Package 'XGR'

June 26, 2016

Type Package

Title Exploring Genomic Relations for Precision Interpretation Through Enrichment, Similarity, Network and Annotation Analysis

Version 1.0.2

Date 2016-6-25

Author Hai Fang, Bogdan Knezevic, Katie L Burnham, Julian C Knight

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

Depends R (>= 3.1.0), igraph, dnet, ggplot2

Imports Matrix, RCircos, grDevices, graphics, GenomicRanges, IRanges, S4Vectors, supraHex, rtracklayer, stats, BiocGenerics

Suggests foreach, doMC

Description The central goal of XGR is to provide a data interpretation system. It is designed to make a user-defined gene or SNP list (or genomic regions) more interpretable by comprehensively utilising ontology annotations and interaction networks to reveal relationships and enhance opportunities for biological discovery. XGR is unique in supporting a broad range of ontologies (including knowledge of biological and molecular functions, pathways, diseases and phenotypes - in both human and mouse) and different types of networks (including functional, physical and pathway interactions). There are two core functionalities of XGR. The first is to provide basic infrastructures for easy access to built-in ontologies and networks. The second is to support data interpretations via 1) enrichment analysis using either built-in or custom ontologies, 2) similarity analysis for calculating semantic similarity between genes (or SNPs) based on their ontology annotation profiles, 3) network analysis for identification of gene networks given a query list of (significant) genes or SNPs, and 4) annotation analysis for interpreting genomic regions using co-localised functional genomic annotations (such as open chromatin, epigenetic marks and TF binding sites) and using nearby gene annotations (by ontologies). Together with its web app, XGR aims to provide a user-friendly tool for exploring genomic relations at the gene, SNP and genomic region level.

URL http://XGR.r-forge.r-project.org, http://galahad.well.ox.ac.uk/XGR

Collate 'xRDataLoader.r' 'xRdWrap.r' 'xFunArgs.r' 'xRd2HTML.r' 'xDAGanno.r' 'xDAGsim.r' 'xConverter.r' 'xEnricher.r' 'xEnricherGenes.r' 'xEnricherSNPs.r' 'xEnricherYours.r' 'xEnrichViewer.r' 'xEnrichBarplot.r' 'xEnrichDAGplot.r'

'xEnrichNetplot.r' 'xEnrichCompare.r' 'xEnrichDAGplotAdv.r'
'xSocialiser.r' 'xSocialiserGenes.r' 'xSocialiserSNPs.r'
'xSocialiserDAGplot.r' 'xSocialiserDAGplotAdv.r'
'xSocialiserNetplot.r' 'xCircos.r' 'xSubneterGenes.r'
'xSubneterSNPs.r' 'xVisNet.r' 'xVisKernels.r' 'xSNPscores.r'
'xSNP2nGenes.r' 'xSparseMatrix.r' 'xSNP2GeneScores.r'
'xGRviaGeneAnno.r' 'xGRviaGenomicAnno.r'
'xGRviaGenomicAnnoAdv.r' 'xGRsampling.r' 'xLiftOver.r'
'xEnrichConciser.r'

License GPL-2

biocViews Bioinformatics

NeedsCompilation no

Repository CRAN

Date/Publication 2016-06-26 09:12:00

R topics documented:

`ImmunoBase` *Immune-disease associated variants, regions and genes from ImmunoBase (hg19)*

Description

This dataset contains data obtained from ImmunoBase. For each of 20 immune-diseases, its associated variants, regions, and nearby genes (within 500kb) are stored.

Usage

```
data(ImmunoBase)
```

Value

a list with 5 components:

- `disease`: a character of disease name
- `variants`: an object of class "GRanges", storing genomic locations of associated variants plus their significance and odd ratios
- `regions`: an object of class "GRanges", storing genomic locations of associated regions
- `genes_variants`: a named vector for nearby genes within 500kb of associated variants; the element names are gene symbols, the element values for the shortest distance to all associated variants
- `genes_regions`: a named vector for nearby genes within 500kb of associated regions; the element names are gene symbols, the element values for the shortest distance to all associated regions

Examples

```
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase')
names(ImmunoBase)
str(ImmunoBase$AS)
ImmunoBase$AS$disease
ImmunoBase$AS$variants
head(ImmunoBase$AS$genes_variants)
head(ImmunoBase$AS$genes_regions)
```

JKscience_TS2A *Table S2A for cis-eQTLs among shared datasets from Benjamin et al. (2014)*

Description

This dataset involves 228 individuals with expression data for all four conditions, that is, in the naive state (Naive), after 2-hour LPS (LPS2), after 24-hour LPS (LPS24), and after exposure to IFN (IFN). Local, likely cis-acting eQTL (referred to as cis-eQTL) were defined as SNPs showing association with gene expression that were located within a 1-Mb window of the associated probe. The eQTL analysis was performed with the R package MatrixEQTL using an additive linear model, reporting both test statistic and FDR.

Usage

```
data(JKscience_TS2A)
```

Value

a data frame. It has the following columns: "variant" (cis-eQTLs), "ArrayAddress" (illuminaHumanv4), "GeneID" (Entrez GeneID), "Symbol" (gene symbol), "Naive_t" (test statistic for naive samples), "LPS2_t", "LPS24_t", "IFN_t", "Naive_fdr" (FDR for naive samples), "LPS2_fdr", "LPS24_fdr" and "IFN_fdr".

References

Fairfax et al. (2014). Innate immune activity conditions the effect of regulatory variants upon monocyte gene expression. *Science*, 343(6175):1246949.

Examples

```
JKscience_TS2A <- xRDataLoader(RData.customised='JKscience_TS2A')
JKscience_TS2A[1:5,]
```

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

Description

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

Usage

```
xCircos(g, entity = c("SNP", "Gene"), top_num = 50, colormap = c("yr",
"bwr", "jet", "gbr", "wyr", "br", "rainbow", "wb",
"lightyellow-orange"),
rescale = T, nodes.query = NULL, ideogram = T, chr.exclude = "auto",
entity.label.cex = 0.7, entity.label.side = c("out", "in"),
entity.label.track = 1, entity.label.query = NULL,
GR.SNP = c("dbSNP_GWAS", "dbSNP_Common"), GR.Gene = c("UCSC_knownGene",
"UCSC_knownCanonical"), verbose = T,
RData.location =
"https://github.com/hfang-bristol/RDataCentre/blob/master/Portal")
```

Arguments

g	an object of class "igraph". For example, it stores semantic similarity results with nodes for genes/SNPs and edges for pair-wise semantic similarity between them
entity	the entity of similarity analysis for which results are being plotted. It can be either "SNP" or "Gene"
top_num	the top number of similarity edges to be plotted
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "lightyellow-orange" (by default), "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
rescale	logical to indicate whether the edge values are rescaled to the range [0,1]. By default, it sets to true
nodes.query	nodes in query for which edges attached to them will be displayed. By default, it sets to NULL meaning no such restriction
ideogram	logical to indicate whether chromosome banding is plotted
chr.exclude	a character vector of chromosomes to exclude from the plot, e.g. c("chrX", "chrY"). By defautl, it is 'auto' meaning those chromosomes without data will be excluded. If NULL, no chromosome is excluded

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

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

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

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

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

`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"

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

Note

none

See Also

`xSocialiserGenes`, `xSocialiserSNPs`

Examples

```
## Not run:
# Load the library
library(XGR)
library(RCircos)
RData.location="~/Sites/SVN/github/RDataCentre/Portal"

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

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

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

## End(Not run)
```

`xConverter` *Function to convert an object between graph classes*

Description

`xConverter` is supposed to convert an object between classes 'dgCMatrix' and 'igraph'.

Usage

```
xConverter(obj, from = c("dgCMatrix", "igraph"), to = c("igraph",
"dgCMatrix"), verbose = TRUE)
```

Arguments

obj	an object of class "dgCMatrix" or "igraph"
from	a character specifying the class converted from. It can be one of "dgCMatrix" and "igraph"
to	a character specifying the class converted to. It can be one of "dgCMatrix" and "igraph"
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 "dgCMatrix" or "igraph"

Note

Conversion is also supported between classes 'dgCMatrix' and 'igraph'

See Also

xRDataLoader

Examples

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

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

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

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

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

## End(Not run)
```

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

Description

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

Usage

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

Arguments

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

Value

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

Note

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

See Also

xRDataLoader

Examples

```
## Not run:
# 1) SNP-based ontology
# 1a) ig.EF (an object of class "igraph" storing as a directed graph)
g <- xRDataLoader('ig.EF')

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

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

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

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

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

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

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

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

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

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

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

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

## End(Not run)
```

`xDAGsim` *Function to calculate pair-wise semantic similarity between input terms based on a direct acyclic graph (DAG) with annotated data*

Description

`xDAGsim` is supposed to calculate pair-wise semantic similarity between input terms based on a direct acyclic graph (DAG) with annotated data. It returns an object of class "igraph", a network representation of input terms. Parallel computing is also supported for Linux or Mac operating systems.

Usage

```
xDAGsim(g, terms = NULL, method.term = c("Resnik", "Lin", "Schlicker",
"Jiang", "Pesquita"), fast = T, parallel = TRUE, multicores = NULL,
verbose = T)
```

Arguments

`g`	an object of class "igraph". It must contain a vertex attribute called 'anno' for storing annotation data (see example for howto)
`terms`	the terms/nodes between which pair-wise semantic similarity is calculated. If NULL, all terms in the input DAG will be used for calcluation, which is very prohibitively expensive!
`method.term`	the method used to measure semantic similarity between input terms. It can be "Resnik" for information content (IC) of most informative common ancestor (MICA) (see http://dl.acm.org/citation.cfm?id=1625914), "Lin" for

2*IC at MICA divided by the sum of IC at pairs of terms, "Schlicker" for weighted version of 'Lin' by the 1-prob(MICA) (see http://www.ncbi.nlm.nih.gov/pubmed/16776819), "Jiang" for 1 - difference between the sum of IC at pairs of terms and 2*IC at MICA (see http://arxiv.org/pdf/cmp-lg/9709008.pdf), "Pesquita" for graph information content similarity related to Tanimoto-Jacard index (ie. summed information content of common ancestors divided by summed information content of all ancestors of term1 and term2 (see http://www.ncbi.nlm.nih.gov/pubmed/18460186)). By default, it uses "Schlicker" method

`fast` logical to indicate whether a vectorised fast computation is used. By default, it sets to true. It is always advisable to use this vectorised fast computation; since the conventional computation is just used for understanding scripts

`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. It can be installed via: `source("http://bioconductor.org/biocLite.R"); biocLite(c("foreach","doMC"))`. If not yet installed, this option will be disabled

`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 an object of class "igraph", with nodes for input terms and edges for pair-wise semantic similarity between terms.

Note

none

See Also

xDAGanno, xConverter

Examples

```
## Not run:
# 1) SNP-based ontology
# 1a) ig.EF (an object of class "igraph" storing as a directed graph)
g <- xRDataLoader('ig.EF')
g

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

```
# 1c) prepare for ontology and its annotation information
dag <- xDAGanno(g=g, annotation=anno, path.mode="all_paths",
true.path.rule=TRUE, verbose=TRUE)

# 1d) calculate pair-wise semantic similarity between 5 randomly chosen terms
terms <- sample(V(dag)$name, 5)
sim <- xDAGsim(g=dag, terms=terms, method.term="Schlicker",
parallel=FALSE)
sim

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

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

# 2c) prepare for annotation data
dag <- xDAGanno(g=g, annotation=anno, path.mode="all_paths",
true.path.rule=TRUE, verbose=TRUE)

# 2d) calculate pair-wise semantic similarity between 5 randomly chosen terms
terms <- sample(V(dag)$name, 5)
sim <- xDAGsim(g=dag, terms=terms, method.term="Schlicker",
parallel=FALSE)
sim

## End(Not run)
```

`xEnrichBarplot` *Function to visualise enrichment results using a barplot*

Description

`xEnrichBarplot` is supposed to visualise enrichment results using a barplot. It returns an object of class "ggplot".

Usage

```
xEnrichBarplot(eTerm, top_num = 10, displayBy = c("fc", "adjp", "fdr",
"zscore", "pvalue"), FDR.cutoff = 0.05, bar.label = TRUE,
bar.label.size = 3, wrap.width = NULL)
```

Arguments

`eTerm`	an object of class "eTerm"
`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 "fc" for enrichment fold change (by default), "adjp" or "fdr" for adjusted p value (or FDR), "pvalue" for p value, "zscore" for enrichment z-score
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
wrap.width	a positive integer specifying wrap width of name

Value

an object of class "ggplot"

Note

none

See Also

xEnricherGenes, xEnricherSNPs, xEnrichViewer

Examples

```
## Not run:
# Load the library
library(XGR)
RData.location="~/Sites/SVN/github/bigdata"

# 1) load eQTL mapping results: cis-eQTLs significantly induced by IFN
cis <- xRDataLoader(RData.customised='JKscience_TS2A',
RData.location=RData.location)
ind <- which(cis$IFN_t > 0 & cis$IFN_fdr < 0.05)
df_cis <- cis[ind, c('variant','Symbol','IFN_t','IFN_fdr')]
data <- df_cis$variant

# 2) Enrichment analysis using Experimental Factor Ontology (EFO)
# Considering LD SNPs and respecting ontology tree
eTerm <- xEnricherSNPs(data, ontology="EF", include.LD="EUR",
LD.r2=0.8, ontology.algorithm="lea", RData.location=RData.location)

# 3) Barplot of enrichment results
bp <- xEnrichBarplot(eTerm, top_num="auto", displayBy="fc")
#pdf(file="enrichment_barplot.pdf", height=6, width=12, compress=TRUE)
print(bp)
#dev.off()
## modify y axis text
bp + theme(axis.text.y=element_text(size=10,color="blue"))
## modify x axis title
bp + theme(axis.title.x=element_text(color="blue"))
```

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

---

xEnrichCompare *Function to compare enrichment results using side-by-side barplots*

Description

xEnrichCompare is supposed to compare enrichment results using side-by-side barplots. It returns an object of class "ggplot".

Usage

```
xEnrichCompare(list_eTerm, displayBy = c("fc", "adjp", "fdr", "zscore",
"pvalue"), FDR.cutoff = 0.05, bar.label = TRUE, bar.label.size = 3,
wrap.width = NULL, sharings = NULL)
```

Arguments

list_eTerm	a list of "eTerm" objects
displayBy	which statistics will be used for comparison. It can be "fc" for enrichment fold change (by default), "adjp" or "fdr" for adjusted p value (or FDR), "pvalue" for p value, "zscore" for enrichment z-score
FDR.cutoff	FDR cutoff used to declare the significant terms. By default, it is set to 0.05
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
wrap.width	a positive integer specifying wrap width of name
sharings	a numeric vector specifying whether only shared terms will be displayed. For example, when comparing three groups of enrichment results, it can be set into c(2,3) to display only shared terms by any two or all three. By default, it is NULL meaning no such restriction

Value

an object of class "ggplot", but appended with a 'g' (an igraph object to represent DAG after being unionised)

Note

none

See Also

xEnricherGenes, xEnricherSNPs, xEnrichDAGplotAdv

Examples

```
## Not run:
# Load the library
library(XGR)
RData.location="~/Sites/SVN/github/bigdata"

# 1) load eQTL mapping results: cis-eQTLs significantly induced by IFN
cis <- xRDataLoader(RData.customised='JKscience_TS2A',
RData.location=RData.location)
ind <- which(cis$IFN_t > 0 & cis$IFN_fdr < 0.05)
df_cis <- cis[ind, c('variant','Symbol','IFN_t','IFN_fdr')]
data <- df_cis$variant

# 2) Enrichment analysis using Experimental Factor Ontology (EFO)
# 2a) Without considering LD SNPs and without respecting ontology tree
eTerm_noLD_noTree <- xEnricherSNPs(data, ontology="EF_disease",
include.LD=NA, ontology.algorithm="none",
RData.location=RData.location)
# 2b) Without considering LD SNPs but respecting ontology tree
eTerm_noLD_Tree <- xEnricherSNPs(data, ontology="EF_disease",
include.LD=NA, ontology.algorithm="lea", RData.location=RData.location)
# 2c) Considering LD SNPs but without respecting ontology tree
eTerm_LD_noTree <- xEnricherSNPs(data, ontology="EF_disease",
include.LD="EUR", LD.r2=0.8, ontology.algorithm="none",
RData.location=RData.location)
# 2d) Considering LD SNPs and respecting ontology tree
eTerm_LD_Tree <- xEnricherSNPs(data, ontology="EF_disease",
include.LD="EUR", LD.r2=0.8, ontology.algorithm="lea",
RData.location=RData.location)

# 3) Compare enrichment results
list_eTerm <- list(eTerm_noLD_noTree, eTerm_noLD_Tree, eTerm_LD_noTree,
eTerm_LD_Tree)
names(list_eTerm) <- c('LD (-) & Tree (-)', 'LD (-) & Tree (+)', 'LD
(+) & Tree (-)', 'LD (+) & Tree (+)')
## side-by-side comparisons
bp <- xEnrichCompare(list_eTerm, displayBy="fc")
#pdf(file="enrichment_compared.pdf", height=6, width=12, compress=TRUE)
print(bp)
#dev.off()
## modify y axis text
bp + theme(axis.text.y=element_text(size=10,color="black"))
## modify x axis title
bp + theme(axis.title.x=element_text(color="black"))
## modify fill colors
bp + scale_fill_manual(values=c("black","#888888"))
## show legend title but hide strip
bp + theme(legend.position="right", strip.text = element_blank())

# 4) DAGplot of comparative enrichment results in the context of ontology tree
xEnrichDAGplotAdv(bp, graph.node.attrs=list(fontsize=100))
```

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

`xEnrichConciser` *Function to make enrichment results conciser by removing redundant terms*

Description

`xEnrichConciser` is supposed to make enrichment 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

```
xEnrichConciser(eTerm, cutoff = c(0.9, 0.5), verbose = T)
```

Arguments

`eTerm`	an object of class "eTerm"
`cutoff`	a cutoff vector used to remove redunant 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 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 "eTerm", after redundant terms being removed.

Note

none

See Also

`xEnricherGenes`, `xEnricherSNPs`

Examples

```
## Not run:
eTerm_concise <- xEnrichConciser(eTerm)

## End(Not run)
```

xEnrichDAGplot *Function to visualise enrichment results using a direct acyclic graph (DAG)*

Description

xEnrichDAGplot is supposed to visualise enrichment results using a direct acyclic graph (DAG) with node colorings. By default, significant terms (of interest) are highlighted by box-shaped nodes, the others by ellipse nodes. It returns an object of class 'Ragraph'.

