Package 'XGR'

August 14, 2018

Type Package

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

Version 1.1.4

Date 2018-8-14

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, plot3D, dplyr, tidyr, ggrepel, ggnetwork, XML

Suggests foreach, doParallel, akima, corrplot, gridExtra, grid, treemapify, GenomeInfoDb, GenomicScores, ggraph, ggbio, scales

Description The central goal of XGR by Fang et al. (2016) <doi:10.1186/s13073-016-0384-y> is to provide a data interpretation system necessary to do ``big data" science. 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, SNPs or genomic regions, and 4) annotation analysis for interpreting genomic regions using co-localised functional genomic annotations (such as open chromatin, epigenetic marks, TF binding sites and genomic segments) 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

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

Collate 'ClassMethod-XGR.r' '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' 'xSNPlocations.r'
'xSNP2nGenes.r' 'xSparseMatrix.r' 'xSM2DF.r'
'xSNP2GeneScores.r' 'xGRviaGeneAnno.r' 'xGRviaGenomicAnno.r'
'xGRviaGenomicAnnoAdv.r' 'xGRsampling.r' 'xLiftOver.r'
'xEnrichConciser.r' 'xColormap.r' 'xGR2nGenes.r' 'xGRscores.r'
'xGR2GeneScores.r' 'xSubneterGR.r' 'xGR.r' 'xCheckParallel.r'
'xSymbol2GeneID.r' 'xDAGpropagate.r' 'xVisInterp.r'
'xVisInterpAnimate.r' 'xDefineNet.r' 'xCombineNet.r'
'xEnrichMatrix.r' 'xVolcano.r' 'xGraphML.r' 'xSimplifyNet.r'
'xHeatmap.r' 'xCorrelation.r' 'xPolarDot.r' 'xPolarBar.r'
'xEnricherGenesAdv.r' 'xHeatmapAdv.r' 'xGScore.r'
'xGScoreAdv.r' 'xEnrichForest.r' 'xGR2xGenes.r'
'xGR2xGeneScores.r' 'xGR2xNet.r' 'xGRviaGeneAnnoAdv.r'
'xBigraph.r' 'xBiheatmap.r' 'xAddCoords.r' 'xRegress.r'
'xGraphSplit.r' 'xGGnetwork.r' 'xOBOcode.r' 'xGR2xGeneAnno.r'
'xGR2xGeneAnnoAdv.r' 'xEnrichTreemap.r' 'xCrosstalk.r'
'xGRsort.r' 'xGRsep.r' 'xGRcse.r' 'xEnrichLadder.r' 'xHEB.r'
'xEnrichHeatmap.r' 'xGRoverlap.r' 'xGRtrack.r' 'xAggregate.r'
'xRPS.r' 'xGRmanhattan.r' 'xLDblock.r' 'xLDsampling.r'
'xLDenricher.r' 'xDefineGenomicAnno.r' 'xDefineOntology.r'
'xDefineEQTL.r' 'xDefineHIC.r' 'xPCHiCplot.r' 'xBiproject.r'
'xRWkernel.r' 'xRWenricher.r' 'xGRkaryogram.r' 'xRepurpose.r'

License GPL-2

biocViews Bioinformatics

NeedsCompilation no

Repository CRAN

Date/Publication 2018-08-14 12:50:02 UTC

R topics documented:

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

Description

`aOnto` has 2 components: g, anno.

Usage

```
aOnto(g, anno)

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

Arguments

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

Value

an object of S3 class `aOnto`

Examples

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

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

## End(Not run)
```

---

bLD *Definition for S3 class* bLD

Description

bLD has 2 components: best, block.

Usage

```
bLD(best, block)

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

Arguments

best	a GR object
block	a GRL object
x	an object of class bLD
...	other parameters

Value

an object of S3 class bLD

Examples

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

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

## End(Not run)
```

cPath *Definition for S3 class* cPath

Description

cPath has 4 components: ig_paths, gp_paths, gp_heatmap, ig_subg.

Usage

```
cPath(ig_paths, gp_paths, gp_heatmap, ig_subg)

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

Arguments

ig_paths	an igraph object
gp_paths	a ggplot object
gp_heatmap	a ggplot object
ig_subg	an igraph object
x	an object of class cPath
...	other parameters

Value

an object of S3 class cPath

Examples

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

## End(Not run)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
cPath(ig_paths, gp_paths, gp_heatmap, ig_subg)

## End(Not run)
```

DR *Definition for S3 class* DR

Description

DR has 3 components: df, index, gp.

Usage

```
DR(df, index, gp)

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

Arguments

df	a data frame
index	a data frame
gp	a ggplot object
x	an object of class DR
...	other parameters

Value

an object of S3 class DR

Examples

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

## End(Not run)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
DR(df, index, gp)

## End(Not run)
```

eTerm *Definition for S3 class* eTerm

Description

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

Usage

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

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

Arguments

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

Value

an object of S3 class eTerm

Examples

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

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

## End(Not run)
```

`Haploid_regulators` *Haploid mutagenesis screens for regulators of protein phenotypes*

Description

This dataset contains regulators of 11 protein phenotypes identified by haploid mutagenesis screens.

Usage

```
data(Haploid_regulators)
```

Value

a data frame. It has the following columns: "Gene" (gene symbol; regulators), "MI" (mutation index; negative values for positive regulators and positive values for negative regulators), "FDR" and "Phenotype" (one of 11 protein phenotypes).

References

Brockmann et al. (2017). Genetic wiring maps of single-cell protein states reveal an off-switch for GPCR signalling. *Nature*, 546:307-11.
Mezzadra et al. (2017). Identification of CMTM6 and CMTM4 as PD-L1 protein regulators. *Nature*, 549:106-10.

Examples

```
## Not run:
Haploid_regulators <- xRDataLoader('Haploid_regulators')
Haploid_regulators[1:5,]
## for 'PDL1'
ind <- grepl('PDL1', Haploid_regulators$Phenotype)
df <- Haploid_regulators[ind,c('Gene','MI','FDR')]

## End(Not run)
```

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

```
## Not run:
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase')
names(ImmunoBase)
ImmunoBase$AS$disease
ImmunoBase$AS$variants
head(ImmunoBase$AS$genes_variants)
head(ImmunoBase$AS$genes_regions)

## End(Not run)
```

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

```
## Not run:
JKscience_TS2A <- xRDataLoader(RData.customised='JKscience_TS2A')
JKscience_TS2A[1:5,]

## End(Not run)
```

ls_eTerm *Definition for S3 class* ls_eTerm

Description

ls_eTerm has 3 components: df, mat and gp.

Usage

```
ls_eTerm(df, mat, gp)

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

Arguments

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

Value

an object of S3 class ls_eTerm

Examples

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

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

## End(Not run)
```

---

mSeed *Definition for S3 class* mSeed

Description

cTarget has 3 components: GR, Gene, Link.

Usage

```
mSeed(GR, Gene, Link)

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

Arguments

GR	a data frame
Gene	a data frame
Link	a data frame
x	an object of class mSeed
...	other parameters

Value

an object of S3 class mSeed

Examples

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

## End(Not run)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
mSeed(GR, Gene, Link)

## End(Not run)
```

---

xAddCoords *Function to add coordinates into a graph according to a node attribute*

Description

xAddCoords is supposed to add coordinates into a graph according to a node attribute such as community or comp.

Usage

```
xAddCoords(g, node.attr = NULL, glayout = layout_with_kk,
edge.color.alternative = c("grey70", "grey95"), seed = 825,
verbose = TRUE)
```

Arguments

g	an object of class "igraph" (or "graphNEL") for a graph with such as a 'community' node attribute
node.attr	a charatter specifying a node attribute. If NULL or no match, it returns NULL

glayout — a graph layout function. This function can be one of "layout_nicely" (previously "layout.auto"), "layout_randomly" (previously "layout.random"), "layout_in_circle" (previously "layout.circle"), "layout_on_sphere" (previously "layout.sphere"), "layout_with_fr" (previously "layout.fruchterman.reingold"), "layout_with_kk" (previously "layout.kamada.kawai"), "layout_as_tree" (previously "layout.reingold.tilford"), "layout_with_lgl" (previously "layout.lgl"), "layout_with_graphopt" (previously "layout.graphopt"), "layout_with_sugiyama" (previously "layout.kamada.kawai"), "layout_with_dh" (previously "layout.davidson.harel"), "layout_with_drl" (previously "layout.drl"), "layout_with_gem" (previously "layout.gem"), "layout_with_mds", and "layout_as_bipartite". A full explanation of these layouts can be found in http://igraph.org/r/doc/layout_nicely.html

edge.color.alternative — two alternative colors for edges within the community (grey70 by default) and edges between communities (grey95 by default)

seed — an integer specifying the seed

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 igraph object, appended by node attributes including "xcoord" for x-coordinates, "ycoord" for y-coordiates, and by edge attributes including "color" for between-community edges ('grey95') and within-community edges ('grey70').

See Also

xGGnetwork

Examples

