPs info, where nSNP is the number of input SNPs and/or LD SNPs, and the 3 columns are "SNP" (dbSNP), "Score" (the SNP score), "Pval" (the SNP p-value)
- `Gene2SNP`: a matrix of Genes X SNPs, each non-zero cell telling an SNP's genetic influential score on a seed gene
- `nGenes`: the relative weight for nearby genes
- `consensus`: a matrix containing details on rank results by decreasing the relative importance of nGenes. In addition to rank matrix, it has columns 'rank_median' for median rank excluding two extremes 'n_1' (nGenes only) and 'n_0' (eGenes only), 'rank_MAD' for median absolute deviation, 'driver' telling who drives the prioritisation ('nGenes','eGenes' or'both'), 'consensus_rank' for the rank of the median rank list
- `call`: the call that produced this result

Note

none

See Also

`xPierSNPs`

Examples

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

## End(Not run)

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 <- xPierSNPsConsensus(data=AS, include.LD="EUR",
include.eQTL=c("JKscience_TS2A","JKscience_TS3A"),
network="PCommonsUN_medium", restart=0.7)

# c) save to the file called 'SNPs_priority.consensus.txt'
```

```
write.table(pNode$priority, file="SNPs_priority.consensus.txt",
sep="\t", row.names=FALSE)

# d) manhattan plot
mp <- xPierManhattan(pNode, highlight.top=10)
#pdf(file="Gene_manhattan.pdf", height=6, width=12, compress=TRUE)
print(mp)
#dev.off()

## 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")
```

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","coex 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

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

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

## End(Not run)
```

```
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", ...)
```

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
`...`	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

`xMLrandomforest`

Examples

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

## End(Not run)

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", ...)
```

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
`...`	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

`xMLrandomforest`

Examples

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

## End(Not run)
```

```
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

`xPredictROCR`

Examples

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

## End(Not run)
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

Examples

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

## End(Not run)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
pPerf <- xPredictROCR(prediction, GSP, GSN)

## 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

`xPier`

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)
```

`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")
```

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

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

`xRDataLoader`

Examples

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

## End(Not run)

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")
```

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

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

`xRDataLoader`

Examples

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

## End(Not run)

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)
```

`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"
`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

`xPierMatrix`

Examples

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

## End(Not run)
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"
`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`

Examples

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

## End(Not run)
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)
```

Index