Usage

```
xEnrichDAGplot(eTerm, top_num = 10, displayBy = c("fc", "adjp", "fdr",
"zscore", "pvalue"), path.mode = c("all_paths", "shortest_paths",
"all_shortest_paths"), height = 7, width = 7, margin = rep(0.1, 4),
colormap = c("yr", "bwr", "jet", "gbr", "wyr", "br", "rainbow", "wb",
"lightyellow-orange"), ncolors = 40, zlim = NULL, colorbar = T,
colorbar.fraction = 0.1, newpage = T,
layout.orientation = c("top_bottom", "left_right", "bottom_top",
"right_left"), node.info = c("none", "term_id", "term_name", "both",
"full_term_name"), wrap.width = NULL, graph.node.attrs = NULL,
graph.edge.attrs = NULL, node.attrs = NULL)
```

Arguments

eTerm	an object of class "eTerm"
top_num	the number of the top terms (sorted according to FDR or adjusted p-values). If it is 'auto', only the significant terms (FDR < 0.05) will be displayed
displayBy	which statistics will be used for displaying. It can be "fc" for enrichment fold change (by default), "adjp" or "fdr" for adjusted p value (or FDR), "pvalue" for p value, "zscore" for enrichment z-score
path.mode	the mode of paths induced by nodes in query. 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)
height	a numeric value specifying the height of device
width	a numeric value specifying the width of device
margin	margins as units of length 4 or 1
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "lightyellow-orange" (by default), "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names

`ncolors` the number of colors specified over the colormap

`zlim` the minimum and maximum z/data values for which colors should be plotted, defaulting to the range of the finite values of z. Each of the given colors will be used to color an equispaced interval of this range. The midpoints of the intervals cover the range, so that values just outside the range will be plotted

`colorbar` logical to indicate whether to append a colorbar. If data is null, it always sets to false

`colorbar.fraction` the relative fraction of colorbar block against the device size

`newpage` logical to indicate whether to open a new page. By default, it sets to true for opening a new page

`layout.orientation` the orientation of the DAG layout. It can be one of "left_right" for the left-right layout (viewed from the DAG root point), "top_bottom" for the top-bottom layout, "bottom_top" for the bottom-top layout, and "right_left" for the right-left layout

`node.info` tells the ontology term information used to label nodes. It can be one of "none" for no node labeling, "term_id" for using Term ID, "term_name" for using Term Name (the first 15 characters), "both" for using both of Term ID and Name (the first 15 characters), and "full_term_name" for using the full Term Name

`wrap.width` a positive integer specifying wrap width of Term Name

`graph.node.attrs` a list of global node attributes. These node attributes will be changed globally. See 'Note' below for details on the attributes

`graph.edge.attrs` a list of global edge attributes. These edge attributes will be changed globally. See 'Note' below for details on the attributes

`node.attrs` a list of local edge attributes. These node attributes will be changed locally; as such, for each attribute, the input value must be a named vector (i.e. using Term ID as names). See 'Note' below for details on the attributes

Value

An object of class 'Ragraph'

Note

A list of global node attributes used in "graph.node.attrs":

- "shape": the shape of the node: "circle", "rectangle", "rect", "box" and "ellipse"
- "fixedsize": the logical to use only width and height attributes. By default, it sets to true for not expanding for the width of the label
- "fillcolor": the background color of the node
- "color": the color for the node, corresponding to the outside edge of the node
- "fontcolor": the color for the node text/labelings
- "fontsize": the font size for the node text/labelings

- "height": the height (in inches) of the node: 0.5 by default
- "width": the width (in inches) of the node: 0.75 by default
- "style": the line style for the node: "solid", "dashed", "dotted", "invis" and "bold"

A list of global edge attributes used in "graph.edge.attrs":

- "color": the color of the edge: gray by default
- "weight": the weight of the edge: 1 by default
- "style": the line style for the edge: "solid", "dashed", "dotted", "invis" and "bold"

A list of local node attributes used in "node.attrs" (only those named Term IDs will be changed locally!):

- "label": a named vector specifying the node text/labelings
- "shape": a named vector specifying the shape of the node: "circle", "rectangle", "rect", "box" and "ellipse"
- "fixedsize": a named vector specifying whether it sets to true for not expanding for the width of the label
- "fillcolor": a named vector specifying the background color of the node
- "color": a named vector specifying the color for the node, corresponding to the outside edge of the node
- "fontcolor": a named vector specifying the color for the node text/labelings
- "fontsize": a named vector specifying the font size for the node text/labelings
- "height": a named vector specifying the height (in inches) of the node: 0.5 by default
- "width": a named vector specifying the width (in inches) of the node: 0.75 by default
- "style": a named vector specifying the line style for the node: "solid", "dashed", "dotted", "invis" and "bold"

See Also

xEnricherGenes, xEnricherSNPs, xEnrichViewer

Examples

```
## Not run:
# Load the library
library(XGR)
RData.location="~/Sites/SVN/github/bigdata"

# 1) load eQTL mapping results: cis-eQTLs significantly induced by IFN
cis <- xRDataLoader(RData.customised='JKscience_TS2A',
RData.location=RData.location)
ind <- which(cis$IFN_t > 0 & cis$IFN_fdr < 0.05)
df_cis <- cis[ind, c('variant','Symbol','IFN_t','IFN_fdr')]
data <- df_cis$variant

# 2) Enrichment analysis using Experimental Factor Ontology (EFO)
```

```
# Considering LD SNPs and respecting ontology tree
eTerm <- xEnricherSNPs(data, ontology="EF", include.LD="EUR",
LD.r2=0.8, ontology.algorithm="lea", RData.location=RData.location)

# 3) DAG plot of enrichment results
agDAG <- xEnrichDAGplot(eTerm, top_num="auto", displayBy="fc",
node.info=c("full_term_name"))
## modify node labels
xEnrichDAGplot(eTerm, top_num="auto", displayBy="fc",
node.info=c("full_term_name"),
graph.node.attrs=list(fontsize=25,fontcolor="blue",color="transparent"))
## modify node shapes
xEnrichDAGplot(eTerm, top_num="auto", displayBy="fc",
node.info=c("full_term_name"),
graph.node.attrs=list(fixedsize=FALSE,shape=c("ellipse","box","circle","plaintext")[2]))
## further modify edge color
xEnrichDAGplot(eTerm, top_num="auto", displayBy="fc",
node.info=c("full_term_name"), graph.node.attrs=list(fontsize=25),
graph.edge.attrs=list(color="black"))

# 4) hide labels for ellipse nodes
library(Rgraphviz)
name_nodes <- sapply(AgNode(agDAG), name)
shape_nodes <- sapply(AgNode(agDAG), shape)
names(shape_nodes) <- name_nodes
ind <- which(shape_nodes=='ellipse')
label_nodes <- rep('', length(ind))
names(label_nodes) <- name_nodes[ind]
xEnrichDAGplot(eTerm, top_num="auto", displayBy="fc",
node.info=c("full_term_name"), node.attrs=list(label=label_nodes,
shape=shape_nodes))


## End(Not run)
```

---

xEnrichDAGplotAdv *Function to visualise comparative enrichment results using a direct acyclic graph (DAG)*

Description

xEnrichDAGplotAdv is supposed to visualise the comparative enrichment results (see the function xEnrichCompare)) using a direct acyclic graph (DAG). Nodes/terms can be colored according to how many times being called significant. If two enrichment results are compared, node names are prefixed with the form of 'x1-x2', where x1 is for result 1 and x2 for result 2 (the value for x1 or x2 can be '0' for being insignificant, and '1' for being significant). It takes input an 'ggplot' object (with two componets already appended 'g' and 'data'), and returns an object of class 'Ragraph'.

Usage

```
xEnrichDAGplotAdv(ggplot, displayBy = c("nSig", "none"),
path.mode = c("all_paths", "shortest_paths", "all_shortest_paths"),
height = 7, width = 7, margin = rep(0.1, 4),
colormap = c("white-lightcyan-cyan", "yr", "bwr", "jet", "gbr", "wyr",
"br",
"rainbow", "wb", "lightyellow-orange"), ncolors = 40, zlim = NULL,
colorbar = T, colorbar.fraction = 0.1, newpage = T,
layout.orientation = c("left_right", "top_bottom", "bottom_top",
"right_left"), node.info = c("term_name", "term_id", "none"),
wrap.width = NULL, graph.node.attrs = NULL, graph.edge.attrs = NULL,
node.attrs = NULL)
```

Arguments

ggplot	an object "ggplot" (resulting from xEnrichCompare)
displayBy	which statistics will be used for displaying. It can be "nSig" for how many times being called significant (by default), "none" for no color-coding on nodes/terms
path.mode	the mode of paths induced by nodes in query. 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)
height	a numeric value specifying the height of device
width	a numeric value specifying the width of device
margin	margins as units of length 4 or 1
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "lightyellow-orange" (by default), "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
ncolors	the number of colors specified over the colormap
zlim	the minimum and maximum z/data values for which colors should be plotted, defaulting to the range of the finite values of z. Each of the given colors will be used to color an equispaced interval of this range. The midpoints of the intervals cover the range, so that values just outside the range will be plotted
colorbar	logical to indicate whether to append a colorbar. If data is null, it always sets to false
colorbar.fraction	the relative fraction of colorbar block against the device size
newpage	logical to indicate whether to open a new page. By default, it sets to true for opening a new page

`layout.orientation`
: the orientation of the DAG layout. It can be one of "left_right" for the left-right layout (viewed from the DAG root point), "top_bottom" for the top-bottom layout, "bottom_top" for the bottom-top layout, and "right_left" for the right-left layout

`node.info`
: tells the ontology term information used to label nodes. It can be "term_id" for using Term ID, "term_name" for using Term Name, 'none' for no labellings

`wrap.width`
: a positive integer specifying wrap width of Term Name

`graph.node.attrs`
: a list of global node attributes. These node attributes will be changed globally. See 'Note' below for details on the attributes

`graph.edge.attrs`
: a list of global edge attributes. These edge attributes will be changed globally. See 'Note' below for details on the attributes

`node.attrs`
: a list of local edge attributes. These node attributes will be changed locally; as such, for each attribute, the input value must be a named vector (i.e. using Term ID as names). See 'Note' below for details on the attributes

Value

An object of class 'Ragraph'

Note

A list of global node attributes used in "graph.node.attrs":

- "shape": the shape of the node: "circle", "rectangle", "rect", "box" and "ellipse"
- "fixedsize": the logical to use only width and height attributes. By default, it sets to true for not expanding for the width of the label
- "fillcolor": the background color of the node
- "color": the color for the node, corresponding to the outside edge of the node
- "fontcolor": the color for the node text/labelings
- "fontsize": the font size for the node text/labelings
- "height": the height (in inches) of the node: 0.5 by default
- "width": the width (in inches) of the node: 0.75 by default
- "style": the line style for the node: "solid", "dashed", "dotted", "invis" and "bold"

A list of global edge attributes used in "graph.edge.attrs":

- "color": the color of the edge: gray by default
- "weight": the weight of the edge: 1 by default
- "style": the line style for the edge: "solid", "dashed", "dotted", "invis" and "bold"

A list of local node attributes used in "node.attrs" (only those named Term IDs will be changed locally!):

- "label": a named vector specifying the node text/labelings

- "shape": a named vector specifying the shape of the node: "circle", "rectangle", "rect", "box" and "ellipse"
- "fixedsize": a named vector specifying whether it sets to true for not expanding for the width of the label
- "fillcolor": a named vector specifying the background color of the node
- "color": a named vector specifying the color for the node, corresponding to the outside edge of the node
- "fontcolor": a named vector specifying the color for the node text/labelings
- "fontsize": a named vector specifying the font size for the node text/labelings
- "height": a named vector specifying the height (in inches) of the node: 0.5 by default
- "width": a named vector specifying the width (in inches) of the node: 0.75 by default
- "style": a named vector specifying the line style for the node: "solid", "dashed", "dotted", "invis" and "bold"

See Also

xEnrichCompare

Examples

```
## Not run:
# Load the library
library(XGR)
RData.location="~/Sites/SVN/github/bigdata"

# 1) load eQTL mapping results: cis-eQTLs significantly induced by IFN
cis <- xRDataLoader(RData.customised='JKscience_TS2A',
RData.location=RData.location)
ind <- which(cis$IFN_t > 0 & cis$IFN_fdr < 0.05)
df_cis <- cis[ind, c('variant','Symbol','IFN_t','IFN_fdr')]
data <- df_cis$variant

# 2) Enrichment analysis using Experimental Factor Ontology (EFO)
# 2a) Without considering LD SNPs and without respecting ontology tree
eTerm_noLD_noTree <- xEnricherSNPs(data, ontology="EF_disease",
include.LD=NA, ontology.algorithm="none",
RData.location=RData.location)
# 2b) Without considering LD SNPs but respecting ontology tree
eTerm_noLD_Tree <- xEnricherSNPs(data, ontology="EF_disease",
include.LD=NA, ontology.algorithm="lea", RData.location=RData.location)
# 2c) Considering LD SNPs but without respecting ontology tree
eTerm_LD_noTree <- xEnricherSNPs(data, ontology="EF_disease",
include.LD="EUR", LD.r2=0.8, ontology.algorithm="none",
RData.location=RData.location)
# 2d) Considering LD SNPs and respecting ontology tree
eTerm_LD_Tree <- xEnricherSNPs(data, ontology="EF_disease",
include.LD="EUR", LD.r2=0.8, ontology.algorithm="lea",
RData.location=RData.location)
```

```
# 3) Compare enrichment results
list_eTerm <- list(eTerm_noLD_noTree, eTerm_noLD_Tree, eTerm_LD_noTree,
eTerm_LD_Tree)
names(list_eTerm) <- c('LD (-) & Tree (-)', 'LD (-) & Tree (+)', 'LD
(+) & Tree (-)', 'LD (+) & Tree (+)')
## side-by-side comparisons
bp <- xEnrichCompare(list_eTerm, displayBy="fc")
#pdf(file="enrichment_compared.pdf", height=6, width=12, compress=TRUE)
print(bp)
#dev.off()

# 4) DAGplot of comparative enrichment results in the context of ontology tree
xEnrichDAGplotAdv(bp, graph.node.attrs=list(fontsize=100))

## End(Not run)
```

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

Description

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

Usage

```
xEnricher(data, annotation, g, background = NULL, size.range = c(10,
2000),
min.overlap = 3, which.distance = NULL, test = c("hypergeo", "fisher",
"binomial"), p.adjust.method = c("BH", "BY", "bonferroni", "holm",
"hochberg", "hommel"), ontology.algorithm = c("none", "pc", "elim",
"lea"),
elim.pvalue = 0.01, lea.depth = 2, path.mode = c("all_paths",
"shortest_paths", "all_shortest_paths"), true.path.rule = TRUE,
verbose = T)
```

Arguments

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

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

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

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

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

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

`test` the 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

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

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

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

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

`path.mode` the mode of paths induced by vertices/nodes with input annotation data. It can be "all_paths" for all possible paths to the root, "shortest_paths" for only one path

	to the root (for each node in query), "all_shortest_paths" for all shortest paths to the root (i.e. for each node, find all shortest paths with the equal lengths)
`true.path.rule`	logical to indicate whether the true-path rule should be applied to propagate annotations. By default, it sets to true
`verbose`	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

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

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

Note

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

- "none": does not consider the ontology hierarchy at all.
- "lea": estimates the significance of a term in terms of the significance of its children at the maximum depth (e.g. 2). Precisely, once snps/genes are already annotated to any children terms with a more signficance than itself, then all these snps/genes are eliminated from the use for the recalculation of the signifance at that term. The final p-values takes the maximum of the original p-value and the recalculated p-value.

- "elim": estimates the significance of a term in terms of the significance of its all children. Precisely, once snps/genes are already annotated to a signficantly enriched term under the cutoff of e.g. pvalue<1e-2, all these snps/genes are eliminated from the ancestors of that term).
- "pc": requires the significance of a term not only using the whole snps/genes as background but also using snps/genes annotated to all its direct parents/ancestors as background. The final p-value takes the maximum of both p-values in these two calculations.
- "Notes": the order of the number of significant terms is: "none" > "lea" > "elim" > "pc".

See Also

xDAGanno, xEnricherGenes, xEnricherSNPs

Examples

```
## Not run:
# 1) SNP-based enrichment analysis using GWAS Catalog traits (mapped to EF)
# 1a) ig.EF (an object of class "igraph" storing as a directed graph)
g <- xRDataLoader('ig.EF')

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

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

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

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

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

# 1f') save enrichment results to the file called 'EF_enrichments.txt'
res <- xEnrichViewer(eTerm, top_num=length(eTerm$adjp), sortBy="adjp",
details=TRUE)
output <- data.frame(term=rownames(res), res)
utils::write.table(output, file="EF_enrichments.txt", sep="\t",
row.names=FALSE)

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

# 1h) visualise the top 10 significant terms in the ontology hierarchy
# color-code terms according to the adjust p-values (taking the form of 10-based negative logarithm)
```

```
xEnrichDAGplot(eTerm, top_num=10, displayBy="adjp",
node.info=c("full_term_name"))
# color-code terms according to the z-scores
xEnrichDAGplot(eTerm, top_num=10, displayBy="zscore",
node.info=c("full_term_name"))

## End(Not run)
```

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

Description

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

Usage

```
xEnricherGenes(data, background = NULL, ontology = c("GOBP", "GOMF",
"GOCC",
"PS", "PS2", "SF", "DO", "HPPA", "HPMI", "HPCM", "HPMA", "MP",
"MsigdbH",
"MsigdbC1", "MsigdbC2CGP", "MsigdbC2CPall", "MsigdbC2CP",
"MsigdbC2KEGG",
"MsigdbC2REACTOME", "MsigdbC2BIOCARTA", "MsigdbC3TFT", "MsigdbC3MIR",
"MsigdbC4CGN", "MsigdbC4CM", "MsigdbC5BP", "MsigdbC5MF", "MsigdbC5CC",
"MsigdbC6", "MsigdbC7", "DGIdb"), size.range = c(10, 2000),
min.overlap = 3, which.distance = NULL, test = c("hypergeo", "fisher",
"binomial"), 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 = F, verbose =
T,
RData.location =
"https://github.com/hfang-bristol/RDataCentre/blob/master/Portal")
```

Arguments

`data`	an input vector. It contains a list of Gene Symbols of interest
`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. 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 domain superfamily assignments, "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, and Drug-Gene Interaction database (DGIdb) for drugable categories, and the molecular signatures database (Msigdb, including "MsigdbH", "MsigdbC1", "MsigdbC2CGP", "MsigdbC2CPall", "MsigdbC2CP", "MsigdbC2KEGG", "MsigdbC2REACTOME", "MsigdbC2BIOCARTA", "MsigdbC3TFT", "MsigdbC3MIR", "MsigdbC4CGN", "MsigdbC4CM", "MsigdbC5BP", "MsigdbC5MF", "MsigdbC5CC", "MsigdbC6", "MsigdbC7")

`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

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

Note

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

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

See Also

xRDataLoader, xEnricher

Examples

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

# Gene-based enrichment analysis using Mammalian Phenotype Ontology (MP)
# a) provide the input Genes of interest (eg 100 randomly chosen human genes)
## load human genes
org.Hs.eg <- xRDataLoader(RData='org.Hs.eg')
data <- as.character(sample(org.Hs.eg$gene_info$Symbol, 100))
data

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

# b) perform enrichment analysis
eTerm <- xEnricherGenes(data=data, ontology="MP")

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

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

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

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

## End(Not run)
```

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

Description

`xEnricherSNPs` is supposed to conduct enrichment analysis given the input data and the ontology in query. It returns an object of class "eTerm". Enrichment analysis is based on either Fisher's exact test or Hypergeometric test. The test can respect the hierarchy of the ontology. Now it supports enrichment analysis for SNPs using GWAS Catalog traits mapped to Experimental Factor Ontology. If required, additional SNPs that are in linkage disequilibrium (LD) with input SNPs are also be used for test.

Usage