```
# 1) generate a random bipartite graph
set.seed(123)
g <- sample_bipartite(100, 50, p=0.1)
V(g)$name <- V(g)

## Not run:
# 2) obtain and append the community
cs <- igraph::cluster_louvain(g)
V(g)$community <- cs$membership
ig <- xAddCoords(g, node.attr="community",
edge.color.alternative=c("grey50","grey95"))
gp <- xGGnetwork(ig, node.label='name', node.label.size=2,
node.label.color='black', node.label.alpha=0.8, node.label.padding=0,
node.label.arrow=0, node.label.force=0.002, node.xcoord='xcoord',
node.ycoord='ycoord', node.color='community',
node.color.title='Community', colormap='jet.both', ncolors=64,
zlim=NULL,
edge.color="color",edge.color.alpha=0.5,edge.curve=0,edge.arrow.gap=0)

## make it discrete for the colorbar
```

```
gp +
scale_colour_gradientn(colors=xColormap('jet')(64),breaks=seq(1,8)) +
guides(color=guide_legend(title="Community"))

## End(Not run)
```

---

xAggregate *Function to aggregate data respecting number of features*

Description

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

Usage

```
xAggregate(data, bin = F, nbin = 10, scale.log = T, verbose = T)
```

Arguments

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

Value

a data frame with an appended column 'Aggregate'

Note

None

See Also

xAggregate

Examples

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

RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
# HiC-gene pairs per cell types/states
g <- xRDataLoader(RData.customised='ig.PCHiC',
RData.location=RData.location)
df <- do.call(cbind, igraph::edge_attr(g))
# aggregate over cell types/states
data <- df
data[data<5] <- NA
res <- xAggregate(data)

## End(Not run)
```

---

xBigraph *Function to obtain communities from a bipartitle graph*

Description

xBigraph is supposed to obtain communities from a bipartitle graph.

Usage

```
xBigraph(g, algorithm = c("louvain", "leading_eigen", "fast_greedy",
"walktrap"), seed = 825, glayout = layout_with_kk,
edge.color.alternative = c("grey70", "grey95"), verbose = TRUE)
```

Arguments

g	an object of class "igraph" (or "graphNEL") for a bipartitel graph with a 'type' node attribute
algorithm	the algorithm to initialise community from a projected graph. It can be 'louvain' using multi-level optimization, 'leading_eigen' using leading eigenvector, 'fast_greedy' using greedy optimization, and 'walktrap' via short random walks
seed	an integer specifying the seed
glayout	a graph layout function. This function can be one of "layout_nicely" (previously "layout.auto"), "layout_randomly" (previously "layout.random"), "layout_in_circle" (previously "layout.circle"), "layout_on_sphere" (previously "layout.sphere"), "layout_with_fr" (previously "layout.fruchterman.reingold"), "layout_with_kk" (previously "layout.kamada.kawai"), "layout_as_tree" (previously "layout.reingold.tilford"), "layout_with_lgl" (previously "layout.lgl"), "layout_with_graphopt" (previously "layout.graphopt"), "layout_with_sugiyama" (previously "layout.kamada.kawai"), "layout_with_dh" (previously "layout.davidson.harel"), "layout_with_drl" (previously "layout.drl"), "layout_with_gem" (previously "layout.gem"), "layout_with_mds",

and "layout_as_bipartite". A full explanation of these layouts can be found in http://igraph.org/r/doc/layout_nicely.html

edge.color.alternative
: two alternative colors for edges within the community (grey70 by default) and edges between communities (grey95 by default)

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 igraph object, appended by node attributes including "community" for communit memberships, "contribution" for contribution to its community, "xcoord" for x-coordinates, "ycoord" for y-coordiates, and by edge attributes including "color" for between-community edges ('grey95') and within-community edges ('grey70').

Note

The input graph will has an equal weight if there is no 'weight' edge attribute associated with

See Also

xGGnetwork

Examples

```
# 1) generate a random bipartite graph
set.seed(123)
g <- sample_bipartite(100, 50, p=0.1)
V(g)$name <- V(g)

## Not run:
# 2) obtain its community
ig <- xBigraph(g)
gp <- xGGnetwork(ig, node.label='name', node.label.size=2,
node.label.color='black', node.label.alpha=0.8, node.label.padding=0,
node.label.arrow=0, node.label.force=0.002, node.shape='type',
node.shape.title='Type', node.xcoord='xcoord', node.ycoord='ycoord',
node.color='community', node.color.title='Community',
colormap='jet.both', ncolors=64, zlim=NULL, node.size='contribution',
node.size.range=c(1,4), node.size.title='Contribution', slim=NULL,
edge.color="color",edge.color.alpha=0.5,edge.curve=0,edge.arrow.gap=0)

## make it discrete for the colorbar
gp +
scale_colour_gradientn(colors=xColormap('jet.both')(64),breaks=seq(1,max(V(ig)$community)))
+ guides(color=guide_legend(title="Community"))

## End(Not run)
```

xBiheatmap *Function to visualise bipartitle graph communities using heatmap*

Description

xBiheatmap is supposed to visualise bipartitle graph communities using heatmap.

Usage

```
xBiheatmap(g, which.communites = NULL, colormap = "spectral",
ncolors = 64, zlim = NULL, barwidth = 0.3, barheight = NULL,
nbin = 64, legend.title = "", x.rotate = 60, x.text.size = 3,
y.text.size = 3, legend.text.size = 4, legend.title.size = 6,
shape = 19, size = 0.5, plot.margin = unit(c(5.5, 5.5, 5.5, 5.5),
"pt"),
font.family = "sans", na.color = "transparent",
intercept.color = "grey95", intercept.size = 0.3)
```

Arguments

g	an object of class "igraph" for a bipartitel graph with node attributes 'type', 'community' and 'contribution'
which.communites	a vector specifying which communites are visualised. If NULL (by default), all communites will be used
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
ncolors	the number of colors specified over the colormap
zlim	the minimum and maximum z values for which colors should be plotted, defaulting to the range of the finite values of displayed matrix
barwidth	the width of the colorbar. Default value is 'legend.key.width' or 'legend.key.size' in 'theme' or theme
barheight	the height of the colorbar. Default value is 'legend.key.height' or 'legend.key.size' in 'theme' or theme
nbin	the number of bins for drawing colorbar
legend.title	the title of the colorbar. By default, it is ''
x.rotate	the angle to rotate the x tick labelings. By default, it is 60
x.text.size	the text size of the x tick labelings. By default, it is 6

y.text.size	the text size of the y tick labelings. By default, it is 6
legend.text.size	the text size of the legend tick labelings. By default, it is 5
legend.title.size	the text size of the legend titles. By default, it is 6
shape	the number specifying the shape. By default, it is 19
size	the number specifying the shape size. By default, it is 2
plot.margin	the margin (t, r, b, l) around plot. By default, it is unit(c(5.5,5.5,5.5,5.5),"pt")
font.family	the font family for texts
na.color	the color for NAs. By default, it is 'grey80'
intercept.color	intercept color
intercept.size	intercept size

Value

a ggplot2 object

See Also

xBigraph, xHeatmap

Examples

```
# 1) generate a random bipartite graph
set.seed(123)
g <- sample_bipartite(100, 50, p=0.1)
V(g)$name <- paste0('node_',1:vcount(g))

## Not run:
# 2) obtain its community
ig <- xBigraph(g)

# 3) heatmap of its community
gp <- xBiheatmap(ig)

## End(Not run)
```

---

xBiproject *Function to obtain a projected graph from a bipartitle graph*

Description

xBiproject is supposed to obtain a projected graph from a bipartitle graph.

Usage

```
xBiproject(g, verbose = TRUE)
```

Arguments

g	an object of class "igraph" (or "graphNEL") for a bipartitel graph with a 'type' node attribute
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 igraph object.

Note

The input graph will has an equal weight if there is no 'weight' edge attribute associated with

See Also

xBiproject

Examples

```
# 1) generate a random bipartite graph
set.seed(123)
g <- sample_bipartite(100, 50, p=0.05)
V(g)$name <- V(g)

## Not run:
# 2) obtain its projected graph
ig <- xBiproject(g)

# 3) estimate pairwise affiinity between nodes
mat <- xRWkernel(ig)

## End(Not run)
```

---

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

Description

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

Usage

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

Arguments

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

Value

TRUE for using parallel computing; FALSE otherwise

Note

Whether parallel computation with multicores is used is system-specific. Also, it will depend on whether these two packages "foreach" and "doParallel" have been installed. It can be installed via: `source("http://bioconductor.org/biocLite.R"); biocLite(c("foreach","doParallel"))`.

See Also

xDAGsim, xSocialiser, xSocialiserGenes, xSocialiserSNPs, xGRviaGenomicAnnoAdv

Examples

```
## Not run:
xCheckParallel()

## End(Not run)
```

---

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

Description

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

Usage

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

Arguments

g	an object of class "igraph". For example, it stores semantic similarity results with nodes for genes/SNPs and edges for pair-wise semantic similarity between them
entity	the entity of similarity analysis for which results are being plotted. It can be either "SNP" or "Gene"
top_num	the top number of similarity edges to be plotted
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "lightyellow-orange" (by default), "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
rescale	logical to indicate whether the edge values are rescaled to the range [0,1]. By default, it sets to true
nodes.query	nodes in query for which edges attached to them will be displayed. By default, it sets to NULL meaning no such restriction
ideogram	logical to indicate whether chromosome banding is plotted
chr.exclude	a character vector of chromosomes to exclude from the plot, e.g. c("chrX", "chrY"). By defautl, it is 'auto' meaning those chromosomes without data will be excluded. If NULL, no chromosome is excluded
entity.label.cex	the font size of genes/SNPs labels. Default is 0.8
entity.label.side	the position of genes/SNPs labels relative to chromosome ideogram. It can be "out" (by default) or "in"
entity.label.track	an integer specifying the plot track for genes/SNPs labels. Default is 1
entity.label.query	which genes/SNPs in query will be labelled. By default, it sets to NULL meaning all will be displayed. If labels in query can not be found, then all will be displayed
GR.SNP	the genomic regions of SNPs. By default, it is 'dbSNP_GWAS', that is, SNPs from dbSNP (version 146) restricted to GWAS SNPs and their LD SNPs (hg19). It can be 'dbSNP_Common', that is, Common SNPs from dbSNP (version 146) plus GWAS SNPs and their LD SNPs (hg19). Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to dbSNP IDs. Then, tell "GR.SNP" with your RData file name (with or without extension), plus specify your file RData path in "RData.location". Note: you can also load your customised GR object directly
GR.Gene	the genomic regions of genes. By default, it is 'UCSC_knownGene', that is, UCSC known genes (together with genomic locations) based on human genome

	assembly hg19. It can be 'UCSC_knownCanonical', that is, UCSC known canonical genes (together with genomic locations) based on human genome assembly hg19. Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to Gene Symbols. Then, tell "GR.Gene" with your RData file name (with or without extension), plus specify your file RData path in "RData.location". Note: you can also load your customised GR object directly
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details

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 XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata/"

library(RCircos)

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

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

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

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

## End(Not run)
```

`xColormap` *Function to define a colormap*

Description

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

Usage

```
xColormap(colormap = c("bwr", "jet", "gbr", "wyr", "br", "yr",
"rainbow",
"wb", "heat", "terrain", "topo", "cm", "ggplot2", "jet.top",
"jet.bottom",
"jet.both", "spectral", "ggplot2.top", "ggplot2.bottom",
"ggplot2.both",
"RdYlBu"), interpolate = c("spline", "linear"))
```

Arguments

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

Value

- palette.name: a function that takes an integer argument for generating that number of colors interpolating the given sequence

Note

The input colormap includes:

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

Examples

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

# 2) define default colormap from ggplot2
palette.name <- xColormap(colormap="ggplot2")
```

```
# use the return function "palette.name" to generate 3 default colors used by ggplot2
palette.name(3)

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

---

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

Description

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

Usage

```
xCombineNet(list_ig, combineBy = c("union", "intersect"),
attrBy = c("intersect", "union"), keep.all.vertices = FALSE,
verbose = TRUE)
```

Arguments

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

Value

an object of class "igraph"

Note

none

See Also

xDefineNet

Examples

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

## End(Not run)

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

## End(Not run)
```

`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)
```

---

xCorrelation *Function to calculate and visualise correlation*

Description

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

Usage

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

Arguments

df	a data frame with two columns ('name' and 'value')
list_vec	a named vector containing numeric values. Alternatively it can be a list of named vectors

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

Value

a list with three componets:

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

Note

none

See Also

xCorrelation

Examples

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

## End(Not run)

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

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

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

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

## End(Not run)
```