```
xEnricherSNPs(data, background = NULL, ontology = c("EF", "EF_disease",
"EF_phenotype", "EF_bp"), include.LD = NA, LD.r2 = 0.8,
size.range = c(10, 2000), min.overlap = 3, which.distance = NULL,
test = c("hypergeo", "fisher", "binomial"), 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 = T, verbose = T,
RData.location =
"https://github.com/hfang-bristol/RDataCentre/blob/master/Portal")
```

Arguments

`data`	an input vector. It contains a list of SNPs of interest
`background`	a background vector. It contains a list of SNPs as the test background. If NULL, by default all annotatable are used as background

`ontology` the ontology supported currently. Now it is only "EF" for Experimental Factor Ontology (used to annotate GWAS Catalog SNPs). However, there are several subparts of this ontology to choose: 'EF_disease' for the subpart under the term 'disease' (EFO:0000408), 'EF_phenotype' for the subpart under the term 'phenotype' (EFO:0000651), 'EF_bp' for the subpart under the term 'biological process' (GO:0008150)

`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.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

`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

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

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

elim.pvalue	the parameter only used when "ontology.algorithm" is "elim". It is used to control how to declare a signficantly enriched term (and subsequently all genes in this term are eliminated from all its ancestors)
lea.depth	the parameter only used when "ontology.algorithm" is "lea". It is used to control how many maximum depth is used to consider the children of a term (and subsequently all genes in these children term are eliminated from the use for the recalculation of the signifance at this term)
path.mode	the mode of paths induced by vertices/nodes with input annotation data. It can be "all_paths" for all possible paths to the root, "shortest_paths" for only one path to the root (for each node in query), "all_shortest_paths" for all shortest paths to the root (i.e. for each node, find all shortest paths with the equal lengths)
true.path.rule	logical to indicate whether the true-path rule should be applied to propagate annotations. By default, it sets to true
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to 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"
- g: an igraph object to represent DAG
- data: a vector containing input data in consideration. It is not always the same as the input data as only those mappable are retained
- background: a vector containing the background data. It is not always the same as the input data as only those mappable are retained
- overlap: a list of overlapped snp/gene sets, each storing snps overlapped between a snp/gene set and the given input data (i.e. the snps of interest). Always, gene sets are identified by "id"
- fc: a vector containing fold changes
- zscore: a vector containing z-scores
- pvalue: a vector containing p-values
- adjp: a vector containing adjusted p-values. It is the p value but after being adjusted for multiple comparisons
- cross: a matrix of nTerm X nTerm, with an on-diagnal cell for the overlapped-members observed in an individaul term, and off-diagnal cell for the overlapped-members shared betwene two terms
- call: the call that produced this result

Note

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

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

See Also

xRDataLoader, xEnricher

Examples

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

# SNP-based enrichment analysis using GWAS Catalog traits (mapped to EF)
# a) provide the input SNPs of interest (eg 'EFO:0002690' for 'systemic lupus erythematosus')
## load GWAS SNPs annotated by EF (an object of class "dgCMatrix" storing a spare matrix)
anno <- xRDataLoader(RData='GWAS2EF')
ind <- which(colnames(anno)=='EFO:0002690')
data <- rownames(anno)[anno[,ind]==1]
data

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

# b) perform enrichment analysis
eTerm <- xEnricherSNPs(data=data, ontology="EF",
path.mode=c("all_paths"))

# b') optionally, enrichment analysis for input SNPs plus their LD SNPs
## LD based on European population (EUR) with r2>=0.8
#eTerm <- xEnricherSNPs(data=data, include.LD="EUR", LD.r2=0.8)

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

# d) save enrichment results to the file called 'EF_enrichments.txt'
```

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

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

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

## End(Not run)
```

`xEnricherYours` *Function to conduct enrichment analysis given YOUR own input data*

Description

`xEnricherYours` is supposed to conduct enrichment analysis given the input data and the ontology in query. It returns an object of class "eTerm". Enrichment analysis is based on either Fisher's exact test or Hypergeometric test.

Usage

```
xEnricherYours(data.file, annotation.file, background.file = NULL,
size.range = c(10, 2000), min.overlap = 3, test = c("hypergeo",
"fisher", "binomial"), p.adjust.method = c("BH", "BY", "bonferroni",
"holm",
"hochberg", "hommel"), verbose = T)
```

Arguments

`data.file`	an input data file, containing a list of entities (e.g. genes or SNPs) to test. The entities can be anything, for example, in this file http://dcgor.r-forge.r-project.org/data/InterPro/InterPro.txt, the entities are InterPro domains (InterPro). As seen in this example, entries in the first column must be domains. If the file also contains other columns, these additional columns will be ignored. Alternatively, the data.file can be a matrix or data frame, assuming that input file has been read. Note: the file should use the tab delimiter as the field separator between columns

`annotation.file`
: an input annotation file containing annotations between entities and ontology terms. For example, a file containing annotations between InterPro domains and GO Molecular Function (GOMF) terms can be found in http://dcgor.r-forge.r-project.org/data/InterPro/Domain2GOMF.txt. As seen in this example, the input file must contain two columns: 1st column for domains, 2nd column for ontology terms. If there are additional columns, these columns will be ignored. Alternatively, the annotation.file can be a matrix or data frame, assuming that input file has been read. Note: the file should use the tab delimiter as the field separator between columns

`background.file`
: an input background file containing a list of entities as the test background. The file format is the same as 'data.file'. By default, it is NULL meaning all annotatable entities (i.e. those entities in 'annotation.file') are used as background

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

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

`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

`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

`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 "eTerm", a list with following components:

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

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

Note

None

See Also

xEnricher

Examples

```
## Not run:
# Load the library
library(XGR)
library(igraph)

# Enrichment analysis using your own data
# a) provide your own data (eg InterPro domains and their annotations by GO terms)
## All InterPro domains
input.file <-
"http://dcgor.r-forge.r-project.org/data/InterPro/InterPro.txt"
data <- utils::read.delim(input.file, header=F, row.names=NULL,
stringsAsFactors=F)[,1]
## provide the input domains of interest (eg 100 randomly chosen domains)
data.file <- sample(data, 100)
## InterPro domains annotated by GO Molecular Function (GOMF) terms
annotation.file <-
"http://dcgor.r-forge.r-project.org/data/InterPro/Domain2GOMF.txt"

# b) perform enrichment analysis
```

```
eTerm <- xEnricherYours(data.file=data.file,
annotation.file=annotation.file)

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

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

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

## End(Not run)

# Using ImmunoBase SNPs and associations/annotations with disease traits
## get ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase')
## get disease associated variants/SNPs
variants_list <- lapply(ImmunoBase, function(x)
cbind(SNP=names(x$variants),
Disease=rep(x$disease,length(x$variants))))
## extract annotations as a data frame: Variant Disease_Name
annotation.file <- do.call(rbind, variants_list)
head(annotation.file)
## provide the input SNPs of interest
## for example, cis-eQTLs induced by interferon gamma
cis <- xRDataLoader(RData.customised='JKscience_TS2A')
data.file <- matrix(cis[which(cis$IFN_t>0),c('variant')], ncol=1)
# perform enrichment analysis
eTerm <- xEnricherYours(data.file=data.file,
annotation.file=annotation.file)
# view enrichment results for the top significant terms
xEnrichViewer(eTerm)
# barplot of significant enrichment results
bp <- xEnrichBarplot(eTerm, top_num="auto", displayBy="adjp")
print(bp)
```

`xEnrichNetplot` *Function to visualise enrichment results using different network layouts*

Description

`xEnrichNetplot` is supposed to visualise enrichment results using different network layouts. Also supported is to visualise the comparative enrichment results (see the function `xEnrichCompare`)) with nodes/terms colored according to how many times being called significant. It returns an object of class 'igraph'.

Usage

```
xEnrichNetplot(eTerm, top_num = 10, displayBy = c("fc", "adjp", "fdr",
"zscore", "pvalue"), path.mode = c("all_paths", "shortest_paths",
"all_shortest_paths"), node.info = c("none", "term_id", "term_name",
"both",
"full_term_name"), wrap.width = 15, colormap = c("yr", "jet", "gbr",
"wyr", "br", "bwr", "rainbow", "wb"), ncolors = 40, zlim = NULL,
colorbar = T, newpage = T, glayout = layout_as_tree,
vertex.frame.color = NA, vertex.size = NULL, vertex.color = NULL,
vertex.shape = NULL, vertex.label = NULL, vertex.label.cex = NULL,
vertex.label.dist = 0.3, vertex.label.color = "blue",
edge.arrow.size = 0.3, ...)
```

Arguments

eTerm	an object of class "eTerm" or an object "ggplot" (resulting from xEnrichCompare)
top_num	the number of the top terms (sorted according to FDR or adjusted p-values). If it is 'auto', only the significant terms (FDR < 0.05) will be displayed
displayBy	which statistics will be used for displaying. It can be "fc" for enrichment fold change (by default), "adjp" or "fdr" for adjusted p value (or FDR), "pvalue" for p value, "zscore" for enrichment z-score
path.mode	the mode of paths induced by nodes in query. 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)
node.info	tells the ontology term information used to label nodes. It can be one of "none" for no node labeling, "term_id" for using Term ID, "term_name" for using Term Name, "both" for using both of Term ID and Name (the first 15 characters), and "full_term_name" for using the full Term Name
wrap.width	a positive integer specifying wrap width of Term Name. By default, first 15 characters
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
ncolors	the number of colors specified over the colormap
zlim	the minimum and maximum z/patttern values for which colors should be plotted, defaulting to the range of the finite values of z. Each of the given colors will be used to color an equispaced interval of this range. The midpoints of the intervals cover the range, so that values just outside the range will be plotted
colorbar	logical to indicate whether to append a colorbar. If pattern is null, it always sets to false

`newpage` logical to indicate whether to open a new page. By default, it sets to true for opening a new page

`glayout` either a function or a numeric matrix configuring how the vertices will be placed on the plot. If layout is a function, this function will be called with the graph as the single parameter to determine the actual coordinates. This function can be one of "layout_nicely" (previously "layout.auto"), "layout_randomly" (previously "layout.random"), "layout_in_circle" (previously "layout.circle"), "layout_on_sphere" (previously "layout.sphere"), "layout_with_fr" (previously "layout.fruchterman.reingold"), "layout_with_kk" (previously "layout.kamada.kawai"), "layout_as_tree" (previously "layout.reingold.tilford"), "layout_with_lgl" (previously "layout.lgl"), "layout_with_graphopt" (previously "layout.graphopt"), "layout_with_sugiyama" (previously "layout.kamada.kawai"), "layout_with_dh" (previously "layout.davidson.harel"), "layout_with_drl" (previously "layout.drl"), "layout_with_gem" (previously "layout.gem"), "layout_with_mds". A full explanation of these layouts can be found in http://igraph.org/r/doc/layout_nicely.html

`vertex.frame.color` the color of the frame of the vertices. If it is NA, then there is no frame

`vertex.size` the size of each vertex. If it is a vector, each vertex may differ in size

`vertex.color` the fill color of the vertices. If it is NA, then there is no fill color. If the pattern is given, this setup will be ignored

`vertex.shape` the shape of each vertex. It can be one of "circle", "square", "csquare", "rectangle", "crectangle", "vrectangle", "pie" (http://igraph.org/r/doc/vertex.shape.pie.html), "sphere", and "none". If it sets to NULL, these vertices with negative will be "csquare" and the rest "circle".

`vertex.label` the label of the vertices. If it is NA, then there is no label. The default vertex labels are the name attribute of the nodes

`vertex.label.cex` the font size of vertex labels.

`vertex.label.dist` the distance of the label from the center of the vertex. If it is 0 then the label is centered on the vertex. If it is 1 then the label is displayed beside the vertex.

`vertex.label.color` the color of vertex labels.

`edge.arrow.size` the size of the arrows for the directed edge. The default value is 1.

`...` additional graphic parameters. See http://igraph.org/r/doc/plot.common.html for the complete list.

Value

an igraph object to represent DAG, appended with a node attribute called 'enrichment'

Note

none

See Also

xEnricherGenes, xEnricherSNPs, xEnrichViewer

Examples

```
## Not run:
# Load the library
library(XGR)
RData.location="~/Sites/SVN/github/bigdata"

# 1) load eQTL mapping results: cis-eQTLs significantly induced by IFN
cis <- xRDataLoader(RData.customised='JKscience_TS2A',
RData.location=RData.location)
ind <- which(cis$IFN_t > 0 & cis$IFN_fdr < 0.05)
df_cis <- cis[ind, c('variant','Symbol','IFN_t','IFN_fdr')]
data <- df_cis$variant

# 2) Enrichment analysis using Experimental Factor Ontology (EFO)
# Considering LD SNPs and respecting ontology tree
eTerm <- xEnricherSNPs(data, ontology="EF", include.LD="EUR",
LD.r2=0.8, ontology.algorithm="lea", RData.location=RData.location)

# 3) Net plot of enrichment results
subg <- xEnrichNetplot(eTerm, top_num="auto", displayBy="fc",
node.info=c("none"), vertex.label=NA, wrap.width=30)

## End(Not run)
```

---

xEnrichViewer *Function to view enrichment results*

Description

xEnrichViewer is supposed to view results of enrichment analysis.

Usage

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

Arguments

eTerm	an object of class "eTerm"
top_num	the number of the top terms (sorted according to 'sortBy' below) will be viewed
sortBy	which statistics will be used for sorting and viewing gene sets (terms). It can be "adjp" or "fdr" for adjusted p value (FDR), "pvalue" for p value, "zscore" for enrichment z-score, "fc" for enrichment fold change, "nAnno" for the number

	of sets (terms), "nOverlap" for the number in overlaps, and "none" for ordering according to ID of terms
`decreasing`	logical to indicate whether to sort in a decreasing order. If it is null, it would be true for "zscore", "nAnno" or "nOverlap"; otherwise it would be false
`details`	logical to indicate whether the detailed information of gene sets (terms) is also viewed. By default, it sets to false for no inclusion

Value

a data frame with following components:

- `id`: term ID; as rownames
- `name`: term name
- `nAnno`: number in members annotated by a term
- `nOverlap`: number in overlaps
- `fc`: enrichment fold changes
- `zscore`: enrichment z-score
- `pvalue`: nominal p value
- `adjp`: adjusted p value (FDR)
- `distance`: term distance; optional, it is only appended when "details" is true
- `members`: members (represented as Gene Symbols) in overlaps; optional, it is only appended when "details" is true

Note

none

See Also

`xEnricherGenes`, `xEnricherSNPs`

Examples

```
## Not run:
xEnrichViewer(eTerm)

## End(Not run)
```

xFunArgs *Function to assign (and evaluate) arguments with default values for a given function*

Description

xFunArgs is supposed to assign (and evaluate) arguments with default values for a given function.

Usage

```
xFunArgs(fun, action = F, verbose = TRUE)
```

Arguments

fun	character specifying the name of the function
action	logical to indicate whether the function will act as it should be (with assigned values in the current environment). By default, it sets to FALSE
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to TRUE for display

Value

a list containing arguments and their default values

Note

This function is potentially useful when debugging as it frees developers from specifying default values for all arguments except those arguments of interest

See Also

xFunArgs

Examples

```
xFunArgs(fun="xRDataLoader")
```

xGRsampling *Function to generate random samples for data genomic regions from background genomic regions*

Description

xGRsampling is supposed to randomly generate samples for data genomic regions from background genomic regions. To do so, we first identify background islands, that is, non-overlapping regions. Then, we keep only parts of data genomic regions that fall into these background islands. For each kept genomic region, a randomised region of the same length is sampled from the corresponding background islands. If required, the randomised region can be restricted to be no more than (eg 10000bp) away from data genomic regions.

Usage

```
xGRsampling(GR.data, GR.background, num.samples = 100, gap.max = 50000,
max.distance = NULL, verbose = T,
RData.location =
"https://github.com/hfang-bristol/RDataCentre/blob/master/Portal")
```

Arguments

GR.data	an input data GR object, containing a set of genomic regions based on which to generate a null distribution
GR.background	an input background GR object, containing a set of genomic regions to randomly sample from. It can be a GR list object or a list of GR objects
num.samples	the number of samples randomly generated
gap.max	the maximum distance of background islands to be considered away from data regions. Only background islands no far way from this distance will be considered. For example, if it is 0, meaning that only background islands that overlapp with genomic regions will be considered. By default, it is 50000
max.distance	the maximum distance away from data regions that is allowed when generating random samples. By default, it is NULl meaning no such restriction
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

a list of GR ojects, each containing an GR oject storing a sample.

See Also

xGRsampling

Examples

```
## Not run:
# Load the library
library(XGR)
RData.location="~/Sites/SVN/github/bigdata"

# Enrichment analysis for GWAS SNPs from ImmunoBase
# a) provide input data GR object storing GWAS SNPs
dbSNP_GWAS <- xRDataLoader(RData.customised='dbSNP_GWAS',
RData.location=RData.location)

# b) provide backgorund data GR object storing FANTOM5 cell-specific enhancers
FANTOM5_Enhancer_Cell <-
xRDataLoader(RData.customised='FANTOM5_Enhancer_Cell',
RData.location=RData.location)

# c) generate random samples as a list of GR objects
sGR_List <- xGRsampling(GR.data=dbSNP_GWAS,
GR.background=FANTOM5_Enhancer_Cell, num.samples=1000,
RData.location=RData.location)

## End(Not run)
```

---

xGRviaGeneAnno *Function to conduct region-based enrichment analysis using nearby gene annotations*

Description

xGRviaGeneAnno is supposed to conduct region-based enrichment analysis for the input genomic region data (genome build h19), using nearby gene annotations. To do so, nearby genes are first defined within the maximum gap between genomic regions and gene location. Enrichment analysis is based on either Fisher's exact test or Hypergeometric test for estimating the significance of overlapped nearby genes. Test background can be provided; by default, the annotatable genes will be used.

Usage

```
xGRviaGeneAnno(data.file, background.file = NULL,
format.file = c("data.frame", "bed", "chr:start-end", "GRanges"),
build.conversion = c(NA, "hg38.to.hg19", "hg18.to.hg19"), gap.max = 0,
GR.Gene = c("UCSC_knownGene", "UCSC_knownCanonical"), ontology =
c("GOBP",
"GOMF", "GOCC", "PS", "PS2", "SF", "DO", "HPPA", "HPMI", "HPCM",
"HPMA", "MP",
"MsigdbH", "MsigdbC1", "MsigdbC2CGP", "MsigdbC2CPall", "MsigdbC2CP",
"MsigdbC2KEGG", "MsigdbC2REACTOME", "MsigdbC2BIOCARTA", "MsigdbC3TFT",
"MsigdbC3MIR", "MsigdbC4CGN", "MsigdbC4CM", "MsigdbC5BP", "MsigdbC5MF",
```