---

xCrosstalk *Function to identify a pathway crosstalk*

Description

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

Usage

```
xCrosstalk(data, entity = c("Gene", "GR"), significance.threshold =
NULL,
score.cap = NULL, build.conversion = c(NA, "hg38.to.hg19",
"hg18.to.hg19"), crosslink = c("genehancer", "PCHiC_combined",
"GTEx_V6p_combined", "nearby"), crosslink.customised = NULL,
cdf.function = c("original", "empirical"), scoring.scheme = c("max",
"sum", "sequential"), nearby.distance.max = 50000,
nearby.decay.kernel = c("rapid", "slow", "linear", "constant"),
nearby.decay.exponent = 2, networks = c("KEGG", "KEGG_metabolism",
"KEGG_genetic", "KEGG_environmental", "KEGG_cellular",
"KEGG_organismal",
"KEGG_disease", "REACTOME", "PCommonsDN_Reactome"), seed.genes = T,
subnet.significance = 0.01, subnet.size = NULL,
ontologies = c("KEGGenvironmental", "KEGG", "KEGGmetabolism",
"KEGGgenetic",
"KEGGcellular", "KEGGorganismal", "KEGGdisease"), size.range = c(10,
2000),
min.overlap = 10, fdr.cutoff = 0.05, crosstalk.top = NULL,
glayout = layout_with_kk, verbose = T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

data	a named input vector containing the significance level for genes (gene symbols) or genomic regions (GR). For this named vector, the element names are gene symbols or GR (in the format of 'chrN:start-end', where N is either 1-22 or X, start/end is genomic positional number; for example, 'chr1:13-20'), the element values for the significance level (measured as p-value or fdr). Alternatively, it can be a matrix or data frame with two columns: 1st column for gene symbols or GR, 2nd column for the significance level. Also supported is the input with GR only (without the significance level)
entity	the entity. It can be either "Gene" or "GR"
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 into scores. If given, those below this are considered significant and thus scored positively. Instead, those above this are considered insignificant and thus receive no score
score.cap	the maximum score being capped. By default, it is set to NULL, meaning that no capping is applied
build.conversion	the conversion from one genome build to another. The conversions supported are "hg38.to.hg19" and "hg18.to.hg19". By default it is NA (no need to do so)
crosslink	the built-in crosslink info with a score quantifying the link of a GR to a gene. See xGR2xGenes for details
crosslink.customised	the crosslink info with a score quantifying the link of a GR to a gene. A user-input matrix or data frame with 4 columns: 1st column for genomic regions (formatted as "chr:start-end", genome build 19), 2nd column for Genes, 3rd for crosslink score (crosslinking a genomic region to a gene, such as -log10 significance level), and 4th for contexts (optional; if nor provided, it will be added as 'C'). Alternatively, it can be a file containing these 4 columns. Required, otherwise it will return NULL
cdf.function	a character specifying how to transform the input crosslink score. It can be one of 'original' (no such transformation), and 'empirical' for looking at empirical Cumulative Distribution Function (cdf; as such it is converted into pvalue-like values [0,1])
scoring.scheme	the method used to calculate seed gene scores under a set of GR (also over Contexts if many). It can be one of "sum" for adding up, "max" for the maximum, and "sequential" for the sequential weighting. The sequential weighting is done via: $\sum_{i=1} \frac{R_i}{i}$, where R_i is the i^{th} rank (in a descreasing order)
nearby.distance.max	the maximum distance between genes and GR. Only those genes no far way from this distance will be considered as seed genes. This parameter will influence the distance-component weights calculated for nearby GR per gene
nearby.decay.kernel	a character specifying a decay kernel function. It can be one of 'slow' for slow decay, 'linear' for linear decay, and 'rapid' for rapid decay. If no distance weight is used, please select 'constant'

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

`networks`
: the built-in network. For direct (pathway-merged) interactions sourced from KEGG, it can be 'KEGG' for all, 'KEGG_metabolism' for pathways grouped into 'Metabolism', 'KEGG_genetic' for 'Genetic Information Processing' pathways, 'KEGG_environmental' for 'Environmental Information Processing' pathways, 'KEGG_cellular' for 'Cellular Processes' pathways, 'KEGG_organismal' for 'Organismal Systems' pathways, and 'KEGG_disease' for 'Human Diseases' pathways. 'REACTOME' for protein-protein interactions derived from Reactome pathways. Pathways Commons pathway-merged network from individual sources, that is, "PCommonsDN_Reactome" for those from Reactome

`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

`ontologies`
: the ontologies supported currently. It can be 'AA' for AA-curated pathways, KEGG pathways (including 'KEGG' for all, 'KEGGmetabolism' for 'Metabolism' pathways, 'KEGGgenetic' for 'Genetic Information Processing' pathways, 'KEGGenvironmental' for 'Environmental Information Processing' pathways, 'KEGGcellular' for 'Cellular Processes' pathways, 'KEGGorganismal' for 'Organismal Systems' pathways, and 'KEGGdisease' for 'Human Diseases' pathways), 'REACTOME' for REACTOME pathways or 'REACTOME_x' for its sub-ontologies (where x can be 'CellCellCommunication', 'CellCycle', 'CellularResponsesToExternalStimuli', 'ChromatinOrganization', 'CircadianClock', 'DevelopmentalBiology', 'DigestionAndAbsorption', 'Disease', 'DNARepair', 'DNAReplication', 'ExtracellularMatrixOrganization', 'GeneExpression(Transcription)', 'Hemostasis', 'ImmuneSystem', 'Metabolism', 'MetabolismOfProteins', 'MetabolismOfRNA', 'Mitophagy', 'MuscleContraction', 'NeuronalSystem', 'OrganelleBiogenesisAndMaintenance', 'ProgrammedCellDeath', 'Reproduction', 'SignalTransduction', 'TransportOfSmallMolecules', 'VesicleMediatedTransport')

`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

`fdr.cutoff`
: fdr cutoff used to declare the significant terms. By default, it is set to 0.05

`crosstalk.top`
: the number of the top paths will be returned. By default, it is NULL meaning no such restrictions

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

- ig_paths: an object of class "igraph". It has graph attribute (enrichment, and/or evidence, gp_evidence and membership if entity is 'GR'), ndoe attributes (crosstalk)
- gp_paths: a 'ggplot' object for pathway crosstalk visualisation
- gp_heatmap: a 'ggplot' object for pathway member gene visualisation
- ig_subg: an object of class "igraph".

See Also

xDefineNet, xCombineNet, xSubneterGenes, xGR2xNet, xEnricherGenesAdv, xGGnetwork, xHeatmap

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata/"

# 1) at the gene level
data(Haploid_regulators)
## only PD-L1 regulators and their significance info (FDR)
data <- subset(Haploid_regulators, Phenotype=='PDL1')[,c('Gene','FDR')]
## pathway crosstalk
cPath <- xCrosstalk(data, entity="Gene", network="KEGG",
subnet.significance=0.05, subnet.size=NULL,
ontologies="KEGGenvironmental", RData.location=RData.location)
cPath
```

```
## visualisation
pdf("xCrosstalk_Gene.pdf", width=7, height=8)
gp_both <-
gridExtra::grid.arrange(grobs=list(cPath$gp_paths,cPath$gp_heatmap),
layout_matrix=cbind(c(1,1,1,1,2)))
dev.off()

# 2) at the genomic region (SNP) level
data(ImmunoBase)
## all ImmunoBase GWAS SNPs and their significance info (p-values)
ls_df <- lapply(ImmunoBase, function(x) as.data.frame(x$variant))
df <- do.call(rbind, ls_df)
data <- unique(cbind(GR=paste0(df$seqnames,':',df$start,'-',df$end),
Sig=df$Pvalue))
## pathway crosstalk
df_xGenes <- xGR2xGenes(data[as.numeric(data[,2])<5e-8,1],
format="chr:start-end", crosslink="PCHiC_combined", scoring=T,
RData.location=RData.location)
mSeed <- xGR2xGeneScores(data, significance.threshold=5e-8,
crosslink="PCHiC_combined", RData.location=RData.location)
subg <- xGR2xNet(data, significance.threshold=5e-8,
crosslink="PCHiC_combined", network="KEGG", subnet.significance=0.1,
RData.location=RData.location)
cPath <- xCrosstalk(data, entity="GR", significance.threshold=5e-8,
crosslink="PCHiC_combined", networks="KEGG", subnet.significance=0.1,
ontologies="KEGGenvironmental", RData.location=RData.location)
cPath
## visualisation
pdf("xCrosstalk_SNP.pdf", width=7, height=8)
gp_both <-
gridExtra::grid.arrange(grobs=list(cPath$gp_paths,cPath$gp_heatmap),
layout_matrix=cbind(c(1,1,1,1,2)))
dev.off()

# 3) at the genomic region (without the significance info) level
Age_CpG <- xRDataLoader(RData.customised='Age_CpG',
RData.location=RData.location)[-1,1]
CgProbes <- xRDataLoader(RData.customised='CgProbes',
RData.location=RData.location)
ind <- match(Age_CpG, names(CgProbes))
gr_CpG <- CgProbes[ind[!is.na(ind)]]
data <- xGRcse(gr_CpG, format='GRanges')
## pathway crosstalk
df_xGenes <- xGR2xGenes(data, format="chr:start-end",
crosslink="PCHiC_combined", scoring=T, RData.location=RData.location)
subg <- xGR2xNet(data, crosslink="PCHiC_combined", network="KEGG",
subnet.significance=0.1, RData.location=RData.location)
cPath <- xCrosstalk(data, entity="GR", crosslink="PCHiC_combined",
networks="KEGG", subnet.significance=0.1,
ontologies="KEGGenvironmental", RData.location=RData.location)
cPath

## 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)
```

`xDAGpropagate` *Function to generate a subgraph of a direct acyclic graph (DAG) propagaged by the input annotation data*