```
"MsigdbC5CC", "MsigdbC6", "MsigdbC7", "DGIdb"), size.range = c(10,
2000),
min.overlap = 3, which.distance = NULL, test = c("hypergeo", "fisher",
"binomial"), 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 = F, verbose =
T,
RData.location =
"https://github.com/hfang-bristol/RDataCentre/blob/master/Portal")
```

Arguments

data.file	an input data file, containing a list of genomic regions to test. If the input file is formatted as a 'data.frame' (specified by the parameter 'format.file' below), the first three columns correspond to the chromosome (1st column), the starting chromosome position (2nd column), and the ending chromosome position (3rd column). If the format is indicated as 'bed' (browser extensible data), the same as 'data.frame' format but the position is 0-based offset from chromomose position. If the genomic regions provided are not ranged but only the single position, the ending chromosome position (3rd column) is allowed not to be provided. If the format is indicated as "chr:start-end", instead of using the first 3 columns, only the first column will be used and processed. If the file also contains other columns, these additional columns will be ignored. Alternatively, the input file can be the content itself assuming that input file has been read. Note: the file should use the tab delimiter as the field separator between columns
background.file	an input background file containing a list of genomic regions as the test background. The file format is the same as 'data.file'. By default, it is NULL meaning all annotatable bases (ig non-redundant bases covered by 'annotation.file') are used as background. However, if only one annotation (eg only a transcription factor) is provided in 'annotation.file', the background must be provided.
format.file	the format for input files. It can be one of "data.frame", "chr:start-end", "bed" or "GRanges"
build.conversion	the conversion from one genome build to another. The conversions supported are "hg38.to.hg19" and "hg18.to.hg19". By default it is NA (no need to do so).
gap.max	the maximum distance to nearby genes. Only those genes no far way from this distance will be considered as nearby genes. By default, it is 0 meaning that nearby genes are those overlapping with genomic regions
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"

`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 domain superfamily assignments, "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, and Drug-Gene Interaction database (DGIdb) for drugable categories, and the molecular signatures database (Msigdb, including "MsigdbH", "MsigdbC1", "MsigdbC2CGP", "MsigdbC2CPall", "MsigdbC2CP", "MsigdbC2KEGG", "MsigdbC2REACTOME", "MsigdbC2BIOCARTA", "MsigdbC3TFT", "MsigdbC3MIR", "MsigdbC4CGN", "MsigdbC4CM", "MsigdbC5BP", "MsigdbC5MF", "MsigdbC5CC", "MsigdbC6", "MsigdbC7")

`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

`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"
- `fc`: a vector containing fold changes
- `zscore`: a vector containing z-scores
- `pvalue`: a vector containing p-values
- `adjp`: a vector containing adjusted p-values. It is the p value but after being adjusted for multiple comparisons
- `cross`: a matrix of nTerm X nTerm, with an on-diagnal cell for the overlapped-members observed in an individaul term, and off-diagnal cell for the overlapped-members shared betwene two terms
- `call`: the call that produced this result

Note

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

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

See Also

xEnrichViewer, xEnricherGenes

Examples

```
## Not run:
# Load the library
library(XGR)
RData.location="~/Sites/SVN/github/bigdata"

# Enrichment analysis for GWAS SNPs from ImmunoBase
# a) provide input data
data.file <- "http://galahad.well.ox.ac.uk/bigdata/ImmunoBase_GWAS.bed"
data.file <- "~/Sites/SVN/github/bigdata/ImmunoBase_GWAS.bed"

# b) perform DO enrichment analysis for nearby genes (with GWAS SNPs)
eTerm <- xGRviaGeneAnno(data.file=data.file, format.file="bed",
gap.max=0, ontology="DO", RData.location=RData.location)

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

# d) save enrichment results to the file called 'Regions2genes_enrichments.txt'
output <- xEnrichViewer(eTerm, top_num=length(eTerm$adjp),
sortBy="adjp", details=TRUE)
utils::write.table(output, file="Regions2genes_enrichments.txt",
sep="\t", row.names=FALSE)

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

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

---

xGRviaGenomicAnno *Function to conduct region-based enrichment analysis using genomic annotations via binomial test*

Description

xGRviaGenomicAnno is supposed to conduct region-based enrichment analysis for the input genomic region data (genome build h19), using genomic annotations (eg active chromatin, transcription factor binding sites/motifs, conserved sites). Enrichment analysis is based on binomial test for estimating the significance of overlaps at the base resolution. Test background can be provided; by default, the annotatable will be used.

Usage

```
xGRviaGenomicAnno(data.file, annotation.file = NULL, background.file =
NULL,
format.file = c("data.frame", "bed", "chr:start-end", "GRanges"),
build.conversion = c(NA, "hg38.to.hg19", "hg18.to.hg19"),
background.annotatable.only = T, p.adjust.method = c("BH", "BY",
"bonferroni", "holm", "hochberg", "hommel"), GR.annotation = c(NA,
"Uniform_TFBS", "ENCODE_TFBS_ClusteredV3",
"ENCODE_TFBS_ClusteredV3_CellTypes", "Uniform_DNaseI_HS",
"ENCODE_DNaseI_ClusteredV3", "ENCODE_DNaseI_ClusteredV3_CellTypes",
"Broad_Histone", "SYDH_Histone", "UW_Histone", "FANTOM5_Enhancer_Cell",
"FANTOM5_Enhancer_Tissue", "FANTOM5_Enhancer_Extensive",
"FANTOM5_Enhancer",
"Segment_Combined_Gm12878", "Segment_Combined_H1hesc",
"Segment_Combined_Helas3", "Segment_Combined_Hepg2",
"Segment_Combined_Huvec",
"Segment_Combined_K562", "TFBS_Conserved", "TS_miRNA", "TCGA",
"ReMap_Public_TFBS", "ReMap_Public_mergedTFBS",
"ReMap_PublicAndEncode_mergedTFBS", "ReMap_Encode_TFBS",
"Blueprint_BoneMarrow_Histone", "Blueprint_CellLine_Histone",
"Blueprint_CordBlood_Histone", "Blueprint_Thymus_Histone",
"Blueprint_VenousBlood_Histone", "Blueprint_DNaseI"), verbose = T,
RData.location =
"https://github.com/hfang-bristol/RDataCentre/blob/master/Portal")
```

Arguments

data.file	an input data file, containing a list of genomic regions to test. If the input file is formatted as a 'data.frame' (specified by the parameter 'format.file' below), the first three columns correspond to the chromosome (1st column), the starting chromosome position (2nd column), and the ending chromosome position (3rd column). If the format is indicated as 'bed' (browser extensible data), the same

as 'data.frame' format but the position is 0-based offset from chromomose position. If the genomic regions provided are not ranged but only the single position, the ending chromosome position (3rd column) is allowed not to be provided. If the format is indicated as "chr:start-end", instead of using the first 3 columns, only the first column will be used and processed. If the file also contains other columns, these additional columns will be ignored. Alternatively, the input file can be the content itself assuming that input file has been read. Note: the file should use the tab delimiter as the field separator between columns.

`annotation.file`

an input annotation file containing genomic annotations for genomic regions. If the input file is formatted as a 'data.frame', the first four columns correspond to the chromosome (1st column), the starting chromosome position (2nd column), the ending chromosome position (3rd column), and the genomic annotations (eg transcription factors and histones; 4th column). If the format is indicated as 'bed', the same as 'data.frame' format but the position is 0-based offset from chromomose position. If the format is indicated as "chr:start-end", the first two columns correspond to the chromosome:start-end (1st column) and the genomic annotations (eg transcription factors and histones; 2nd column). If the file also contains other columns, these additional columns will be ignored. Alternatively, the input file can be the content itself assuming that input file has been read. Note: the file should use the tab delimiter as the field separator between columns.

`background.file`

an input background file containing a list of genomic regions as the test background. The file format is the same as 'data.file'. By default, it is NULL meaning all annotatable bases (ig non-redundant bases covered by 'annotation.file') are used as background. However, if only one annotation (eg only a transcription factor) is provided in 'annotation.file', the background must be provided.

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

`build.conversion`

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

`background.annotatable.only`

logical to indicate whether the background is further restricted to annotatable bases (covered by 'annotation.file'). In other words, if the background is provided, the background bases are those after being overlapped with annotatable bases. Notably, if only one annotation (eg only a transcription factor) is provided in 'annotation.file', it should be false.

`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

GR.annotation	the genomic regions of annotation data. By default, it is 'NA' to disable this option. Pre-built genomic annotation data are detailed the section 'Note'. Beyond pre-built annotation data, the user can specify the customised input. To do so, first save your RData file (a list of GR objects, each is an GR object correponding to an annotation) into your local computer. Then, tell "GR.annotation" with your RData file name (with or without extension), plus specify your file RData path in "RData.location"
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

a data frame with 8 columns:

- name: the annotation name
- nAnno: the number of bases covered by that annotation. If the background is provided, they are also restricted by this
- nOverlap: the number of regions overlapped between input regions and annotation regions. If the background is provided, they are also restricted by this
- fc: fold change
- zscore: z-score
- pvalue: p-value
- adjp: adjusted p-value. It is the p value but after being adjusted for multiple comparisons
- expProb: the probability of expecting bases overlapped between background regions and annotation regions
- obsProb: the probability of observing regions overlapped between input regions and annotation regions

Note

The genomic annotation data are described below according to the data sources and data types.

1. ENCODE Transcription Factor ChIP-seq data

- Uniform_TFBS: a list (690 combinations of cell types and transcription factors) of GenomicRanges objects; each is an GR object containing uniformly identified peaks per cell type per transcription factor.
- ENCODE_TFBS_ClusteredV3: a list (161 transcription factors) of GenomicRanges objects; each is an GR object containing clustered peaks per transcription factor, along with a meta-column 'cells' telling cell types associtated with a clustered peak.
- ENCODE_TFBS_ClusteredV3_CellTypes: a list (91 cell types) of a list (transcription factors) of GenomicRanges objects. Each cell type is a list (transcription factor) of GenomicRanges objects; each is an GR object containing clustered peaks per transcription factor.

2. ENCODE DNaseI Hypersensitivity site data

- Uniform_DNaseI_HS: a list (125 cell types) of GenomicRanges objects; each is an GR object containing uniformly identified peaks per cell type.
- ENCODE_DNaseI_ClusteredV3: an GR object containing clustered peaks, along with a meta-column 'num_cells' telling how many cell types associtated with a clustered peak.
- ENCODE_DNaseI_ClusteredV3_CellTypes: a list (125 cell types) of GenomicRanges objects; each is an GR object containing clustered peaks per cell type.

3. ENCODE Histone Modification ChIP-seq data from different sources

- Broad_Histone: a list (156 combinations of cell types and histone modifications) of GenomicRanges objects; each is an GR object containing identified peaks per cell type and per histone modification.
- SYDH_Histone: a list (29 combinations of cell types and histone modifications) of GenomicRanges objects; each is an GR object containing identified peaks per cell type and per histone modification.
- UW_Histone: a list (172 combinations of cell types and histone modifications) of GenomicRanges objects; each is an GR object containing identified peaks per cell type and per histone modification.

4. FANTOM5 expressed enhancer atlas

- FANTOM5_Enhancer_Cell: a list (71 cell types) of GenomicRanges objects; each is an GR object containing enhancers specifically expressed in a cell type.
- FANTOM5_Enhancer_Tissue: a list (41 tissues) of GenomicRanges objects; each is an GR object containing enhancers specifically expressed in a tissue.
- FANTOM5_Enhancer_Extensive: a list (5 categories of extensitive enhancers) of GenomicRanges objects; each is an GR object containing extensitive enhancers. They are: "Extensive_ubiquitous_enhancers_cells" for ubiquitous enhancers expressed over the entire set of cell types; "Extensive_ubiquitous_enhancers_organs" for ubiquitous enhancers expressed over the entire set of tissues; "Extensive_enhancers_tss_associations" for TSS-enhancer associations(RefSeq promoters only); "Extensive_permissive_enhancers" and "Extensive_robust_enhancers" for permissive and robust enhancer sets.
- FANTOM5_Enhancer: a list (117 cell types/tissues/categories) of GenomicRanges objects; each is an GR object.

5. ENCODE combined (ChromHMM and Segway) Genome Segmentation data

- Segment_Combined_Gm12878: a list (7 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the cell line GM12878 (human lymphoblastoid cell line).
- Segment_Combined_H1hesc: a list (7 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the cell line H1-hESC (H1 human embryonic stem cell line).
- Segment_Combined_Helas3: a list (7 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the cell line HeLa S3 (human epithelial carcinoma cell line).

- Segment_Combined_Hepg2: a list (7 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the cell line HepG2 (human liver hepatocellular carcinoma cell line).
- Segment_Combined_Huvec: a list (7 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the cell line HUVEC (human umbilical vein endothelial cell line).
- Segment_Combined_K562: a list (7 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the cell line K562 (human erythromyeloblastoid leukemia cell line).

6. UCSC Conserved TFBS

- TFBS_Conserved: a list (245 PWM) of GenomicRanges objects; each is an GR object containing human/mouse/rat conserved TFBS for each PWM.

7. TargetScan miRNA regulatory sites

- TS_miRNA: a list (153 miRNA) of GenomicRanges objects; each is an GR object containing miRNA regulatory sites for each miRNA.

8. TCGA exome mutation data

- TCGA: a list (11 tumor types) of GenomicRanges objects; each is an GR object containing exome mutation across tumor patients of the same tumor type.

9. ReMap integration of transcription factor ChIP-seq data (publicly available and ENCODE)

- ReMap_Public_TFBS: a list (395 combinations of GSE studies and transcription factors and cell types) of GenomicRanges objects; each is an GR object containing identified peaks per GSE study per transcripton factor per cell type.
- ReMap_Public_mergedTFBS: a list (131 transcription factors under GSE studies) of GenomicRanges objects; each is an GR object containing merged peaks per transcripton factor.
- ReMap_PublicAndEncode_mergedTFBS: a list (237 transcription factors under GSE studies and ENCODE) of GenomicRanges objects; each is an GR object containing merged peaks per transcripton factor.
- ReMap_Encode_TFBS: a list (155 transcription factors under ENCODE) of GenomicRanges objects; each is an GR object containing identified peaks per transcripton factor.

10. BLUEPRINT Histone Modification ChIP-seq data from bone marrow

- Blueprint_BoneMarrow_Histone: a list (132 combinations of histone modifications and samples) of GenomicRanges objects; each is an GR object containing identified peaks per histone per sample.

11. BLUEPRINT Histone Modification ChIP-seq data from cell lines

- Blueprint_CellLine_Histone: a list (38 combinations of histone modifications and cell lines) of GenomicRanges objects; each is an GR object containing identified peaks per histone per cell line.

12. BLUEPRINT Histone Modification ChIP-seq data from cord blood

- `Blueprint_CordBlood_Histone`: a list (126 combinations of histone modifications and samples) of GenomicRanges objects; each is an GR object containing identified peaks per histone per sample.

13. BLUEPRINT Histone Modification ChIP-seq data from thymus

- `Blueprint_Thymus_Histone`: a list (5 combinations of histone modifications and samples) of GenomicRanges objects; each is an GR object containing identified peaks per histone per sample.

14. BLUEPRINT Histone Modification ChIP-seq data from venous blood

- `Blueprint_VenousBlood_Histone`: a list (296 combinations of histone modifications and samples) of GenomicRanges objects; each is an GR object containing identified peaks per histone per sample.

15. BLUEPRINT DNaseI Hypersensitivity site data

- `Blueprint_DNaseI`: a list (36 samples) of GenomicRanges objects; each is an GR object containing identified peaks per sample.

See Also

xEnrichViewer

Examples

```
## Not run:
# Load the library
library(XGR)
RData.location="~/Sites/SVN/github/bigdata"

# Enrichment analysis for GWAS SNPs from ImmunoBase
# a) provide input data
data.file <- "http://galahad.well.ox.ac.uk/bigdata/ImmunoBase_GWAS.bed"
data.file <- "~/Sites/SVN/github/bigdata/ImmunoBase_GWAS.bed"

# b) perform enrichment analysis using FANTOM expressed enhancers
eTerm <- xGRviaGenomicAnno(data.file=data.file, format.file="bed",
GR.annotation="FANTOM5_Enhancer_Cell", RData.location=RData.location)

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

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

## End(Not run)
```

xGRviaGenomicAnnoAdv *Function to conduct region-based enrichment analysis using genomic annotations via sampling*

Description

xGRviaGenomicAnnoAdv is supposed to conduct region-based enrichment analysis for the input genomic region data (genome build h19), using genomic annotations (eg active chromatin, transcription factor binding sites/motifs, conserved sites). Enrichment analysis is achieved by comparing the observed overalps against the expected overlaps which are estimated from the null distribution. The null distribution is generated via sampling, that is, randomly generating samples for data genomic regions from background genomic regions. Background genomic regions can be provided by the user; by default, the annotatable genomic regions will be used.

Usage

```
xGRviaGenomicAnnoAdv(data.file, annotation.file = NULL,
background.file = NULL, format.file = c("data.frame", "bed",
"chr:start-end", "GRanges"), build.conversion = c(NA, "hg38.to.hg19",
"hg18.to.hg19"), background.annotatable.only = F, num.samples = 1000,
gap.max = 50000, max.distance = NULL, p.adjust.method = c("BH", "BY",
"bonferroni", "holm", "hochberg", "hommel"), GR.annotation = c(NA,
"Uniform_TFBS", "ENCODE_TFBS_ClusteredV3",
"ENCODE_TFBS_ClusteredV3_CellTypes", "Uniform_DNaseI_HS",
"ENCODE_DNaseI_ClusteredV3", "ENCODE_DNaseI_ClusteredV3_CellTypes",
"Broad_Histone", "SYDH_Histone", "UW_Histone", "FANTOM5_Enhancer_Cell",
"FANTOM5_Enhancer_Tissue", "FANTOM5_Enhancer_Extensive",
"FANTOM5_Enhancer",
"Segment_Combined_Gm12878", "Segment_Combined_H1hesc",
"Segment_Combined_Helas3", "Segment_Combined_Hepg2",
"Segment_Combined_Huvec",
"Segment_Combined_K562", "TFBS_Conserved", "TS_miRNA", "TCGA",
"ReMap_Public_TFBS", "ReMap_Public_mergedTFBS",
"ReMap_PublicAndEncode_mergedTFBS", "ReMap_Encode_TFBS",
"Blueprint_BoneMarrow_Histone", "Blueprint_CellLine_Histone",
"Blueprint_CordBlood_Histone", "Blueprint_Thymus_Histone",
"Blueprint_VenousBlood_Histone", "Blueprint_DNaseI"), parallel = TRUE,
multicores = NULL, verbose = T,
RData.location =
"https://github.com/hfang-bristol/RDataCentre/blob/master/Portal")
```

Arguments

data.file	an input data file, containing a list of genomic regions to test. If the input file is formatted as a 'data.frame' (specified by the parameter 'format.file' below), the first three columns correspond to the chromosome (1st column), the starting chromosome position (2nd column), and the ending chromosome position (3rd

column). If the format is indicated as 'bed' (browser extensible data), the same as 'data.frame' format but the position is 0-based offset from chromomose position. If the genomic regions provided are not ranged but only the single position, the ending chromosome position (3rd column) is allowed not to be provided. If the format is indicated as "chr:start-end", instead of using the first 3 columns, only the first column will be used and processed. If the file also contains other columns, these additional columns will be ignored. Alternatively, the input file can be the content itself assuming that input file has been read. Note: the file should use the tab delimiter as the field separator between columns.

`annotation.file`

an input annotation file containing genomic annotations for genomic regions. If the input file is formatted as a 'data.frame', the first four columns correspond to the chromosome (1st column), the starting chromosome position (2nd column), the ending chromosome position (3rd column), and the genomic annotations (eg transcription factors and histones; 4th column). If the format is indicated as 'bed', the same as 'data.frame' format but the position is 0-based offset from chromomose position. If the format is indicated as "chr:start-end", the first two columns correspond to the chromosome:start-end (1st column) and the genomic annotations (eg transcription factors and histones; 2nd column). If the file also contains other columns, these additional columns will be ignored. Alternatively, the input file can be the content itself assuming that input file has been read. Note: the file should use the tab delimiter as the field separator between columns.

`background.file`

an input background file containing a list of genomic regions as the test background. The file format is the same as 'data.file'. By default, it is NULL meaning all annotatable bases (ig non-redundant bases covered by 'annotation.file') are used as background. However, if only one annotation (eg only a transcription factor) is provided in 'annotation.file', the background must be provided.

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

`build.conversion`

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

`background.annotatable.only`

logical to indicate whether the background is further restricted to annotatable bases (covered by 'annotation.file'). In other words, if the background is provided, the background bases are those after being overlapped with annotatable bases. Notably, if only one annotation (eg only a transcription factor) is provided in 'annotation.file', it should be false.

`num.samples` the number of samples randomly generated

`gap.max` the maximum distance of background islands to be considered away from data regions. Only background islands no far way from this distance will be considered. For example, if it is 0, meaning that only background islands that overlapp with genomic regions will be considered. By default, it is 50000

`max.distance` the maximum distance away from data regions that is allowed when generating random samples. By default, it is NULl meaning no such restriction

`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

`GR.annotation`
: the genomic regions of annotation data. By default, it is 'NA' to disable this option. Pre-built genomic annotation data are detailed the section 'Note'. Beyond pre-built annotation data, the user can specify the customised input. To do so, first save your RData file (a list of GR objects, each is an GR object correponding to an annotation) into your local computer. Then, tell "GR.annotation" with your RData file name (with or without extension), plus specify your file RData path in "RData.location"

`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. It can be installed via: `source("http://bioconductor.org/biocLite.R"); biocLite(c("foreach","doMC"))`. If not yet installed, this option will be disabled

`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 false for no display

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

Value

a data frame with 8 columns:

- `name`: the annotation name
- `nAnno`: the number of bases covered by that annotation. If the background is provided, they are also restricted by this
- `nOverlap`: the number of bases overlapped between input regions and annotation regions. If the background is provided, they are also restricted by this
- `fc`: fold change
- `zscore`: z-score
- `pvalue`: p-value
- `adjp`: adjusted p-value. It is the p value but after being adjusted for multiple comparisons
- `nData`: the number of bases covered by input regions
- `nBG`: the number of bases covered by background regions

Note

The genomic annotation data are described below according to the data sources and data types.

1. ENCODE Transcription Factor ChIP-seq data

   - `Uniform_TFBS`: a list (690 combinations of cell types and transcription factors) of GenomicRanges objects; each is an GR object containing uniformly identified peaks per cell type per transcription factor.
   - `ENCODE_TFBS_ClusteredV3`: a list (161 transcription factors) of GenomicRanges objects; each is an GR object containing clustered peaks per transcription factor, along with a meta-column 'cells' telling cell types associtated with a clustered peak.
   - `ENCODE_TFBS_ClusteredV3_CellTypes`: a list (91 cell types) of a list (transcription factors) of GenomicRanges objects. Each cell type is a list (transcription factor) of GenomicRanges objects; each is an GR object containing clustered peaks per transcription factor.

2. ENCODE DNaseI Hypersensitivity site data

   - `Uniform_DNaseI_HS`: a list (125 cell types) of GenomicRanges objects; each is an GR object containing uniformly identified peaks per cell type.
   - `ENCODE_DNaseI_ClusteredV3`: an GR object containing clustered peaks, along with a meta-column 'num_cells' telling how many cell types associtated with a clustered peak.
   - `ENCODE_DNaseI_ClusteredV3_CellTypes`: a list (125 cell types) of GenomicRanges objects; each is an GR object containing clustered peaks per cell type.

3. ENCODE Histone Modification ChIP-seq data from different sources

   - `Broad_Histone`: a list (156 combinations of cell types and histone modifications) of GenomicRanges objects; each is an GR object containing identified peaks per cell type and per histone modification.
   - `SYDH_Histone`: a list (29 combinations of cell types and histone modifications) of GenomicRanges objects; each is an GR object containing identified peaks per cell type and per histone modification.
   - `UW_Histone`: a list (172 combinations of cell types and histone modifications) of GenomicRanges objects; each is an GR object containing identified peaks per cell type and per histone modification.

4. FANTOM5 expressed enhancer atlas

   - `FANTOM5_Enhancer_Cell`: a list (71 cell types) of GenomicRanges objects; each is an GR object containing enhancers specifically expressed in a cell type.
   - `FANTOM5_Enhancer_Tissue`: a list (41 tissues) of GenomicRanges objects; each is an GR object containing enhancers specifically expressed in a tissue.
   - `FANTOM5_Enhancer_Extensive`: a list (5 categories of extensitive enhancers) of GenomicRanges objects; each is an GR object containing extensitive enhancers. They are: "Extensive_ubiquitous_enhancers_cells" for ubiquitous enhancers expressed over the entire set of cell types; "Extensive_ubiquitous_enhancers_organs" for ubiquitous enhancers expressed over the entire set of tissues; "Extensive_enhancers_tss_associations" for TSS-enhancer associations(RefSeq promoters only); "Extensive_permissive_enhancers" and "Extensive_robust_enhancers" for permissive and robust enhancer sets.

- FANTOM5_Enhancer: a list (117 cell types/tissues/categories) of GenomicRanges objects; each is an GR object.

5. ENCODE combined (ChromHMM and Segway) Genome Segmentation data

- Segment_Combined_Gm12878: a list (7 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the cell line GM12878 (a lymphoblastoid cell line).
- Segment_Combined_H1hesc: a list (7 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the cell line H1-hESC (H1 human embryonic stem cells).
- Segment_Combined_Helas3: a list (7 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the cell line HeLa S3.
- Segment_Combined_Hepg2: a list (7 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the cell line HepG2 (liver hepatocellular carcinoma).
- Segment_Combined_Huvec: a list (7 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the cell line HUVEC (Human Umbilical Vein Endothelial Cells).
- Segment_Combined_K562: a list (7 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the cell line K562 (human erythromyeloblastoid leukemia cell line).

6. Conserved TFBS

- TFBS_Conserved: a list (245 PWM) of GenomicRanges objects; each is an GR object containing human/mouse/rat conserved TFBS for each PWM.

7. TargetScan miRNA regulatory sites

- TS_miRNA: a list (153 miRNA) of GenomicRanges objects; each is an GR object containing miRNA regulatory sites for each miRNA.

8. TCGA exome mutation data

- TCGA: a list (11 tumor types) of GenomicRanges objects; each is an GR object containing exome mutation across tumor patients of the same tumor type.

9. ReMap integration of transcription factor ChIP-seq data (publicly available and ENCODE)

- ReMap_Public_TFBS: a list (395 combinations of GSE studies and transcription factors and cell types) of GenomicRanges objects; each is an GR object containing identified peaks per GSE study per transcripton factor per cell type.
- ReMap_Public_mergedTFBS: a list (131 transcription factors under GSE studies) of GenomicRanges objects; each is an GR object containing merged peaks per transcripton factor.
- ReMap_PublicAndEncode_mergedTFBS: a list (237 transcription factors under GSE studies and ENCODE) of GenomicRanges objects; each is an GR object containing merged peaks per transcripton factor.
- ReMap_Encode_TFBS: a list (155 transcription factors under ENCODE) of GenomicRanges objects; each is an GR object containing identified peaks per transcripton factor.

10. Blueprint Histone Modification ChIP-seq data from bone marrow

    - `Blueprint_BoneMarrow_Histone`: a list (132 combinations of histone modifications and samples) of GenomicRanges objects; each is an GR object containing identified peaks per histone per sample.

11. Blueprint Histone Modification ChIP-seq data from cell lines

    - `Blueprint_CellLine_Histone`: a list (38 combinations of histone modifications and cell lines) of GenomicRanges objects; each is an GR object containing identified peaks per histone per cell line.

12. Blueprint Histone Modification ChIP-seq data from cord blood

    - `Blueprint_CordBlood_Histone`: a list (126 combinations of histone modifications and samples) of GenomicRanges objects; each is an GR object containing identified peaks per histone per sample.

13. Blueprint Histone Modification ChIP-seq data from thymus

    - `Blueprint_Thymus_Histone`: a list (5 combinations of histone modifications and samples) of GenomicRanges objects; each is an GR object containing identified peaks per histone per sample.

14. Blueprint Histone Modification ChIP-seq data from venous blood

    - `Blueprint_VenousBlood_Histone`: a list (296 combinations of histone modifications and samples) of GenomicRanges objects; each is an GR object containing identified peaks per histone per sample.

15. Blueprint DNaseI Hypersensitivity site data

    - `Blueprint_DNaseI`: a list (36 samples) of GenomicRanges objects; each is an GR object containing identified peaks per sample.

See Also

`xEnrichViewer`

Examples

```
## Not run:
# Load the library
library(XGR)
RData.location="~/Sites/SVN/github/bigdata"

# Enrichment analysis for GWAS SNPs from ImmunoBase
# a) provide input data
data.file <- "http://galahad.well.ox.ac.uk/bigdata/ImmunoBase_GWAS.bed"
#data.file <- "~/Sites/SVN/github/bigdata/ImmunoBase_GWAS.bed"

# b) perform enrichment analysis using FANTOM expressed enhancers
eTerm <- xGRviaGenomicAnnoAdv(data.file=data.file, format.file="bed",
GR.annotation="FANTOM5_Enhancer_Cell", num.samples=1000, gap.max=50000,
```

```
RData.location=RData.location)

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

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

## End(Not run)
```

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

Description

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

Usage

```
xLiftOver(data.file, format.file = c("data.frame", "bed",
"chr:start-end",
"GRanges"), build.conversion = c(NA, "hg38.to.hg19", "hg19.to.hg38",
"hg19.to.hg18", "hg18.to.hg38", "hg18.to.hg19"), merged = T, verbose =
T,
RData.location =
"https://github.com/hfang-bristol/RDataCentre/blob/master/Portal")
```

Arguments

`data.file`	an input data file, containing a list of genomic regions to test. If the input file is formatted as a 'data.frame' (specified by the parameter 'format.file' below), the first three columns correspond to the chromosome (1st column), the starting chromosome position (2nd column), and the ending chromosome position (3rd column). If the format is indicated as 'bed' (browser extensible data), the same as 'data.frame' format but the position is 0-based offset from chromomose position. If the genomic regions provided are not ranged but only the single position, the ending chromosome position (3rd column) is allowed not to be provided. If the format is indicated as "chr:start-end", instead of using the first 3 columns, only the first column will be used and processed. If the file also contains other columns, these additional columns will be ignored. Alternatively, the input file can be the content itself assuming that input file has been read. Note: the file should use the tab delimiter as the field separator between columns
`format.file`	the format for input files. It can be one of "data.frame", "chr:start-end", "bed"

build.conversion
: the conversion from one genome build to another. The conversions supported are "hg38.to.hg19", "hg19.to.hg38", "hg19.to.hg18", "hg18.to.hg38" and "hg18.to.hg19". By default it is NA, forcing the user to specify the corrent one.

merged
: logical to indicate whether multiple ranges should be merged into the one per a range in query. By default, it sets to true

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

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

Value

an GR oject storing converted genomic intervals.

See Also

xLiftOver

Examples

```
## Not run:
# Load the library
library(XGR)
RData.location="~/Sites/SVN/github/bigdata"

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

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

## End(Not run)
```

---

xRd2HTML *Function to convert Rd files to HTML files*

Description

xRd2HTML is supposed to convert Rd files to HTML files.

Usage

```
xRd2HTML(path.from = "./XGR/man", path.to = "./XGR/vignettes")
```

Arguments

path.from	a directory containing Rd files converted from
path.to	a directory containing HTML files converted to

Value

none

Note

This auxiliary function helps create a new package.

See Also

xRd2HTML

Examples

```
# xRd2HTML(path.from="./XGR/man", path.to="./XGR/vignettes")
```

---

xRDataLoader *Function to load the package built-in RData*

Description

xRDataLoader is supposed to load the package built-in RData.

Usage

```
xRDataLoader(RData = c(NA, "GWAS2EF", "GWAS_LD", "IlluminaHumanHT",
"IlluminaOmniExpress", "ig.DO", "ig.EF", "ig.GOBP", "ig.GOCC",
"ig.GOMF",
"ig.HPCM", "ig.HPMA", "ig.HPMI", "ig.HPPA", "ig.MP", "org.Hs.eg",
"org.Hs.egDGIdb", "org.Hs.egDO", "org.Hs.egGOBP", "org.Hs.egGOCC",
"org.Hs.egGOMF", "org.Hs.egHPCM", "org.Hs.egHPMA", "org.Hs.egHPMI",
"org.Hs.egHPPA", "org.Hs.egMP", "org.Hs.egMsigdbC1",
"org.Hs.egMsigdbC2BIOCARTA", "org.Hs.egMsigdbC2CGP",
"org.Hs.egMsigdbC2CPall",
"org.Hs.egMsigdbC2CP", "org.Hs.egMsigdbC2KEGG",
"org.Hs.egMsigdbC2REACTOME", "org.Hs.egMsigdbC3MIR",
"org.Hs.egMsigdbC3TFT",
"org.Hs.egMsigdbC4CGN", "org.Hs.egMsigdbC4CM", "org.Hs.egMsigdbC5BP",
"org.Hs.egMsigdbC5CC", "org.Hs.egMsigdbC5MF", "org.Hs.egMsigdbC6",
"org.Hs.egMsigdbC7", "org.Hs.egMsigdbH", "org.Hs.egPS", "org.Hs.egSF",
"org.Hs.string", "org.Hs.PCommons_DN", "org.Hs.PCommons_UN"),
RData.customised = NULL, verbose = T,
RData.location =
"https://github.com/hfang-bristol/RDataCentre/blob/master/Portal")
```

Arguments

RData	which built-in RData to load. It can be one of "GWAS2EF", "GWAS_LD", "IlluminaHumanHT", "IlluminaOmniExpress", "ig.DO", "ig.EF", "ig.GOBP", "ig.GOCC", "ig.GOMF", "ig.HPCM", "ig.HPMA", "ig.HPMI", "ig.HPPA", "ig.MP", "org.Hs.eg", "org.Hs.egDGIdb", "org.Hs.egDO", "org.Hs.egGOBP", "org.Hs.egGOCC", "org.Hs.egGOMF", "org.Hs.egHPCM", "org.Hs.egHPMA", "org.Hs.egHPMI", "org.Hs.egHPPA", "org.Hs.egMP", "org.Hs.egMsigdbC1", "org.Hs.egMsigdbC2BIOCARTA", "org.Hs.egMsigdbC2CGP", "org.Hs.egMsigdbC2CPall", "org.Hs.egMsigdbC2CP", "org.Hs.egMsigdbC2KEGG", "org.Hs.egMsigdbC2REACTOME", "org.Hs.egMsigdbC3MIR", "org.Hs.egMsigdbC3TFT", "org.Hs.egMsigdbC4CGN", "org.Hs.egMsigdbC4CM", "org.Hs.egMsigdbC5BP", "org.Hs.egMsigdbC5CC", "org.Hs.egMsigdbC5MF", "org.Hs.egMsigdbC6", "org.Hs.egMsigdbC7", "org.Hs.egMsigdbH", "org.Hs.egPS", "org.Hs.egSF", "org.Hs.string", "org.Hs.PCommons_DN", "org.Hs.PCommons_UN"
RData.customised	a file name for RData-formatted file. By default, it is NULL. It is designed when the user wants to import customised RData that are not listed in the above argument 'RData'. However, this argument can be always used even for those RData that are listed in the argument 'RData'
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to TRUE for display
RData.location	the characters to tell the location of built-in RData files. By default, it remotely locates at https://github.com/hfang-bristol/RDataCentre/blob/master/Portal and http://galahad.well.ox.ac.uk/bigdata. For the user equipped with fast internet connection, this option can be just left as default. But it is always advisable to download these files locally. Especially when the user needs to run this function many times, there is no need to ask the function to remotely download every time (also it will unnecessarily increase the runtime). For examples, these files (as a whole or part of them) can be first downloaded into your current working directory, and then set this option as: $RData.location = "."$. Surely, the location can be anywhere as long as the user provides the correct path pointing to (otherwise, the script will have to remotely download each time)

Value

any use-specified variable that is given on the right side of the assigement sign '<-', which contains the loaded RData. If the data cannot be loaded, it returns NULL.

Note

If there are no use-specified variable that is given on the right side of the assigement sign '<-', then no RData will be loaded onto the working environment.

See Also

xRDataLoader

Examples

```
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase')
## Not run:
org.Hs.eg <- xRDataLoader(RData='org.Hs.eg')
ig.HPPA <- xRDataLoader(RData='ig.HPPA')
org.Hs.egHPPA <- xRDataLoader(RData='org.Hs.egHPPA')
org.Hs.egHPPA <- xRDataLoader(RData.customised='org.Hs.egHPPA')

## End(Not run)
```

---

xRdWrap *Function to wrap texts from Rd files*

Description

xRdWrap is supposed to wrap texts from Rd files under a given directory.

Usage

```
xRdWrap(path = "./XGR/man", remove.dontrun = FALSE)
```

Arguments

path	a directory containing Rd files
remove.dontrun	logical to indicate whether to remove the restriction of not running examples. By default, it sets to FALSE without any modefications

Value

none

Note

This auxiliary function helps create a new package. The orignal Rd files will be replaced with new ones.

See Also

xRdWrap

Examples

```
# xRdWrap(path="./XGR/man", remove.dontrun=FALSE)
```

xSNP2GeneScores *Function to identify likely modulated seed genes given a list of SNPs together with the significance level (e.g. GWAS reported p-values)*

Description

xSNP2GeneScores is supposed to identify likely modulated seed genes from a list of SNPs together with the significance level (measured as p-values or fdr). To do so, it defines seed genes and their scores that take into account the distance to and the significance of input SNPs. It returns an object of class "mSeed".

Usage

```
xSNP2GeneScores(data, include.LD = NA, LD.customised = NULL, LD.r2 =
0.8,
significance.threshold = 5e-05, distance.max = 50000,
decay.kernel = c("slow", "linear", "rapid"), decay.exponent = 2,
GR.SNP = c("dbSNP_GWAS", "dbSNP_Common"), GR.Gene = c("UCSC_knownGene",
"UCSC_knownCanonical"), scoring.scheme = c("max", "sum", "sequential"),
verbose = T,
RData.location =
"https://github.com/hfang-bristol/RDataCentre/blob/master/Portal")
```

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 precalcuated LD info. This customisation (if provided) has the high priority over built-in LD SNPs
LD.r2	the LD r2 value. By default, it is 0.8, meaning that SNPs in LD (r2>=0.8) with input SNPs will be considered as LD SNPs. It can be any value from 0.8 to 1

`significance.threshold`
: the given significance threshold. By default, it is set to NULL, meaning there is no constraint on the significance level when transforming the significance level of SNPs into scores. If given, those SNPs below this are considered significant and thus scored positively. Instead, those above this are considered insigificant and thus receive no score

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

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

`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"

`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)

`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 "mSeed", a list with following components:

- `SNP`: a matrix of nSNP X 3 containing SNP information, where nSNP is the number of SNPs, and the 3 columns are "SNP" (Lead and/or LD SNPs), "Score" (the scores for SNPs calculated based on p-values taking into account the given threshold of the significant level), "Pval" (the input p-values for Lead SNPs or R2-adjusted p-values for LD SNPs)

- Gene: a matrix of nGene X 3 containing Gene information, where nGene is the number of seed genes, and the 3 columns are "Gene" (gene symbol), "Score" (the scores for seed genes), "Pval" (pvalue-like significance level transformed from gene scores)
- call: the call that produced this result

Note

This function uses xSNPscores and xSNP2nGenes to define and score nearby genes that are located within distance window of input and/or LD SNPs.

See Also

xSNPscores, xSNP2nGenes, xSparseMatrix

Examples

```
## Not run:
# Load the library
library(XGR)
RData.location="~/Sites/SVN/github/RDataCentre/Portal"

# a) provide the seed SNPs with the significance info
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase')
## get lead SNPs reported in AS GWAS and their significance info (p-values)
gr <- ImmunoBase$AS$variant
data <- GenomicRanges::mcols(gr)[,c(1,3)]

# b) define and score seed geens
mSeed <- xSNP2GeneScores(data=data, RData.location=RData.location)

# c) extract SNP info
head(mSeed$SNP)

# d) extract gene info
head(mSeed$Gene)

## End(Not run)
```

---

xSNP2nGenes *Function to define nearby genes given a list of SNPs*

Description

xSNP2nGenes is supposed to define nearby genes given a list of SNPs within certain distance window. The distance weight is calcualted as a decaying function of the gene-to-SNP distance.