Description

`xDAGpropagate` 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 annotations are propagated to the ontology root (eg, retaining the minmum pvalue). The propagaged 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

```
xDAGpropagate(g, annotation, path.mode = c("all_paths",
"shortest_paths",
"all_shortest_paths"), propagation = c("all", "min", "max"),
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 data frame with three columns: 1st column for variants/genes, 2nd column

	for terms, and 3rd column for values, or a list (with the name for terms) each element storing a named vactor (that is, value for the content and variants/genes as names)
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)
propagation	how to propagate the score. It can be "max" for retaining the maximum value from its children, "min" for retaining the minimum value from its children, and 'all' for retaining all from its children (by default)
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/propagated 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 (with numeric values as elements); 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:
# Load the library
library(XGR)

## End(Not run)

RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# 1) EF ontology
# ig.EF (an object of class "igraph" storing as a directed graph)
ig <- xRDataLoader('ig.EF', RData.location=RData.location)
## optional: extract the disease part (EFO:0000408)
neighs.out <- igraph::neighborhood(ig, order=vcount(ig),
nodes='EFO:0000408', mode="out")
vids <- V(ig)[unique(unlist(neighs.out))]$name
g <- igraph::induced.subgraph(ig, vids=vids)
```

```
################
# 2a) load GWAS SNPs annotated by EF (an object of class "dgCMatrix" storing a sparse matrix)
annotation <- xRDataLoader(RData='GWAS2EF',
RData.location=RData.location)
## only significant
annotation[as.matrix(annotation>5e-8)] <- 0

# 2b) propagation based on shortest paths (ie the most concise subgraph)
dag <- xDAGpropagate(g, annotation, path.mode="shortest_paths",
propagation="min")

# 2c) color-code nodes/terms according to the number of annotations
data <- sapply(V(dag)$anno, length)
names(data) <- V(dag)$name
## only those GWAS>=100
nodes <- V(dag)$name[data>=100]
dagg <- igraph::induced.subgraph(dag, vids=nodes)
### DAG plot
dnet::visDAG(dagg, data, node.info="both")
### Net plot
set.seed(825); glayout <- layout_with_kk(dagg)
xVisNet(dagg, pattern=data, colormap="yr", glayout=glayout,
vertex.label=V(dagg)$term_name, vertex.shape="sphere",
vertex.label.font=2, vertex.label.dist=0.2, vertex.label.cex=0.5,
zlim=c(100,300))
### interpolation plot
set.seed(825); glayout <- layout_with_kk(dagg)
pattern <- sapply(V(dagg)$anno, length)
ls_xyz <- data.frame(x=glayout[,1], y=glayout[,2], z=log10(pattern))
xVisInterp(ls_xyz, nD="auto", image=TRUE)

################
3a) load ChEMBL targets annotated by EF (an object of class "dgCMatrix"
storing a sparse matrix)
annotation <- xRDataLoader(RData='Target2EF',
RData.location=RData.location)
## only approved (phase=4)
annotation[as.matrix(annotation<4)] <- 0

3b) propagation based on all paths
dag <- xDAGpropagate(g, annotation, path.mode="all_paths",
propagation="max")

3c) color-code nodes/terms according to the number of annotations
data <- sapply(V(dag)$anno, length)
names(data) <- V(dag)$name
## only those Targets>=50
nodes <- V(dag)$name[data>=50]
dagg <- igraph::induced.subgraph(dag, vids=nodes)
### DAG plot
dnet::visDAG(dagg, data, node.info="both")
### Net plot
set.seed(825); glayout <- layout_with_kk(dagg)
```

```
xVisNet(dagg, pattern=data, colormap="yr", glayout=glayout,
vertex.label=V(dagg)$term_name, vertex.shape="sphere",
vertex.label.font=2, vertex.label.dist=0.2, vertex.label.cex=0.5,
zlim=c(50,300))
### interpolation plot
set.seed(825); glayout <- layout_with_kk(dagg)
pattern <- sapply(V(dagg)$anno, length)
ls_xyz <- data.frame(x=glayout[,1], y=glayout[,2], z=log10(pattern))
xVisInterp(ls_xyz, nD="3D", contour=TRUE)

## 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 https://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. It will depend on whether these two packages "foreach" and "doParallel" have been installed. It can be installed via: `source("http://bioconductor.org/biocLite.R"); biocLite(c("foreach","doParallel"))` 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)
```

---

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

Description

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

Usage

```
xDefineEQTL(data = NULL, include.eQTL = c(NA, "JKscience_CD14",
"JKscience_LPS2", "JKscience_LPS24", "JKscience_IFN", "JKscience_TS2A",
"JKscience_TS2A_CD14", "JKscience_TS2A_LPS2", "JKscience_TS2A_LPS24",
"JKscience_TS2A_IFN", "JKscience_TS2B", "JKscience_TS2B_CD14",
"JKscience_TS2B_LPS2", "JKscience_TS2B_LPS24", "JKscience_TS2B_IFN",
"JKscience_TS3A", "JKng_bcell", "JKng_bcell_cis", "JKng_bcell_trans",
"JKng_mono", "JKng_mono_cis", "JKng_mono_trans", "JKpg_CD4",
"JKpg_CD4_cis",
"JKpg_CD4_trans", "JKpg_CD8", "JKpg_CD8_cis", "JKpg_CD8_trans",
"JKnc_neutro", "JKnc_neutro_cis", "JKnc_neutro_trans",
"WESTRAng_blood",
"WESTRAng_blood_cis", "WESTRAng_blood_trans", "JK_nk", "JK_nk_cis",
"JK_nk_trans", "GTEx_V4_Adipose_Subcutaneous", "GTEx_V4_Artery_Aorta",
"GTEx_V4_Artery_Tibial", "GTEx_V4_Esophagus_Mucosa",
"GTEx_V4_Esophagus_Muscularis", "GTEx_V4_Heart_Left_Ventricle",
"GTEx_V4_Lung", "GTEx_V4_Muscle_Skeletal", "GTEx_V4_Nerve_Tibial",
"GTEx_V4_Skin_Sun_Exposed_Lower_leg", "GTEx_V4_Stomach",
"GTEx_V4_Thyroid",
```

```
"GTEx_V4_Whole_Blood", "GTEx_V6p_Adipose_Subcutaneous",
"GTEx_V6p_Adipose_Visceral_Omentum", "GTEx_V6p_Adrenal_Gland",
"GTEx_V6p_Artery_Aorta", "GTEx_V6p_Artery_Coronary",
"GTEx_V6p_Artery_Tibial",
"GTEx_V6p_Brain_Anterior_cingulate_cortex_BA24",
"GTEx_V6p_Brain_Caudate_basal_ganglia",
"GTEx_V6p_Brain_Cerebellar_Hemisphere", "GTEx_V6p_Brain_Cerebellum",
"GTEx_V6p_Brain_Cortex", "GTEx_V6p_Brain_Frontal_Cortex_BA9",
"GTEx_V6p_Brain_Hippocampus", "GTEx_V6p_Brain_Hypothalamus",
"GTEx_V6p_Brain_Nucleus_accumbens_basal_ganglia",
"GTEx_V6p_Brain_Putamen_basal_ganglia",
"GTEx_V6p_Breast_Mammary_Tissue",
"GTEx_V6p_Cells_EBVtransformed_lymphocytes",
"GTEx_V6p_Cells_Transformed_fibroblasts", "GTEx_V6p_Colon_Sigmoid",
"GTEx_V6p_Colon_Transverse",
"GTEx_V6p_Esophagus_Gastroesophageal_Junction",
"GTEx_V6p_Esophagus_Mucosa", "GTEx_V6p_Esophagus_Muscularis",
"GTEx_V6p_Heart_Atrial_Appendage", "GTEx_V6p_Heart_Left_Ventricle",
"GTEx_V6p_Liver", "GTEx_V6p_Lung", "GTEx_V6p_Muscle_Skeletal",
"GTEx_V6p_Nerve_Tibial", "GTEx_V6p_Ovary", "GTEx_V6p_Pancreas",
"GTEx_V6p_Pituitary", "GTEx_V6p_Prostate",
"GTEx_V6p_Skin_Not_Sun_Exposed_Suprapubic",
"GTEx_V6p_Skin_Sun_Exposed_Lower_leg",
"GTEx_V6p_Small_Intestine_Terminal_Ileum", "GTEx_V6p_Spleen",
"GTEx_V6p_Stomach", "GTEx_V6p_Testis", "GTEx_V6p_Thyroid",
"GTEx_V6p_Uterus",
"GTEx_V6p_Vagina", "GTEx_V6p_Whole_Blood"), eQTL.customised = NULL,
verbose = TRUE, RData.location =
"http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

data	NULL or an input vector containing SNPs. If NULL, all SNPs will be considered. If a input vector containing SNPs, SNPs should be provided as dbSNP ID (ie starting with rs). Alternatively, they can be in the format of 'chrN:xxx', where N is either 1-22 or X, xxx is number; for example, 'chr16:28525386'
include.eQTL	genes modulated by eQTL (also Lead SNPs or in LD with Lead SNPs) are also included. By default, it is 'NA' to disable this option. Otherwise, those genes modulated by eQTL will be included. Pre-built eQTL datasets are detailed in the section 'Note'
eQTL.customised	a user-input matrix or data frame with 4 columns: 1st column for SNPs/eQTLs, 2nd column for Genes, 3rd for eQTL mapping significance level (p-values or FDR), and 4th for contexts (required even though only one context is input). Alternatively, it can be a file containing these 4 columns. It is designed to allow the user analysing their eQTL data. This customisation (if provided) will populate built-in eQTL data; mysql -e "use pi; SELECT rs_id_dbSNP147_GRCh37p13,gene_name,pval_nominal,' FROM GTEx_V7_pair WHERE rs_id_dbSNP147_GRCh37p13!='.';" > /var/www/bigdata/eQTL.custom

verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details

Value

a data frame with following columns:

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

Note

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

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

- JKscience_IFN: cis and trans-eQTLs in the activating state induced by 24-hour interferon-gamma (based on 228 individuals).

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

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

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

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

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

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

5. eQTLs in NK cells. Unpublished

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

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

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

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

- JKpg_CD4: cis- and trans-eQTLs.
- JKpg_CD4_cis: cis-eQTLs only.
- JKpg_CD4_trans: trans-eQTLs only.

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

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

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

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

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

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

- GTEx_V6p_Colon_Sigmoid: cis-eQTLs in tissue "Colon Sigmoid".
- GTEx_V6p_Colon_Transverse: cis-eQTLs in tissue "Colon Transverse".
- GTEx_V6p_Esophagus_Gastroesophageal_Junction: cis-eQTLs in tissue "Esophagus Gastroesophageal Junction".
- GTEx_V6p_Esophagus_Mucosa: cis-eQTLs in tissue "Esophagus Mucosa".
- GTEx_V6p_Esophagus_Muscularis: cis-eQTLs in tissue "Esophagus Muscularis".
- GTEx_V6p_Heart_Atrial_Appendage: cis-eQTLs in tissue "Heart Atrial Appendage".
- GTEx_V6p_Heart_Left_Ventricle: cis-eQTLs in tissue "Heart Left Ventricle".
- GTEx_V6p_Liver: cis-eQTLs in tissue "Liver".
- GTEx_V6p_Lung: cis-eQTLs in tissue "Lung".
- GTEx_V6p_Muscle_Skeletal: cis-eQTLs in tissue "Muscle Skeletal".
- GTEx_V6p_Nerve_Tibial: cis-eQTLs in tissue "Nerve Tibial".
- GTEx_V6p_Ovary: cis-eQTLs in tissue "Ovary".
- GTEx_V6p_Pancreas: cis-eQTLs in tissue "Pancreas".
- GTEx_V6p_Pituitary: cis-eQTLs in tissue "Pituitary".
- GTEx_V6p_Prostate: cis-eQTLs in tissue "Prostate".
- GTEx_V6p_Skin_Not_Sun_Exposed_Suprapubic: cis-eQTLs in tissue "Skin Not Sun Exposed (Suprapubic)".
- GTEx_V6p_Skin_Sun_Exposed_Lower_leg: cis-eQTLs in tissue "Skin Sun Exposed (Lower leg)".
- GTEx_V6p_Small_Intestine_Terminal_Ileum: cis-eQTLs in tissue "Small Intestine Terminal Ileum".
- GTEx_V6p_Spleen: cis-eQTLs in tissue "Spleen".
- GTEx_V6p_Stomach: cis-eQTLs in tissue "Stomach".
- GTEx_V6p_Testis: cis-eQTLs in tissue "Testis".
- GTEx_V6p_Thyroid: cis-eQTLs in tissue "Thyroid".
- GTEx_V6p_Uterus: cis-eQTLs in tissue "Uterus".
- GTEx_V6p_Vagina: cis-eQTLs in tissue "Vagina".
- GTEx_V6p_Whole_Blood: cis-eQTLs in tissue "Whole Blood".

See Also

xRDataLoader

Examples

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

## End(Not run)
```

```
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# a) provide the SNPs with the significance info
data(ImmunoBase)
gr <- ImmunoBase$AS$variants
AS <- as.data.frame(GenomicRanges::mcols(gr)[, c('Variant','Pvalue')])

# b) define eQTL genes
df_SGS <- xDefineEQTL(data=AS[,1], include.eQTL="JKscience_TS2A",
RData.location=RData.location)

## End(Not run)
```