Usage

```
xSNP2nGenes(data, distance.max = 2e+05, decay.kernel = c("rapid",
"slow",
"linear"), decay.exponent = 2, GR.SNP = c("dbSNP_GWAS",
"dbSNP_Common"),
GR.Gene = c("UCSC_knownGene", "UCSC_knownCanonical"), verbose = T,
RData.location =
"https://github.com/hfang-bristol/RDataCentre/blob/master/Portal")
```

Arguments

data	a input vector containing SNPs. SNPs should be provided as dbSNP ID (ie starting with rs). Alternatively, they can be in the format of 'chrN:xxx', where N is either 1-22 or X, xxx is number; for example, 'chr16:28525386'
distance.max	the maximum distance between genes and SNPs. Only those genes no far way from this distance will be considered as seed genes. This parameter will influence the distance-component weights calculated for nearby SNPs per gene
decay.kernel	a character specifying a decay kernel function. It can be one of 'slow' for slow decay, 'linear' for linear decay, and 'rapid' for rapid decay
decay.exponent	a numeric specifying a decay exponent. By default, it sets to 2
GR.SNP	the genomic regions of SNPs. By default, it is 'dbSNP_GWAS', that is, SNPs from dbSNP (version 146) restricted to GWAS SNPs and their LD SNPs (hg19). It can be 'dbSNP_Common', that is, Common SNPs from dbSNP (version 146) plus GWAS SNPs and their LD SNPs (hg19). Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to dbSNP IDs. Then, tell "GR.SNP" with your RData file name (with or without extension), plus specify your file RData path in "RData.location"
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"
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: nearby genes

- SNP: SNPs
- Dist: the genomic distance between the gene and the SNP
- Weight: the distance weight based on the gnomic distance

Note

For details on the decay kernels, please refer to xVisKernels

See Also

xRDataLoader, xVisKernels

Examples

```
## Not run:
# Load the library
library(XGR)
RData.location="~/Sites/SVN/github/RDataCentre/Portal"

# a) provide the seed SNPs with the significance info
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase')
## get lead SNPs reported in AS GWAS and their significance info (p-values)
gr <- ImmunoBase$AS$variant
data <- names(gr)

# b) define nearby genes
df_nGenes <- xSNP2nGenes(data=data, distance.max=200000,
decay.kernel="slow", decay.exponent=2, RData.location=RData.location)

## End(Not run)
```

---

xSNPscores — *Function to score lead or LD SNPs based on the given significance level*

Description

xSNPscores is supposed to score a list of Lead SNPs together with the significance level. It can consider LD SNPs and the given threshold of the significant level.

Usage

```
xSNPscores(data, include.LD = NA, LD.customised = NULL, LD.r2 = 0.8,
significance.threshold = 5e-05, verbose = T,
RData.location =
"https://github.com/hfang-bristol/RDataCentre/blob/master/Portal")
```

Arguments

data	a named input vector containing the sinificance level for nodes (dbSNP). For this named vector, the element names are dbSNP (starting with rs or in the format of 'chrN:xxx', where N is either 1-22 or X, xxx is number; for example, 'chr16:28525386'), the element values for the significance level (measured as p-value or fdr). Alternatively, it can be a matrix or data frame with two columns: 1st column for dbSNP, 2nd column for the significance level.
include.LD	additional SNPs in LD with Lead SNPs are also included. By default, it is 'NA' to disable this option. Otherwise, LD SNPs will be included based on one or more of 26 populations and 5 super populations from 1000 Genomics Project data (phase 3). The population can be one of 5 super populations ("AFR", "AMR", "EAS", "EUR", "SAS"), or one of 26 populations ("ACB", "ASW", "BEB", "CDX", "CEU", "CHB", "CHS", "CLM", "ESN", "FIN", "GBR", "GIH", "GWD", "IBS", "ITU", "JPT", "KHV", "LWK", "MSL", "MXL", "PEL", "PJL", "PUR", "STU", "TSI", "YRI"). Explanations for population code can be found at http://www.1000genomes.org/faq/which-populations-are-part-your-study
LD.customised	a user-input matrix or data frame with 3 columns: 1st column for Lead SNPs, 2nd column for LD SNPs, and 3rd for LD r2 value. It is designed to allow the user analysing their precalcuated LD info. This customisation (if provided) has the high priority over built-in LD SNPs
LD.r2	the LD r2 value. By default, it is 0.8, meaning that SNPs in LD (r2>=0.8) with input SNPs will be considered as LD SNPs. It can be any value from 0.8 to 1
significance.threshold	the given significance threshold. By default, it is set to NULL, meaning there is no constraint on the significance level when transforming the significance level of SNPs into scores. If given, those SNPs below this are considered significant and thus scored positively. Instead, those above this are considered insigificant and thus receive no score
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:

- SNP: Lead and/or LD SNPs
- Score: the scores for SNPs calculated based on p-values taking into account the given threshold of the significant level
- Pval: the input p-values for Lead SNPs or R2-adjusted p-values for LD SNPs

Note

None

See Also

xRDataLoader

Examples

```
## Not run:
# Load the library
library(XGR)
RData.location="~/Sites/SVN/github/RDataCentre/Portal"

# a) provide the seed SNPs with the significance info
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase')
## get lead SNPs reported in AS GWAS and their significance info (p-values)
gr <- ImmunoBase$AS$variant
data <- GenomicRanges::mcols(gr)[,c(1,3)]

# b) calculate SNP scores (considering significant cutoff 5e-5)
## without inclusion of LD SNPs
df_SNP <- xSNPscores(data=data, significance.threshold=5e-5,
RData.location=RData.location)
## include LD SNPs (calculated based on European populations)
df_SNP <- xSNPscores(data=data, significance.threshold=5e-5,
include.LD="EUR", RData.location=RData.location)

## End(Not run)
```

---

xSocialiser *Function to calculate pair-wise semantic similarity given the input data and the ontology and its annotation*

Description

xSocialiser is supposed to calculate pair-wise semantic similarity given the input data and the ontology direct acyclic graph (DAG) and its annotation. It returns an object of class "igraph", a network representation of socialized genes/SNPs. It first calculates semantic similarity between terms and then derives semantic similarity from term-term semantic similarity. Parallel computing is also supported for Linux or Mac operating systems.

Usage

```
xSocialiser(data, annotation, g, measure = c("BM.average", "BM.max",
"BM.complete", "average", "max"), method.term = c("Resnik", "Lin",
"Schlicker", "Jiang", "Pesquita"), rescale = TRUE, force = TRUE,
fast = TRUE, parallel = TRUE, multicores = NULL,
path.mode = c("all_paths", "shortest_paths", "all_shortest_paths"),
true.path.rule = TRUE, verbose = T)
```

Arguments

data	an input vector containing a list of genes or SNPs of interest between which pair-wise semantic similarity is calculated/socialized
annotation	the vertices/nodes for which annotation data are provided. It can be a sparse Matrix of class "dgCMatrix" (with variants/genes as rows and terms as columns), or a list of nodes/terms each containing annotation data, or an object of class 'GS' (basically a list for each node/term with annotation data)
g	an object of class "igraph" to represent DAG. It must have node/vertice attributes: "name" (i.e. "Term ID"), "term_id" (i.e. "Term ID"), "term_name" (i.e "Term Name") and "term_distance" (i.e. Term Distance: the distance to the root; always 0 for the root itself)
measure	the measure used to derive semantic similarity between genes/SNPs from semantic similarity between terms. Take the semantic similartity between SNPs as an example. It can be "average" for average similarity between any two terms (one from SNP 1, the other from SNP 2), "max" for the maximum similarity between any two terms, "BM.average" for best-matching (BM) based average similarity (i.e. for each term of either SNP, first calculate maximum similarity to any term in the other SNP, then take average of maximum similarity; the final BM-based average similiary is the pre-calculated average between two SNPs in pair), "BM.max" for BM based maximum similarity (i.e. the same as "BM.average", but the final BM-based maximum similiary is the maximum of the pre-calculated average between two SNPs in pair), "BM.complete" for BM-based complete-linkage similarity (inspired by complete-linkage concept: the least of any maximum similarity between a term of one SNP and a term of the other SNP). When comparing BM-based similarity between SNPs, "BM.average" and "BM.max" are sensitive to the number of terms involved; instead, "BM.complete" is much robust in this aspect. By default, it uses "BM.average"
method.term	the method used to measure semantic similarity between terms. It can be "Resnik" for information content (IC) of most informative common ancestor (MICA) (see http://dl.acm.org/citation.cfm?id=1625914), "Lin" for 2*IC at MICA divided by the sum of IC at pairs of terms, "Schlicker" for weighted version of 'Lin' by the 1-prob(MICA) (see http://www.ncbi.nlm.nih.gov/pubmed/16776819), "Jiang" for 1 - difference between the sum of IC at pairs of terms and 2*IC at MICA (see http://arxiv.org/pdf/cmp-lg/9709008.pdf), "Pesquita" for graph information content similarity related to Tanimoto-Jacard index (ie. summed information content of common ancestors divided by summed information content of all ancestors of term1 and term2 (see http://www.ncbi.nlm.nih.gov/pubmed/18460186))
rescale	logical to indicate whether the resulting values are rescaled to the range [0,1]. By default, it sets to true
force	logical to indicate whether the only most specific terms (for each SNP) will be used. By default, it sets to true. It is always advisable to use this since it is computationally fast but without compromising accuracy (considering the fact that true-path-rule has been applied when running xDAGanno)
fast	logical to indicate whether a vectorised fast computation is used. By default, it sets to true. It is always advisable to use this vectorised fast computation; since the conventional computation is just used for understanding scripts

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. It can be installed via: `source("http://bioconductor.org/biocLite.R"); biocLite(c("foreach","doMC"))`. If not yet installed, this option will be disabled
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
path.mode	the mode of paths induced by vertices/nodes with input annotation data. It can be "all_paths" for all possible paths to the root, "shortest_paths" for only one path to the root (for each node in query), "all_shortest_paths" for all shortest paths to the root (i.e. for each node, find all shortest paths with the equal lengths)
true.path.rule	logical to indicate whether the true-path rule should be applied to propagate annotations. By default, it sets to true
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

It returns an object of class "igraph", with nodes for input genes/SNPs and edges for pair-wise semantic similarity between them. Also added graph attribute is 'dag' storing the annotated ontology DAG used. If no similarity is calculuated, it returns NULL.

Note

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

See Also

xDAGsim, xSocialiserGenes, xSocialiserSNPs

Examples

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

# 1) SNP-based enrichment analysis using GWAS Catalog traits (mapped to EF)
# 1a) ig.EF (an object of class "igraph" storing as a directed graph)
g <- xRDataLoader('ig.EF')
g

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

```
# 1c) prepare the input SNPs of interest (eg 8 randomly chosen SNPs)
allSNPs <- rownames(anno)
data <- sample(allSNPs,8)

# 1d) perform calculate pair-wise semantic similarity between 8 randomly chosen SNPs
sim <- xSocialiser(data=data, annotation=anno, g=g, parallel=FALSE,
verbose=TRUE)
sim

# 1e) save similarity results to the file called 'EF_similarity.txt'
output <- igraph::get.data.frame(sim, what="edges")
utils::write.table(output, file="EF_similarity.txt", sep="\t",
row.names=FALSE)

# 1f) visualise the SNP network
## extract edge weight (with 2-digit precision)
x <- signif(as.numeric(E(sim)$weight), digits=2)
## rescale into an interval [1,4] as edge width
edge.width <- 1 + (x-min(x))/(max(x)-min(x))*3
## do visualisation
xVisNet(g=sim, vertex.shape="sphere", edge.width=edge.width,
edge.label=x, edge.label.cex=0.7)

## End(Not run)
```

`xSocialiserDAGplot` *Function to draw DAG plot for visualising terms used to annotate an input SNP or gene*

Description

`xSocialiserDAGplot` is supposed to draw DAG plot for visualising terms used to annotate an input SNP or gene. By default, terms used for direct/original annotations by box-shaped nodes, and terms for indirect/inherited annotations by ellipse nodes. This function is part of utilities in understanding calculated similarity. It returns an object of class 'Ragraph' or class 'igraph'.

Usage

```
xSocialiserDAGplot(g, query, displayBy = c("IC", "none"),
path.mode = c("all_paths", "shortest_paths", "all_shortest_paths"),
height = 7, width = 7, margin = rep(0.1, 4), colormap = c("yr", "bwr",
"jet", "gbr", "wyr", "br", "rainbow", "wb", "lightyellow-orange"),
ncolors = 40, zlim = NULL, colorbar = T, colorbar.fraction = 0.1,
newpage = T, layout.orientation = c("top_bottom", "left_right",
"bottom_top", "right_left"), node.info = c("none", "term_id",
"term_name",
"both", "full_term_name"), wrap.width = NULL, graph.node.attrs = NULL,
graph.edge.attrs = NULL, node.attrs = NULL, output.format =
c("Ragraph",
"igraph"))
```

Arguments

g	an object of class "igraph" (resulting from similarity analysis)
query	an object in query (for example, an SNP or Gene)
displayBy	which statistics will be used for displaying. It can be "IC" for information content (by default), "none" for no color-coding on nodes/terms
path.mode	the mode of paths induced by nodes/terms. 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)
height	a numeric value specifying the height of device
width	a numeric value specifying the width of device
margin	margins as units of length 4 or 1
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "lightyellow-orange" (by default), "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
ncolors	the number of colors specified over the colormap
zlim	the minimum and maximum z/data values for which colors should be plotted, defaulting to the range of the finite values of z. Each of the given colors will be used to color an equispaced interval of this range. The midpoints of the intervals cover the range, so that values just outside the range will be plotted
colorbar	logical to indicate whether to append a colorbar. If data is null, it always sets to false
colorbar.fraction	the relative fraction of colorbar block against the device size
newpage	logical to indicate whether to open a new page. By default, it sets to true for opening a new page
layout.orientation	the orientation of the DAG layout. It can be one of "left_right" for the left-right layout (viewed from the DAG root point), "top_bottom" for the top-bottom layout, "bottom_top" for the bottom-top layout, and "right_left" for the right-left layout
node.info	tells the ontology term information used to label nodes. It can be one of "none" for no node labeling, "term_id" for using Term ID, "term_name" for using Term Name (the first 15 characters), "both" for using both of Term ID and Name (the first 15 characters), and "full_term_name" for using the full Term Name
wrap.width	a positive integer specifying wrap width of Term Name
graph.node.attrs	a list of global node attributes. These node attributes will be changed globally. See 'Note' below for details on the attributes

`graph.edge.attrs`
: a list of global edge attributes. These edge attributes will be changed globally. See 'Note' below for details on the attributes

`node.attrs`
: a list of local edge attributes. These node attributes will be changed locally; as such, for each attribute, the input value must be a named vector (i.e. using Term ID as names). See 'Note' below for details on the attributes

`output.format`
: the format specifying the return value. It can be "Ragraph" (by default) or "igraph"

Value

An object of class 'Ragraph' or 'igraph'. If the returned is an Ragraph object, an image will be shown. If the returned is an igraph object, no image will be shown; in this case, the returned igraph object stores ontology terms used to annotate the query, including a new node attribute 'inherited' indicative of whether terms are inherited or not.

Note

A list of global node attributes used in "graph.node.attrs":

- "shape": the shape of the node: "circle", "rectangle", "rect", "box" and "ellipse"
- "fixedsize": the logical to use only width and height attributes. By default, it sets to true for not expanding for the width of the label
- "fillcolor": the background color of the node
- "color": the color for the node, corresponding to the outside edge of the node
- "fontcolor": the color for the node text/labelings
- "fontsize": the font size for the node text/labelings
- "height": the height (in inches) of the node: 0.5 by default
- "width": the width (in inches) of the node: 0.75 by default
- "style": the line style for the node: "solid", "dashed", "dotted", "invis" and "bold"

A list of global edge attributes used in "graph.edge.attrs":

- "color": the color of the edge: gray by default
- "weight": the weight of the edge: 1 by default
- "style": the line style for the edge: "solid", "dashed", "dotted", "invis" and "bold"

A list of local node attributes used in "node.attrs" (only those named Term IDs will be changed locally!):

- "label": a named vector specifying the node text/labelings
- "shape": a named vector specifying the shape of the node: "circle", "rectangle", "rect", "box" and "ellipse"
- "fixedsize": a named vector specifying whether it sets to true for not expanding for the width of the label
- "fillcolor": a named vector specifying the background color of the node

- "color": a named vector specifying the color for the node, corresponding to the outside edge of the node
- "fontcolor": a named vector specifying the color for the node text/labelings
- "fontsize": a named vector specifying the font size for the node text/labelings
- "height": a named vector specifying the height (in inches) of the node: 0.5 by default
- "width": a named vector specifying the width (in inches) of the node: 0.75 by default
- "style": a named vector specifying the line style for the node: "solid", "dashed", "dotted", "invis" and "bold"

See Also

xSocialiserGenes, xSocialiserSNPs

Examples

```
## Not run:
# Load the library
library(XGR)
RData.location="~/Sites/SVN/github/bigdata"

# 1) SNP-based similarity analysis using GWAS Catalog traits (mapped to EF)
# provide genes and SNPs reported in AS GWAS studies
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase')
## get lead SNPs reported in AS GWAS
example.snps <- names(ImmunoBase$AS$variants)
SNP.g <- xSocialiserSNPs(example.snps, include.LD=NA,
RData.location=RData.location)

# 2) Circos plot involving nodes 'rs6871626'
xCircos(g=SNP.g, entity="SNP", nodes.query="rs6871626",
RData.location=RData.location)

# 3) DAG plot visualising terms used to annotate an SNP 'rs6871626'
agDAG <- xSocialiserDAGplot(g=SNP.g, query='rs6871626', displayBy="IC",
node.info=c("full_term_name"))
## modify node labels
xSocialiserDAGplot(g=SNP.g, query='rs6871626', displayBy="IC",
node.info=c("full_term_name"),
graph.node.attrs=list(fontsize=20,fontcolor="blue",color="transparent"))

# 4) Return an igraph object storing ontology terms used to annotate an SNP 'rs6871626'
dag <- xSocialiserDAGplot(g=SNP.g, query='rs6871626', displayBy="IC",
output.format="igraph")

## End(Not run)
```

xSocialiserDAGplotAdv *Function to draw DAG plot for comparing two sets of terms used to annotate two SNPs or genes in query*

Description

xSocialiserDAGplotAdv is supposed to use DAG plot for comparing two sets of terms used to annotate two queried SNPs or genes (usually predicted to be similar). Per term, comparative results are coded in the form of 'x1-x2', where x1 is for query 1 and x2 for query 2 (the value for x1 or x2 can be '0' encoding for no anntation, '1' for inherited annotation, '2' for direct annotation). It returns an object of class 'Ragraph' or class 'igraph'.

Usage

```
xSocialiserDAGplotAdv(g, query1, query2, displayBy = c("IC", "none"),
path.mode = c("all_paths", "shortest_paths", "all_shortest_paths"),
height = 7, width = 7, margin = rep(0.1, 4), colormap = c("wyr",
"bwr", "jet", "gbr", "yr", "br", "rainbow", "wb",
"lightyellow-orange"),
ncolors = 40, zlim = NULL, colorbar = T, colorbar.fraction = 0.1,
newpage = T, layout.orientation = c("top_bottom", "left_right",
"bottom_top", "right_left"), node.info = c("term_name", "term_id"),
wrap.width = NULL, graph.node.attrs = NULL, graph.edge.attrs = NULL,
node.attrs = NULL, output.format = c("Ragraph", "igraph"))
```

Arguments

g	an object of class "igraph" (resulting from similarity analysis)
query1	the first object in query (for example, an SNP or Gene)
query2	the second object in query (for example, an SNP or Gene)
displayBy	which statistics will be used for displaying. It can be "IC" for information content (by default), "none" for no color-coding on nodes/terms
path.mode	the mode of paths induced by nodes/terms. 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)
height	a numeric value specifying the height of device
width	a numeric value specifying the width of device
margin	margins as units of length 4 or 1
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color

names, e.g. "lightyellow-orange" (by default), "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names

`ncolors` the number of colors specified over the colormap

`zlim` the minimum and maximum z/data values for which colors should be plotted, defaulting to the range of the finite values of z. Each of the given colors will be used to color an equispaced interval of this range. The midpoints of the intervals cover the range, so that values just outside the range will be plotted

`colorbar` logical to indicate whether to append a colorbar. If data is null, it always sets to false

`colorbar.fraction` the relative fraction of colorbar block against the device size

`newpage` logical to indicate whether to open a new page. By default, it sets to true for opening a new page

`layout.orientation` the orientation of the DAG layout. It can be one of "left_right" for the left-right layout (viewed from the DAG root point), "top_bottom" for the top-bottom layout, "bottom_top" for the bottom-top layout, and "right_left" for the right-left layout

`node.info` tells the ontology term information used to label nodes. It can be "term_id" for using Term ID, "term_name" for using Term Name

`wrap.width` a positive integer specifying wrap width of Term Name

`graph.node.attrs` a list of global node attributes. These node attributes will be changed globally. See 'Note' below for details on the attributes

`graph.edge.attrs` a list of global edge attributes. These edge attributes will be changed globally. See 'Note' below for details on the attributes

`node.attrs` a list of local edge attributes. These node attributes will be changed locally; as such, for each attribute, the input value must be a named vector (i.e. using Term ID as names). See 'Note' below for details on the attributes

`output.format` the format specifying the return value. It can be "Ragraph" (by default) or "igraph"

Value

An object of class 'Ragraph' or 'igraph'. If the returned is an Ragraph object, an image will be shown. If the returned is an igraph object, no image will be shown; in this case, the returned igraph object stores ontology terms used to annotate the query, including a new node attribute 'code' indicative of how terms are shared or unique to two queries (in the form of 'x1-x2', x1 for query 1 and x2 for query 2, x1 or x2 can be '0' for no anntation, '1' for inherited annotation, '2' for direct annotation).

Note

A list of global node attributes used in "graph.node.attrs":

- "shape": the shape of the node: "circle", "rectangle", "rect", "box" and "ellipse"
- "fixedsize": the logical to use only width and height attributes. By default, it sets to true for not expanding for the width of the label
- "fillcolor": the background color of the node
- "color": the color for the node, corresponding to the outside edge of the node
- "fontcolor": the color for the node text/labelings
- "fontsize": the font size for the node text/labelings
- "height": the height (in inches) of the node: 0.5 by default
- "width": the width (in inches) of the node: 0.75 by default
- "style": the line style for the node: "solid", "dashed", "dotted", "invis" and "bold"

A list of global edge attributes used in "graph.edge.attrs":

- "color": the color of the edge: gray by default
- "weight": the weight of the edge: 1 by default
- "style": the line style for the edge: "solid", "dashed", "dotted", "invis" and "bold"

A list of local node attributes used in "node.attrs" (only those named Term IDs will be changed locally!):

- "label": a named vector specifying the node text/labelings
- "shape": a named vector specifying the shape of the node: "circle", "rectangle", "rect", "box" and "ellipse"
- "fixedsize": a named vector specifying whether it sets to true for not expanding for the width of the label
- "fillcolor": a named vector specifying the background color of the node
- "color": a named vector specifying the color for the node, corresponding to the outside edge of the node
- "fontcolor": a named vector specifying the color for the node text/labelings
- "fontsize": a named vector specifying the font size for the node text/labelings
- "height": a named vector specifying the height (in inches) of the node: 0.5 by default
- "width": a named vector specifying the width (in inches) of the node: 0.75 by default
- "style": a named vector specifying the line style for the node: "solid", "dashed", "dotted", "invis" and "bold"

See Also

xSocialiserGenes, xSocialiserSNPs, xSocialiserDAGplot

Examples

```
## Not run:
# Load the library
library(XGR)
RData.location="~/Sites/SVN/github/bigdata"

# 1) SNP-based similarity analysis using GWAS Catalog traits (mapped to EF)
# provide genes and SNPs reported in AS GWAS studies
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase')
## get lead SNPs reported in AS GWAS
example.snps <- names(ImmunoBase$AS$variants)
SNP.g <- xSocialiserSNPs(example.snps, include.LD=NA,
RData.location=RData.location)

# 2) Circos plot involving nodes 'rs6871626'
xCircos(g=SNP.g, entity="SNP", nodes.query="rs6871626",
RData.location=RData.location)

# 3) DAG plot visualising terms used to annotate an SNP
## 3a) for 'rs6871626'
xSocialiserDAGplot(g=SNP.g, query='rs6871626', displayBy="IC",
node.info=c("term_name"),
graph.node.attrs=list(fontsize=20,fontcolor="blue",color="transparent"))
## 3b) for 'rs1250550'
xSocialiserDAGplot(g=SNP.g, query='rs1250550', displayBy="IC",
node.info=c("term_name"),
graph.node.attrs=list(fontsize=20,fontcolor="blue",color="transparent"))

# 4) DAG plot comparing two sets of terms used to annotate two queried SNPs
xSocialiserDAGplotAdv(g=SNP.g, query1='rs6871626', query2='rs1250550',
node.info=c("term_name"),
graph.node.attrs=list(fontsize=25,fontcolor="blue",color="transparent"))

# 5) Return an igraph object storing ontology terms used to annotate an SNP 'rs6871626'
dag <- xSocialiserDAGplotAdv(g=SNP.g, query1='rs6871626',
query2='rs1250550', output.format="igraph")

## End(Not run)
```

`xSocialiserGenes` *Function to calculate pair-wise semantic similarity given a list of genes and the ontology in query*

Description

`xSocialiserGenes` is supposed to calculate pair-wise semantic similarity between a list of input genes and the ontology in query. It returns an object of class "igraph", a network representation of socialized genes. Now it supports enrichment analysis using a wide variety of ontologies such as Gene Ontology and Phenotype Ontologies. It first calculates semantic similarity between terms

and then derives semantic similarity from term-term semantic similarity. Parallel computing is also supported for Linux or Mac operating systems.

Usage

```
xSocialiserGenes(data, ontology = c("GOBP", "GOMF", "GOCC", "DO",
"HPPA",
"HPMI", "HPCM", "HPMA", "MP"), measure = c("BM.average", "BM.max",
"BM.complete", "average", "max"), method.term = c("Resnik", "Lin",
"Schlicker", "Jiang", "Pesquita"), rescale = TRUE, force = TRUE,
fast = TRUE, parallel = TRUE, multicores = NULL,
path.mode = c("all_paths", "shortest_paths", "all_shortest_paths"),
true.path.rule = T, verbose = T,
RData.location =
"https://github.com/hfang-bristol/RDataCentre/blob/master/Portal")
```

Arguments

data	an input vector. It contains a list of Gene Symbols of interest
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, "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
measure	the measure used to derive semantic similarity between genes/SNPs from semantic similarity between terms. Take the semantic similartity between SNPs as an example. It can be "average" for average similarity between any two terms (one from SNP 1, the other from SNP 2), "max" for the maximum similarity between any two terms, "BM.average" for best-matching (BM) based average similarity (i.e. for each term of either SNP, first calculate maximum similarity to any term in the other SNP, then take average of maximum similarity; the final BM-based average similiary is the pre-calculated average between two SNPs in pair), "BM.max" for BM based maximum similarity (i.e. the same as "BM.average", but the final BM-based maximum similiary is the maximum of the pre-calculated average between two SNPs in pair), "BM.complete" for BM-based complete-linkage similarity (inspired by complete-linkage concept: the least of any maximum similarity between a term of one SNP and a term of the other SNP). When comparing BM-based similarity between SNPs, "BM.average" and "BM.max" are sensitive to the number of terms involved; instead, "BM.complete" is much robust in this aspect. By default, it uses "BM.average"
method.term	the method used to measure semantic similarity between terms. It can be "Resnik" for information content (IC) of most informative common ancestor (MICA) (see http://dl.acm.org/citation.cfm?id=1625914), "Lin" for 2*IC at MICA divided by the sum of IC at pairs of terms, "Schlicker" for weighted version of 'Lin' by the 1-prob(MICA) (see http://www.ncbi.nlm.nih.gov/pubmed/16776819), "Jiang" for 1 - difference between the sum of IC at pairs of terms and

2*IC at MICA (see http://arxiv.org/pdf/cmp-lg/9709008.pdf), "Pesquita" for graph information content similarity related to Tanimoto-Jacard index (ie. summed information content of common ancestors divided by summed information content of all ancestors of term1 and term2 (see http://www.ncbi.nlm.nih.gov/pubmed/18460186))

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

`force` logical to indicate whether the only most specific terms (for each SNP) will be used. By default, it sets to true. It is always advisable to use this since it is computationally fast but without compromising accuracy (considering the fact that true-path-rule has been applied when running xDAGanno)

`fast` logical to indicate whether a vectorised fast computation is used. By default, it sets to true. It is always advisable to use this vectorised fast computation; since the conventional computation is just used for understanding scripts

`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. It can be installed via: `source("http://bioconductor.org/biocLite.R"); biocLite(c("foreach","doMC"))`. If not yet installed, this option will be disabled

`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

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

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

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

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

Value

It returns an object of class "igraph", with nodes for input genes and edges for pair-wise semantic similarity between them. Also added graph attribute is 'dag' storing the annotated ontology DAG used. If no similarity is calculuated, it returns NULL.

Note

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

See Also

xSocialiser

Examples

```
## Not run:
# Load the library
library(XGR)
RData.location="~/Sites/SVN/github/RDataCentre/Portal"

# Gene-based similarity analysis using Mammalian Phenotype Ontology (MP)
# a) provide the input Genes of interest (eg 100 randomly chosen human genes)
## load human genes
org.Hs.eg <- xRDataLoader(RData='org.Hs.eg')
data <- as.character(sample(org.Hs.eg$gene_info$Symbol, 100))
data

# b) perform similarity analysis
sim <- xSocialiserGenes(data=data, ontology="MP",
RData.location=RData.location)

# c) save similarity results to the file called 'MP_similarity.txt'
output <- igraph::get.data.frame(sim, what="edges")
utils::write.table(output, file="MP_similarity.txt", sep="\t",
row.names=FALSE)

# d) visualise the gene network
## extract edge weight (with 2-digit precision)
x <- signif(as.numeric(E(sim)$weight), digits=2)
## rescale into an interval [1,4] as edge width
edge.width <- 1 + (x-min(x))/(max(x)-min(x))*3
## do visualisation
xVisNet(g=sim, vertex.shape="sphere", edge.width=edge.width,
edge.label=x, edge.label.cex=0.7)

## End(Not run)
```

---

xSocialiserNetplot *Function to visualise terms used to annotate an input SNP or gene using different network layouts*

Description

xSocialiserNetplot is supposed to visualise terms used to annotate an input SNP or gene using different network layouts. It returns an object of class 'igraph'.

Usage

```
xSocialiserNetplot(g, query, displayBy = c("IC", "none"),
path.mode = c("all_paths", "shortest_paths", "all_shortest_paths"),
node.info = c("none", "term_id", "term_name", "both",
"full_term_name"),
wrap.width = 15, colormap = c("yr", "jet", "gbr", "wyr", "br", "bwr",
"rainbow", "wb"), ncolors = 40, zlim = NULL, colorbar = T,
newpage = T, glayout = layout_as_tree, vertex.frame.color = NA,
vertex.size = NULL, vertex.color = NULL, vertex.shape = NULL,
vertex.label = NULL, vertex.label.cex = NULL, vertex.label.dist = 0.3,
vertex.label.color = "blue", edge.arrow.size = 0.3, ...)
```

Arguments

g	an object of class "igraph" (resulting from similarity analysis)
query	an object in query (for example, an SNP or Gene)
displayBy	which statistics will be used for displaying. It can be "IC" for information content (by default), "none" for no color-coding on nodes/terms
path.mode	the mode of paths induced by nodes in query. 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)
node.info	tells the ontology term information used to label nodes. It can be one of "none" for no node labeling, "term_id" for using Term ID, "term_name" for using Term Name, "both" for using both of Term ID and Name (the first 15 characters), and "full_term_name" for using the full Term Name
wrap.width	a positive integer specifying wrap width of Term Name. By default, first 15 characters
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
ncolors	the number of colors specified over the colormap
zlim	the minimum and maximum z/patttern values for which colors should be plotted, defaulting to the range of the finite values of z. Each of the given colors will be used to color an equispaced interval of this range. The midpoints of the intervals cover the range, so that values just outside the range will be plotted
colorbar	logical to indicate whether to append a colorbar. If pattern is null, it always sets to false
newpage	logical to indicate whether to open a new page. By default, it sets to true for opening a new page

`glayout` either a function or a numeric matrix configuring how the vertices will be placed on the plot. If layout is a function, this function will be called with the graph as the single parameter to determine the actual coordinates. This function can be one of "layout_nicely" (previously "layout.auto"), "layout_randomly" (previously "layout.random"), "layout_in_circle" (previously "layout.circle"), "layout_on_sphere" (previously "layout.sphere"), "layout_with_fr" (previously "layout.fruchterman.reingold"), "layout_with_kk" (previously "layout.kamada.kawai"), "layout_as_tree" (previously "layout.reingold.tilford"), "layout_with_lgl" (previously "layout.lgl"), "layout_with_graphopt" (previously "layout.graphopt"), "layout_with_sugiyama" (previously "layout.kamada.kawai"), "layout_with_dh" (previously "layout.davidson.harel"), "layout_with_drl" (previously "layout.drl"), "layout_with_gem" (previously "layout.gem"), "layout_with_mds". A full explanation of these layouts can be found in http://igraph.org/r/doc/layout_nicely.html

`vertex.frame.color` the color of the frame of the vertices. If it is NA, then there is no frame

`vertex.size` the size of each vertex. If it is a vector, each vertex may differ in size

`vertex.color` the fill color of the vertices. If it is NA, then there is no fill color. If the pattern is given, this setup will be ignored

`vertex.shape` the shape of each vertex. It can be one of "circle", "square", "csquare", "rectangle", "crectangle", "vrectangle", "pie" (http://igraph.org/r/doc/vertex.shape.pie.html), "sphere", and "none". If it sets to NULL, these vertices with negative will be "csquare" and the rest "circle".

`vertex.label` the label of the vertices. If it is NA, then there is no label. The default vertex labels are the name attribute of the nodes

`vertex.label.cex` the font size of vertex labels.

`vertex.label.dist` the distance of the label from the center of the vertex. If it is 0 then the label is centered on the vertex. If it is 1 then the label is displayed beside the vertex.

`vertex.label.color` the color of vertex labels.

`edge.arrow.size` the size of the arrows for the directed edge. The default value is 1.

`...` additional graphic parameters. See http://igraph.org/r/doc/plot.common.html for the complete list.

Value

an igraph object to represent DAG, appended with a node attribute called 'inherited' indicative of whether terms are inherited or not

Note

none

See Also

xSocialiserGenes, xSocialiserSNPs

Examples

```
## Not run:
# Load the library
library(XGR)
RData.location="~/Sites/SVN/github/bigdata"

# 1) SNP-based similarity analysis using GWAS Catalog traits (mapped to EF)
# provide genes and SNPs reported in AS GWAS studies
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase')
## get lead SNPs reported in AS GWAS
example.snps <- names(ImmunoBase$AS$variants)
SNP.g <- xSocialiserSNPs(example.snps, include.LD=NA,
RData.location=RData.location)

# 2) Circos plot involving nodes 'rs6871626'
xCircos(g=SNP.g, entity="SNP", nodes.query="rs6871626",
RData.location=RData.location)

# 3) Net plot visualising terms used to annotate an SNP 'rs6871626'
dag <- xSocialiserNetplot(g=SNP.g, query='rs6871626', displayBy="IC",
node.info=c("none"), vertex.label=NA, wrap.width=30)

## End(Not run)
```

---

xSocialiserSNPs *Function to calculate pair-wise semantic similarity given a list of SNPs and the ontology in query*

Description

xSocialiserSNPs is supposed to calculate pair-wise semantic similarity between a list of input SNPs and the ontology in query. It returns an object of class "igraph", a network representation of socialized SNPs. Now it supports analysis for SNPs using GWAS Catalog traits mapped to Experimental Factor Ontology. If required, additional SNPs that are in linkage disequilibrium (LD) with input SNPs are also be used for calculation. It first calculates semantic similarity between terms and then derives semantic similarity from term-term semantic similarity. Parallel computing is also supported for Linux or Mac operating systems.

Usage

```
xSocialiserSNPs(data, ontology = c("EF", "EF_disease", "EF_phenotype",
"EF_bp"), include.LD = NA, LD.r2 = 0.8, measure = c("BM.average",
"BM.max", "BM.complete", "average", "max"), method.term = c("Resnik",
"Lin",
```

```
"Schlicker", "Jiang", "Pesquita"), rescale = TRUE, force = TRUE,
fast = TRUE, parallel = TRUE, multicores = NULL,
path.mode = c("all_paths", "shortest_paths", "all_shortest_paths"),
true.path.rule = T, verbose = T,
RData.location =
"https://github.com/hfang-bristol/RDataCentre/blob/master/Portal")
```

Arguments

data	an input vector. It contains a list of SNPs of interest
ontology	the ontology supported currently. Now it is only "EF" for Experimental Factor Ontology (used to annotate GWAS Catalog SNPs). However, there are several subparts of this ontology to choose: 'EF_disease' for the subpart under the term 'disease' (EFO:0000408), 'EF_phenotype' for the subpart under the term 'phenotype' (EFO:0000651), 'EF_bp' for the subpart under the term 'biological process' (GO:0008150)
include.LD	additional SNPs in LD with input 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.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
measure	the measure used to derive semantic similarity between genes/SNPs from semantic similarity between terms. Take the semantic similartity between SNPs as an example. It can be "average" for average similarity between any two terms (one from SNP 1, the other from SNP 2), "max" for the maximum similarity between any two terms, "BM.average" for best-matching (BM) based average similarity (i.e. for each term of either SNP, first calculate maximum similarity to any term in the other SNP, then take average of maximum similarity; the final BM-based average similiary is the pre-calculated average between two SNPs in pair), "BM.max" for BM based maximum similarity (i.e. the same as "BM.average", but the final BM-based maximum similiary is the maximum of the pre-calculated average between two SNPs in pair), "BM.complete" for BM-based complete-linkage similarity (inspired by complete-linkage concept: the least of any maximum similarity between a term of one SNP and a term of the other SNP). When comparing BM-based similarity between SNPs, "BM.average" and "BM.max" are sensitive to the number of terms involved; instead, "BM.complete" is much robust in this aspect. By default, it uses "BM.average"
method.term	the method used to measure semantic similarity between terms. It can be "Resnik" for information content (IC) of most informative common ancestor (MICA) (see http://dl.acm.org/citation.cfm?id=1625914), "Lin" for 2*IC at MICA divided by the sum of IC at pairs of terms, "Schlicker" for weighted version

	of 'Lin' by the 1-prob(MICA) (see http://www.ncbi.nlm.nih.gov/pubmed/16776819), "Jiang" for 1 - difference between the sum of IC at pairs of terms and 2*IC at MICA (see http://arxiv.org/pdf/cmp-lg/9709008.pdf), "Pesquita" for graph information content similarity related to Tanimoto-Jacard index (ie. summed information content of common ancestors divided by summed information content of all ancestors of term1 and term2 (see http://www.ncbi.nlm.nih.gov/pubmed/18460186))
rescale	logical to indicate whether the resulting values are rescaled to the range [0,1]. By default, it sets to true
force	logical to indicate whether the only most specific terms (for each SNP) will be used. By default, it sets to true. It is always advisable to use this since it is computationally fast but without compromising accuracy (considering the fact that true-path-rule has been applied when running xDAGanno)
fast	logical to indicate whether a vectorised fast computation is used. By default, it sets to true. It is always advisable to use this vectorised fast computation; since the conventional computation is just used for understanding scripts
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. It can be installed via: `source("http://bioconductor.org/biocLite.R"); biocLite(c("foreach","doMC"))`. If not yet installed, this option will be disabled
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
path.mode	the mode of paths induced by vertices/nodes with input annotation data. It can be "all_paths" for all possible paths to the root, "shortest_paths" for only one path to the root (for each node in query), "all_shortest_paths" for all shortest paths to the root (i.e. for each node, find all shortest paths with the equal lengths)
true.path.rule	logical to indicate whether the true-path rule should be applied to propagate annotations. By default, it sets to true
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to false for no display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details

Value

It returns an object of class "igraph", with nodes for input SNPs and edges for pair-wise semantic similarity between them. Also added graph attribute is 'dag' storing the annotated ontology DAG used. If no similarity is calculuated, it returns NULL.