---

xDefineGenomicAnno *Function to define genomic annotations*

Description

xDefineGenomicAnno is supposed to define genomic annotations. It returns an object of class "GenomicRangesList" (GRL).

Usage

```
xDefineGenomicAnno(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_Encode_TFBS", "ReMap_PublicAndEncode_TFBS",
"ReMap_Public_mergedTFBS",
"ReMap_Encode_mergedTFBS", "ReMap_PublicAndEncode_mergedTFBS",
"Blueprint_BoneMarrow_Histone", "Blueprint_CellLine_Histone",
"Blueprint_CordBlood_Histone", "Blueprint_Thymus_Histone",
"Blueprint_VenousBlood_Histone", "Blueprint_DNaseI",
"Blueprint_Methylation_hyper", "Blueprint_Methylation_hypo",
"EpigenomeAtlas_15Segments_E029", "EpigenomeAtlas_15Segments_E030",
"EpigenomeAtlas_15Segments_E031", "EpigenomeAtlas_15Segments_E032",
"EpigenomeAtlas_15Segments_E033", "EpigenomeAtlas_15Segments_E034",
"EpigenomeAtlas_15Segments_E035", "EpigenomeAtlas_15Segments_E036",
"EpigenomeAtlas_15Segments_E037", "EpigenomeAtlas_15Segments_E038",
"EpigenomeAtlas_15Segments_E039", "EpigenomeAtlas_15Segments_E040",
"EpigenomeAtlas_15Segments_E041", "EpigenomeAtlas_15Segments_E042",
```

```
"EpigenomeAtlas_15Segments_E043", "EpigenomeAtlas_15Segments_E044",
"EpigenomeAtlas_15Segments_E045", "EpigenomeAtlas_15Segments_E046",
"EpigenomeAtlas_15Segments_E047", "EpigenomeAtlas_15Segments_E048",
"EpigenomeAtlas_15Segments_E050", "EpigenomeAtlas_15Segments_E051",
"EpigenomeAtlas_15Segments_E062", "CpG_anno", "Genic_anno",
"FANTOM5_CAT_Cell", "FANTOM5_CAT_Tissue", "FANTOM5_CAT_DO",
"FANTOM5_CAT_EFO",
"FANTOM5_CAT_HPO", "FANTOM5_CAT_MESH", "FANTOM5_CAT_PICS"), verbose =
T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

GR.annotation	the genomic regions of annotation data. By default, it is 'NA' to disable this option. Pre-built genomic annotation data are detailed in the section 'Note'. Alternatively, the user can also directly provide a customised GR object (or a list of GR objects or a GRL object)
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 GRL object

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. This dataset was generated from ENCODE/Broad Institute.
   - `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. This dataset was generated from ENCODE/Stanford/Yale/Davis/Harvard.
   - `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. This dataset was generated from ENCODE/University of Washington.

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 (1759 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_Encode_TFBS: a list (1066 combinations of ENCODE transcription factors and cell types) of GenomicRanges objects; each is an GR object containing identified peaks per ENCODE study per transcripton factor per cell type.
   - ReMap_PublicAndEncode_TFBS: a list (2825 combinations of GSE/ENCODE studies and transcription factors and cell types) of GenomicRanges objects; each is an GR object containing identified peaks per GSE/ENCODE study per transcripton factor per cell type.
   - ReMap_Public_mergedTFBS: a list (331 transcription factors under GSE studies) of GenomicRanges objects; each is an GR object containing merged peaks per transcripton factor.
   - ReMap_Encode_mergedTFBS: a list (279 transcription factors under ENCODE) of GenomicRanges objects; each is an GR object containing merged peaks per transcripton factor.
   - ReMap_PublicAndEncode_mergedTFBS: a list (485 transcription factors under GSE studies and ENCODE) of GenomicRanges objects; each is an GR object containing identified peaks per transcripton factor.

10. Blueprint Histone Modification ChIP-seq data

    - 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 (from bone marrow).
    - 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.
    - 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 (from cord blood).
    - 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 (from thymus).

- `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 (from venous blood).

11. BLUEPRINT DNaseI Hypersensitivity site data

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

12. BLUEPRINT DNA Methylation data

- `Blueprint_Methylation_hyper`: a list (206 samples) of GenomicRanges objects; each is an GR object containing hyper-methylated CpG regions per sample.
- `Blueprint_Methylation_hypo`: a list (206 samples) of GenomicRanges objects; each is an GR object containing hypo-methylated CpG regions per sample.

13. Roadmap Epigenomics Core 15-state Genome Segmentation data for primary cells (blood and T cells)

- `EpigenomeAtlas_15Segments_E033`: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E033 (Primary T cells from cord blood).
- `EpigenomeAtlas_15Segments_E034`: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E034 (Primary T cells from peripheral blood).
- `EpigenomeAtlas_15Segments_E037`: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E037 (Primary T helper memory cells from peripheral blood 2).
- `EpigenomeAtlas_15Segments_E038`: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E038 (Primary T helper naive cells from peripheral blood).
- `EpigenomeAtlas_15Segments_E039`: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E039 (Primary T helper naive cells from peripheral blood).
- `EpigenomeAtlas_15Segments_E040`: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E040 (Primary T helper memory cells from peripheral blood 1).
- `EpigenomeAtlas_15Segments_E041`: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E041 (Primary T helper cells PMA-I stimulated).
- `EpigenomeAtlas_15Segments_E042`: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E042 (Primary T helper 17 cells PMA-I stimulated).
- `EpigenomeAtlas_15Segments_E043`: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E043 (Primary T helper cells from peripheral blood).

- EpigenomeAtlas_15Segments_E044: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E044 (Primary T regulatory cells from peripheral blood).
- EpigenomeAtlas_15Segments_E045: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E045 (Primary T cells effector/memory enriched from peripheral blood).
- EpigenomeAtlas_15Segments_E047: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E047 (Primary T killer naive cells from peripheral blood).
- EpigenomeAtlas_15Segments_E048: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E048 (Primary T killer memory cells from peripheral blood).
- EpigenomeAtlas_15Segments_E062: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E062 (Primary mononuclear cells from peripheral blood).

14. Roadmap Epigenomics Core 15-state Genome Segmentation data for primary cells (HSC and B cells)

- EpigenomeAtlas_15Segments_E029: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E029 (Primary monocytes from peripheral blood).
- EpigenomeAtlas_15Segments_E030: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E030 (Primary neutrophils from peripheral blood).
- EpigenomeAtlas_15Segments_E031: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E031 (Primary B cells from cord blood).
- EpigenomeAtlas_15Segments_E032: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E032 (Primary B cells from peripheral blood).
- EpigenomeAtlas_15Segments_E035: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E035 (Primary hematopoietic stem cells).
- EpigenomeAtlas_15Segments_E036: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E036 (Primary hematopoietic stem cells short term culture).
- EpigenomeAtlas_15Segments_E046: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E046 (Primary Natural Killer cells from peripheral blood).
- EpigenomeAtlas_15Segments_E050: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E050 (Primary hematopoietic stem cells G-CSF-mobilized Female).
- EpigenomeAtlas_15Segments_E051: a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome E051 (Primary hematopoietic stem cells G-CSF-mobilized Male).

15. CpG annotation

    - CpG_anno: a list (4 categories) of GenomicRanges objects; each is an GR object. They are exclusive, including (in order) "CpG_islands", "CpG_shores" (2Kb upstream/downstream from the ends of the CpG islands), "CpG_shelves" (2Kb upstream/downstream of the farthest upstream/downstream limits of the CpG shores), and "CpG_inter" (the remaining inter-CGI genomic regions 'open sea').

16. Genic annotation

    - Genic_anno: a list (12 categories) of GenomicRanges objects; each is an GR object. They are not exclusively, including "Genic_1to5kb" (1-5Kb upstream of TSS), "Genic_promoters" (1Kb upstream of TSS), "Genic_5UTRs", "Genic_firstexons" (first exons), "Genic_exons", "Genic_exonintronboundaries", "Genic_introns", "Genic_intronexonboundaries", "Genic_cds", "Genic_3UTRs", "Genic_intergenic" (the intergenic regions exclude the previous list of annotations), and "Genic_lncRNA" (GENCODE long non-coding RNA (lncRNA) transcripts).

17. FANTOM5 sample-ontology-enriched CAT genes

    - FANTOM5_CAT_Cell: a list (173 cell types) of GenomicRanges objects; each is an GR object containing CAT genes specifically expressed in a cell type.
    - FANTOM5_CAT_Tissue: a list (174 tissues) of GenomicRanges objects; each is an GR object containing CAT genes specifically expressed in a tissue.

18. FANTOM5 trait-associated CAT genes

    - FANTOM5_CAT_DO: a list (299 traits grouped by disease ontology) of GenomicRanges objects; each is an GR object containing CAT genes harboring at least one trait-associated SNP.
    - FANTOM5_CAT_EFO: a list (93 traits grouped by experiment factor ontology) of GenomicRanges objects; each is an GR object containing CAT genes harboring at least one trait-associated SNP.
    - FANTOM5_CAT_HPO: a list (176 traits grouped by human phenotype ontology) of GenomicRanges objects; each is an GR object containing CAT genes harboring at least one trait-associated SNP.
    - FANTOM5_CAT_MESH: a list (210 traits grouped by Medical Subject Headings) of GenomicRanges objects; each is an GR object containing CAT genes harboring at least one trait-associated SNP.
    - FANTOM5_CAT_PICS: a list (39 traits grouped by PICS dieases) of GenomicRanges objects; each is an GR object containing CAT genes harboring at least one trait-associated SNP.

19. GWAS Catalog trait-associated SNPs

    - GWAScatalog_alltraits: a list (390 traits grouped by EFO) of GenomicRanges objects; each is an GR object containing trait-associated SNPs.
    - GWAScatalog_bloodindex: a list (29 traits grouped by EFO) of GenomicRanges objects; each is an GR object containing trait-associated SNPs.

See Also

xEnrichViewer

Examples

```
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

## Not run:
grl <- xDefineGenomicAnno("Uniform_TFBS",
RData.location=RData.location)
grl <- xDefineGenomicAnno("Uniform_DNaseI_HS",
RData.location=RData.location)
grl <- xDefineGenomicAnno("FANTOM5_Enhancer_Cell",
RData.location=RData.location)
grl <- xDefineGenomicAnno("ReMap_Public_TFBS",
RData.location=RData.location)
grl <- xDefineGenomicAnno("EpigenomeAtlas_15Segments_E029",
RData.location=RData.location)
grl <- xDefineGenomicAnno("FANTOM5_CAT_Cell",
RData.location=RData.location)
grl <- xDefineGenomicAnno("GWAScatalog_alltraits",
RData.location=RData.location)

# the customised
## a GR object
GR.annotation <- grl[[1]]
grl_customised <- xDefineGenomicAnno(GR.annotation,
RData.location=RData.location)
## a list of GR objects
GR.annotation <- lapply(grl[1:2], function(x) x)
grl_customised <- xDefineGenomicAnno(GR.annotation,
RData.location=RData.location)

## End(Not run)
```

---

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

Description

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

Usage

```
xDefineHIC(data = NULL, entity = c("SNP", "chr:start-end",
"data.frame",
"bed", "GRanges"), include.HiC = c(NA, "Monocytes", "Macrophages_M0",
"Macrophages_M1", "Macrophages_M2", "Neutrophils", "Megakaryocytes",
"Endothelial_precursors", "Erythroblasts", "Fetal_thymus",
"Naive_CD4_T_cells", "Total_CD4_T_cells",
```

```
"Activated_total_CD4_T_cells",
"Nonactivated_total_CD4_T_cells", "Naive_CD8_T_cells",
"Total_CD8_T_cells",
"Naive_B_cells", "Total_B_cells", "PE.Monocytes", "PE.Macrophages_M0",
"PE.Macrophages_M1", "PE.Macrophages_M2", "PE.Neutrophils",
"PE.Megakaryocytes", "PE.Erythroblasts", "PE.Naive_CD4_T_cells",
"PE.Naive_CD8_T_cells", "Combined", "Combined_PE"), GR.SNP =
c("dbSNP_GWAS",
"dbSNP_Common", "dbSNP_Single"), verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

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

Value

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

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

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

- `from`: 'from/bait' genomic regions
- `to`: 'to/prey' genomic regions
- `score`: CHiCAGO scores quantifying the strength of physical interactions between baits and preys
- `from_genes`: genes associated with 'from/bait' genomic regions
- `to_genes`: genes associated with 'to/prey' genomic regions
- `SNP`: input SNPs (in query)
- `SNP_end`: specify which end SNPs in query fall into (either 'bait/from' or 'prey/to')
- `SNP_harbor`: genomic regions harbors the SNPs in query
- `Context`: the context in which PCHiC data was generated

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

Note

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

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

- Total_B_cells: promoter interactomes in Total B cells.
- Combined: promoter interactomes combined above; with score for the number of significant cell types plus scaled average.

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

- PE.Monocytes: physical interactions (CHiCAGO score >=5) of promoters (baits) with the other end (enhancers as preys) in Monocytes.
- PE.Macrophages_M0: promoter-enhancer interactomes in Macrophages M0.
- PE.Macrophages_M1: promoter-enhancer interactomes in Macrophages M1.
- PE.Macrophages_M2: promoter-enhancer interactomes in Macrophages M2.
- PE.Neutrophils: promoter-enhancer interactomes in Neutrophils.
- PE.Megakaryocytes: promoter-enhancer interactomes in Megakaryocytes.
- PE.Erythroblasts: promoter-enhancer interactomes in Erythroblasts.
- PE.Naive_CD4_T_cells: promoter-enhancer interactomes in Naive CD4+ T cells.
- PE.Naive_CD8_T_cells: promoter-enhancer interactomes in Naive CD8+ T cells.
- Combined_PE: promoter interactomes combined above; with score for the number of significant cell types plus scaled average.