Note

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

subgraph.

See Also

xSocialiser

Examples

```
## Not run:
# Load the library
library(XGR)
RData.location="~/Sites/SVN/github/RDataCentre/Portal"

# SNP-based similarity analysis using GWAS Catalog traits (mapped to EF)
# a) provide the input SNPs of interest (eg 8 randomly chosen SNPs)
anno <- xRDataLoader(RData='GWAS2EF')
allSNPs <- rownames(anno)
data <- sample(allSNPs,8)
data

# b) perform similarity analysis
sim <- xSocialiserSNPs(data=data, RData.location=RData.location)

# b') optionally, enrichment analysis for input SNPs plus their LD SNPs
## LD based on European population (EUR) with r2>=0.8
#sim <- xSocialiserSNPs(data=data, include.LD="EUR", LD.r2=0.8, RData.location=RData.location)

# c) save similarity results to the file called 'EF_similarity.txt'
output <- igraph::get.data.frame(sim, what="edges")
utils::write.table(output, file="EF_similarity.txt", sep="\t",
row.names=FALSE)

# d) visualise the SNP network
## extract edge weight (with 2-digit precision)
x <- signif(as.numeric(E(sim)$weight), digits=2)
## rescale into an interval [1,4] as edge width
edge.width <- 1 + (x-min(x))/(max(x)-min(x))*3
## do visualisation
xVisNet(g=sim, vertex.shape="sphere", edge.width=edge.width,
edge.label=x, edge.label.cex=0.7)

## End(Not run)
```

---

xSparseMatrix *Function to create a sparse matrix for an input file with three columns*

Description

xSparseMatrix is supposed to create a sparse matrix for an input file with three columns.

Usage

```
xSparseMatrix(input.file, rows = NULL, columns = NULL, verbose = T)
```

Arguments

input.file	an input file containing three columns: 1st column for rows, 2nd for columns, and 3rd for numeric values. Alternatively, the input.file can be a matrix or data frame, assuming that input file has been read. Note: the file should use the tab delimiter as the field separator between columns
rows	a vector specifying row names. By default, it is NULL
columns	a vector specifying column names. By default, it is NULL
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to TRUE for display

Value

an object of the dgCMatrix class (a sparse matrix)

Note

If rows (or columns) are not NULL, the rows (or columns) of resulting sparse matrix will be union of those from input.file and those from rows (or columns). None

See Also

xSparseMatrix

Examples

```
# create a sparse matrix of 4 X 2
input.file <- rbind(c('R1','C1',1), c('R2','C1',1), c('R2','C2',1),
c('R3','C2',1), c('R4','C1',1))
res <- xSparseMatrix(input.file)
res
# get a full matrix
as.matrix(res)

res <- xSparseMatrix(input.file, columns=c('C1','C2','C3'))
res
```

xSubneterGenes *Function to identify a subnetwork from an input network and the significance level imposed on its nodes*

Description

xSubneterGenes is supposed to identify maximum-scoring subnetwork from an input graph with the node information on the significance (measured as p-values or fdr). It returns an object of class "igraph".

Usage

```
xSubneterGenes(data, network = c("STRING_highest", "STRING_high",
"STRING_medium", "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"), network.customised = NULL, seed.genes = T,
subnet.significance = 0.01, subnet.size = NULL, verbose = T,
RData.location =
"https://github.com/hfang-bristol/RDataCentre/blob/master/Portal")
```

Arguments

data	a named input vector containing the significance level for nodes (gene symbols). For this named vector, the element names are gene symbols, the element values for the significance level (measured as p-value or fdr). Alternatively, it can be a matrix or data frame with two columns: 1st column for gene symbols, 2nd column for the significance level
network	the built-in network. Currently two sources of network information are supported: the STRING database (version 10) and the 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), and "STRING_medium" for interactions with medium confidence (confidence scores>=400). 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

`network.customised`
: an object of class "igraph". By default, it is NULL. It is designed to allow the user analysing their customised network data that are not listed in the above argument 'network'. This customisation (if provided) has the high priority over built-in network

`seed.genes`
: logical to indicate whether the identified network is restricted to seed genes (ie input genes with the signficant level). By default, it sets to true

`subnet.significance`
: the given significance threshold. By default, it is set to NULL, meaning there is no constraint on nodes/genes. If given, those nodes/genes with p-values below this are considered significant and thus scored positively. Instead, those p-values above this given significance threshold are considered insigificant and thus scored negatively

`subnet.size`
: the desired number of nodes constrained to the resulting subnet. It is not nulll, a wide range of significance thresholds will be scanned to find the optimal significance threshold leading to the desired number of nodes in the resulting subnet. Notably, the given significance threshold will be overwritten by this option

`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"

Note

The algorithm identifying a subnetwork is implemented in the dnet package (http://genomemedicine.biomedcentral.com/artic
014-0064-8). In brief, from an input network with input node/gene information (the significant level; p-values or FDR), the way of searching for a maximum-scoring subnetwork is done as follows. Given the threshold of tolerable p-value, it gives positive scores for nodes with p-values below the threshold (nodes of interest), and negative scores for nodes with threshold-above p-values (intolerable). After score transformation, the search for a maximum scoring subnetwork is deduced to find the connected subnetwork that is enriched with positive-score nodes, allowing for a few

negative-score nodes as linkers. This objective is met through minimum spanning tree finding and post-processing, previously used as a heuristic solver of prize-collecting Steiner tree problem. The solver is deterministic, only determined by the given tolerable p-value threshold. For identification of the subnetwork with a desired number of nodes, an iterative procedure is also developed to fine-tune tolerable thresholds. This explicit control over the node size may be necessary for guiding follow-up experiments.

See Also

xRDataLoader

Examples

```
## Not run:
# Load the library
library(XGR)
RData.location="~/Sites/SVN/github/RDataCentre/Portal"

# a) provide the input nodes/genes with the significance info
## load human genes
org.Hs.eg <- xRDataLoader(RData='org.Hs.eg')
sig <- rbeta(500, shape1=0.5, shape2=1)
data <- data.frame(symbols=org.Hs.eg$gene_info$Symbol[1:500], sig)

# b) perform network analysis
# b1) find maximum-scoring subnet based on the given significance threshold
subnet <- xSubneterGenes(data=data, network="STRING_high",
subnet.significance=0.01)
# b2) find maximum-scoring subnet with the desired node number=50
subnet <- xSubneterGenes(data=data, network="STRING_high",
subnet.size=50, RData.location=RData.location)

# c) save subnet results to the files called 'subnet_edges.txt' and 'subnet_nodes.txt'
output <- igraph::get.data.frame(subnet, what="edges")
utils::write.table(output, file="subnet_edges.txt", sep="\t",
row.names=FALSE)
output <- igraph::get.data.frame(subnet, what="vertices")
utils::write.table(output, file="subnet_nodes.txt", sep="\t",
row.names=FALSE)

# d) visualise the identified subnet
## do visualisation with nodes colored according to the significance (you provide)
xVisNet(g=subnet, pattern=-log10(as.numeric(V(subnet)$significance)),
vertex.shape="sphere", colormap="wyr")
## do visualisation with nodes colored according to transformed scores
xVisNet(g=subnet, pattern=V(subnet)$score, vertex.shape="sphere")

# e) visualise the identified subnet as a circos plot
library(RCircos)
xCircos(g=subnet, entity="Gene")

## End(Not run)
```

xSubneterSNPs *Function to identify a gene network from an input network given a list of seed SNPs together with the significance level (e.g. GWAS reported p-values)*

Description

xSubneterSNPs is supposed to identify maximum-scoring gene subnetwork from an input graph with the node information on the significance (measured as p-values or fdr). To do so, it defines seed genes and their scores that take into account the distance to and the significance of input SNPs. It returns an object of class "igraph".

Usage

```
xSubneterSNPs(data, include.LD = NA, LD.customised = NULL, LD.r2 = 0.8,
significance.threshold = 5e-05, distance.max = 2e+05,
decay.kernel = c("slow", "linear", "rapid"), decay.exponent = 2,
GR.SNP = "dbSNP_GWAS", GR.Gene = "UCSC_genes", scoring.scheme =
c("max",
"sum", "sequential"), network = c("STRING_highest", "STRING_high",
"STRING_medium", "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"), network.customised = NULL, seed.genes = T,
subnet.significance = 5e-05, subnet.size = NULL, verbose = T,
RData.location =
"https://github.com/hfang-bristol/RDataCentre/blob/master/Portal")
```

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 precalcuated LD info. This customisation (if provided) has the high priority over built-in LD SNPs

`LD.r2` the LD r2 value. By default, it is 0.8, meaning that SNPs in LD (r2>=0.8) with input SNPs will be considered as LD SNPs. It can be any value from 0.8 to 1

`significance.threshold` the given significance threshold. By default, it is set to NULL, meaning there is no constraint on the significance level when transforming the significance level of SNPs into scores. If given, those SNPs below this are considered significant and thus scored positively. Instead, those above this are considered insigificant and thus receive no score

`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_genes', that is, UCSC known canonical genes (together with genomic locations) based on human genome assembly hg19. Even 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"

`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), and "STRING_medium"

for interactions with medium confidence (confidence scores>=400). 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

`network.customised`
: an object of class "igraph". By default, it is NULL. It is designed to allow the user analysing their customised network data that are not listed in the above argument 'network'. This customisation (if provided) has the high priority over built-in network

`seed.genes`
: logical to indicate whether the identified network is restricted to seed genes (ie nearby genes that are located within defined distance window centred on lead or LD SNPs). By default, it sets to true

`subnet.significance`
: the given significance threshold. By default, it is set to NULL, meaning there is no constraint on nodes/genes. If given, those nodes/genes with p-values below this are considered significant and thus scored positively. Instead, those p-values above this given significance threshold are considered insigificant and thus scored negatively

`subnet.size`
: the desired number of nodes constrained to the resulting subnet. It is not nulll, a wide range of significance thresholds will be scanned to find the optimal significance threshold leading to the desired number of nodes in the resulting subnet. Notably, the given significance threshold will be overwritten by this option

`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: significance, score

Note

The algorithm identifying a gene subnetwork that is likely modulated by input SNPs and/or their LD SNPs includes two major steps. The first step is to use xSNP2GeneScores for defining and scoring nearby genes that are located within distance window of input and/or LD SNPs. The second step is to use xSubneterGenes for identifying a maximum-scoring gene subnetwork that contains as many highly scored genes as possible but a few less scored genes as linkers.

See Also

xSNP2GeneScores, xSubneterGenes

Examples

```
## Not run:
# Load the library
library(XGR)
RData.location="~/Sites/SVN/github/RDataCentre/Portal"

# a) provide the seed SNPs with the weight info
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase')
## get lead SNPs reported in AS GWAS and their significance info (p-values)
gr <- ImmunoBase$AS$variant
data <- GenomicRanges::mcols(gr)[,c(1,3)]

# b) perform network analysis
# b1) find maximum-scoring subnet based on the given significance threshold
subnet <- xSubneterSNPs(data=data, network="STRING_high", seed.genes=F,
subnet.significance=0.01, RData.location=RData.location)
# b2) find maximum-scoring subnet with the desired node number=30
subnet <- xSubneterSNPs(data=data, network="STRING_high", seed.genes=F,
subnet.size=30)

# c) save subnet results to the files called 'subnet_edges.txt' and 'subnet_nodes.txt'
output <- igraph::get.data.frame(subnet, what="edges")
utils::write.table(output, file="subnet_edges.txt", sep="\t",
row.names=FALSE)
output <- igraph::get.data.frame(subnet, what="vertices")
utils::write.table(output, file="subnet_nodes.txt", sep="\t",
row.names=FALSE)

# d) visualise the identified subnet
## do visualisation with nodes colored according to the significance
xVisNet(g=subnet, pattern=-log10(as.numeric(V(subnet)$significance)),
vertex.shape="sphere", colormap="wyr")
## do visualisation with nodes colored according to transformed scores
xVisNet(g=subnet, pattern=V(subnet)$score, vertex.shape="sphere")

# e) visualise the identified subnet as a circos plot
library(RCircos)
xCircos(g=subnet, entity="Gene", RData.location=RData.location)
```

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

`xVisKernels` *Function to visualise distance kernel functions*

Description

`xVisKernels` is supposed to visualise distance kernels, each of which is a decaying function of: i) the relative distance d_{gs} between the gene g and the SNP s, and ii) the decay exponent λ.

Usage

```
xVisKernels(exponent = 2, newpage = T)
```

Arguments

`exponent`	an integer specifying decay exponent. By default, it sets to 2
`newpage`	logical to indicate whether to open a new page. By default, it sets to true for opening a new page

Value

invisible

Note

There are five kernels that are currently supported:

- For "slow decay" kernel, $h_{ds}(t) = 1 - d_{gs}/D\lambda * (d_{gs} \leq D)$
- For "linear decay" kernel, $h_{ds}(t) = 1 - d_{gs}/D * (d_{gs} \leq D)$
- For "rapid decay" kernel, $h_{ds}(t) = 1 - d_{gs}/D^{\lambda} * (d_{gs} \leq D)$

See Also

`xSNP2nGenes`

Examples

```
# visualise distance kernels
xVisKernels(exponent=2)
xVisKernels(exponent=3)
```

xVisNet *Function to visualise a graph object of class "igraph"*

Description

xVisNet is supposed to visualise a graph object of class "igraph". It also allows vertices/nodes color-coded according to the input pattern.

Usage

```
xVisNet(g, pattern = NULL, colormap = c("yr", "jet", "gbr", "wyr",
"br",
"bwr", "rainbow", "wb"), ncolors = 40, zlim = NULL, colorbar = T,
newpage = T, glayout = layout_with_kk, vertex.frame.color = NA,
vertex.size = NULL, vertex.color = NULL, vertex.shape = NULL,
vertex.label = NULL, vertex.label.cex = NULL, vertex.label.dist = 0.3,
vertex.label.color = "blue", edge.arrow.size = 0.3, ...)
```

Arguments

g	an object of class "igraph"
pattern	a numeric vector used to color-code vertices/nodes. Notably, if the input vector contains names, then these names should include all node names of input graph, i.e. V(g)$name, since there is a mapping operation. After mapping, the length of the patern vector should be the same as the number of nodes of input graph; otherwise, this input pattern will be ignored. The way of how to color-code is to map values in the pattern onto the whole colormap (see the next arguments: colormap, ncolors, zlim and colorbar)
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
ncolors	the number of colors specified over the colormap
zlim	the minimum and maximum z/patttern values for which colors should be plotted, defaulting to the range of the finite values of z. Each of the given colors will be used to color an equispaced interval of this range. The midpoints of the intervals cover the range, so that values just outside the range will be plotted
colorbar	logical to indicate whether to append a colorbar. If pattern is null, it always sets to false
newpage	logical to indicate whether to open a new page. By default, it sets to true for opening a new page

glayout — either a function or a numeric matrix configuring how the vertices will be placed on the plot. If layout is a function, this function will be called with the graph as the single parameter to determine the actual coordinates. This function can be one of "layout_nicely" (previously "layout.auto"), "layout_randomly" (previously "layout.random"), "layout_in_circle" (previously "layout.circle"), "layout_on_sphere" (previously "layout.sphere"), "layout_with_fr" (previously "layout.fruchterman.reingold"), "layout_with_kk" (previously "layout.kamada.kawai"), "layout_as_tree" (previously "layout.reingold.tilford"), "layout_with_lgl" (previously "layout.lgl"), "layout_with_graphopt" (previously "layout.graphopt"), "layout_with_sugiyama" (previously "layout.kamada.kawai"), "layout_with_dh" (previously "layout.davidson.harel"), "layout_with_drl" (previously "layout.drl"), "layout_with_gem" (previously "layout.gem"), "layout_with_mds". A full explanation of these layouts can be found in http://igraph.org/r/doc/layout_nicely.html

vertex.frame.color — the color of the frame of the vertices. If it is NA, then there is no frame

vertex.size — the size of each vertex. If it is a vector, each vertex may differ in size

vertex.color — the fill color of the vertices. If it is NA, then there is no fill color. If the pattern is given, this setup will be ignored

vertex.shape — the shape of each vertex. It can be one of "circle", "square", "csquare", "rectangle", "crectangle", "vrectangle", "pie" (http://igraph.org/r/doc/vertex.shape.pie.html), "sphere", and "none". If it sets to NULL, these vertices with negative will be "csquare" and the rest "circle".

vertex.label — the label of the vertices. If it is NA, then there is no label. The default vertex labels are the name attribute of the nodes

vertex.label.cex — the font size of vertex labels.

vertex.label.dist — the distance of the label from the center of the vertex. If it is 0 then the label is centered on the vertex. If it is 1 then the label is displayed beside the vertex.

vertex.label.color — the color of vertex labels.

edge.arrow.size — the size of the arrows for the directed edge. The default value is 1.

... — additional graphic parameters. See http://igraph.org/r/doc/plot.common.html for the complete list.

Value

invisible

Note

none

See Also

xSubneterGenes, xSubneterSNPs

Examples

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

# 2) visualise the graph
# 2a) visualise in one go
xVisNet(g=g, vertex.shape="sphere", glayout=layout_with_kk)
# 2b) visualise the graph with layout first calculated
glayout <- layout_(g, with_kk(), component_wise())
xVisNet(g=g, vertex.shape="sphere", glayout=glayout)
# 2c) visualise the graph with layout appended to the graph itself
g <- add_layout_(g, with_kk(), component_wise())
xVisNet(g=g, vertex.shape="sphere")

# 4) visualise the graph with vertices being color-coded by the pattern
pattern <- runif(vcount(g))
names(pattern) <- V(g)$name
xVisNet(g=g, pattern=pattern, colormap="bwr", vertex.shape="sphere")
```

Index