See Also

xRDataLoader

Examples

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

## End(Not run)

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

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

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

```
xPCHiCplot(g, glayout=layout_in_circle, vertex.label.cex=0.5)

## End(Not run)
```

---

xDefineNet *Function to define a gene network*

Description

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

Usage

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

Arguments

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

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

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

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

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

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

Value

an object of class "igraph"

Note

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

See Also

`xCombineNet`

Examples

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

## End(Not run)

RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
# STRING (high quality)
g <- xDefineNet(network="STRING_high", RData.location=RData.location)
# STRING (high quality), with edges weighted
g <- xDefineNet(network="STRING_high", weighted=T,
RData.location=RData.location)
# STRING (high quality), only edges sourced from experimental or curated data
g <- xDefineNet(network="STRING_high",
STRING.only=c("experimental_score","database_score"),
RData.location=RData.location)

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

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

## End(Not run)
```

---

xDefineOntology *Function to define ontology and its annotations*

Description

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

Usage

```
xDefineOntology(ontology = c(NA, "GOBP", "GOMF", "GOCC", "PSG", "PS",
"PS2",
"SF", "Pfam", "DO", "HPPA", "HPMI", "HPCM", "HPMA", "MP", "EF",
"MsigdbH",
"MsigdbC1", "MsigdbC2CGP", "MsigdbC2CPall", "MsigdbC2CP",
"MsigdbC2KEGG",
"MsigdbC2REACTOME", "MsigdbC2BIOCARTA", "MsigdbC3TFT", "MsigdbC3MIR",
"MsigdbC4CGN", "MsigdbC4CM", "MsigdbC5BP", "MsigdbC5MF", "MsigdbC5CC",
```

```
"MsigdbC6", "MsigdbC7", "DGIdb", "GTExV4", "GTExV6p", "GTExV7",
"CreedsDisease", "CreedsDiseaseUP", "CreedsDiseaseDN", "CreedsDrug",
"CreedsDrugUP", "CreedsDrugDN", "CreedsGene", "CreedsGeneUP",
"CreedsGeneDN", "KEGG", "KEGGmetabolism", "KEGGgenetic",
"KEGGenvironmental",
"KEGGcellular", "KEGGorganismal", "KEGGdisease", "REACTOME",
"REACTOME_ImmuneSystem", "REACTOME_SignalTransduction", "CGL",
"SIFTS2GOBP",
"SIFTS2GOMF", "SIFTS2GOCC"), verbose = T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

ontology	the ontology supported currently. It can be "GOBP" for Gene Ontology Biological Process, "GOMF" for Gene Ontology Molecular Function, "GOCC" for Gene Ontology Cellular Component, "PSG" for phylostratigraphy (phylostratific age), "PS" for sTOL-based phylostratific age information, "PS2" for the collapsed PS version (inferred ancestors being collapsed into one with the known taxonomy information), "SF" for SCOP domain superfamilies, "Pfam" for Pfam domain families, "DO" for Disease Ontology, "HPPA" for Human Phenotype Phenotypic Abnormality, "HPMI" for Human Phenotype Mode of Inheritance, "HPCM" for Human Phenotype Clinical Modifier, "HPMA" for Human Phenotype Mortality Aging, "MP" for Mammalian Phenotype, "EF" for Experimental Factor Ontology (used to annotate GWAS Catalog genes), Drug-Gene Interaction database ("DGIdb") for druggable categories, tissue-specific eQTL-containing genes from GTEx ("GTExV4", "GTExV6p" and "GTExV7"), crowd extracted expression of differential signatures from CREEDS ("CreedsDisease", "CreedsDiseaseUP", "CreedsDiseaseDN", "CreedsDrug", "CreedsDrugUP", "CreedsDrugDN", "CreedsGene", "CreedsGeneUP" and "CreedsGeneDN"), KEGG pathways (including 'KEGG' for all, 'KEGGmetabolism' for 'Metabolism' pathways, 'KEGGgenetic' for 'Genetic Information Processing' pathways, 'KEGGenvironmental' for 'Environmental Information Processing' pathways, 'KEGGcellular' for 'Cellular Processes' pathways, 'KEGGorganismal' for 'Organismal Systems' pathways, and 'KEGGdisease' for 'Human Diseases' pathways), 'REACTOME' for REACTOME pathways or 'REACTOME_x' for its sub-ontologies (where x can be 'CellCellCommunication', 'CellCycle', 'CellularResponsesToExternalStimuli', 'ChromatinOrganization', 'CircadianClock', 'DevelopmentalBiology', 'DigestionAndAbsorption', 'Disease', 'DNARepair', 'DNAReplication', 'ExtracellularMatrixOrganization', 'GeneExpression(Transcription)', 'Hemostasis', 'ImmuneSystem', 'Metabolism', 'MetabolismOfProteins', 'MetabolismOfRNA', 'Mitophagy', 'MuscleContraction', 'NeuronalSystem', 'OrganelleBiogenesisAndMaintenance', 'ProgrammedCellDeath', 'Reproduction', 'SignalTransduction', 'TransportOfSmallMolecules', 'VesicleMediatedTransport'), and the molecular signatures database (Msigdb, including "MsigdbH", "MsigdbC1", "MsigdbC2CGP", "MsigdbC2CPall", "MsigdbC2CP", "MsigdbC2KEGG", "MsigdbC2REACTOME", "MsigdbC2BIOCARTA", "MsigdbC3TFT", "MsigdbC3MIR", "MsigdbC4CGN", "MsigdbC4CM", "MsigdbC5BP", "MsigdbC5MF", "MsigdbC5CC", "MsigdbC6", "MsigdbC7"), and the SIFTS database ("SIFTS2GOBP" for Gene Ontology Biological Process, "SIFTS2GOMF" for Gene Ontology Molecular Function,

	"SIFTS2GOCC" for Gene Ontology Cellular Component)
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 "aOnto", a list with two components (an igraph object 'g' and a list 'anno')

Note

none

See Also

xRDataLoader

Examples

```
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

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

## 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, bar.color = "lightyellow-orange", bar.width = 0.8,
wrap.width = NULL, font.family = "sans", signature = TRUE)
```

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
bar.color	either NULL or fill color names ('lightyellow-orange' by default)
bar.width	bar width. By default, 80 data
wrap.width	a positive integer specifying wrap width of name
font.family	the font family for texts
signature	logical to indicate whether the signature is assigned to the plot caption. By default, it sets TRUE showing which function is used to draw this graph

Value

an object of class "ggplot"

Note

none

See Also

xEnricherGenes, xEnricherSNPs, xEnrichViewer

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/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()

## End(Not run)

# 4) use font family (Arial)
## Not run:
source("http://bioconductor.org/biocLite.R"); biocLite("extrafont")
library(extrafont)
font_import()
fonttable()
## creating PDF files with fonts
library(extrafont)
loadfonts()
bp <- xEnrichBarplot(eTerm, top_num="auto", displayBy="fc",
font.family="Arial Black")
pdf(file="enrichment_barplot_fonts.pdf", height=6, width=12,
family="Arial Black")
print(bp)
dev.off()

## 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 = 2.5,
wrap.width = NULL, sharings = NULL, font.family = "sans",
facet = TRUE, signature = TRUE)
```

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
font.family	the font family for texts
facet	logical to indicate whether to facet/wrap a 1d of panels into 2d. By default, it sets TRUE
signature	logical to indicate whether the signature is assigned to the plot caption. By default, it sets TRUE showing which function is used to draw this graph

Value

an object of class "ggplot", 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 XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/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 redundant terms. By default, it has the first element 0.9 and the second element 0.5. It means, for a term (less significant; called 'B'), if there is a more significant term (called 'A'), their overlapped members cover at least 90 this term B will be defined as redundant and thus being removed
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to false for no display

Value

an object of class "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, ig = NULL, 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
ig	the igraph object. If provided, only those terms within it will be visualised. By default, it is NULL meaning no surch restriction
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 XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/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 XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/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"), background.annotatable.only = NULL, p.tail = c("one-tail",
"two-tails"), p.adjust.method = c("BH", "BY", "bonferroni", "holm",
"hochberg", "hommel"), ontology.algorithm = c("none", "pc", "elim",
"lea"),
elim.pvalue = 0.01, lea.depth = 2, path.mode = c("all_paths",
"shortest_paths", "all_shortest_paths"), true.path.rule = TRUE,
verbose = 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 test statistic used. It can be "fisher" for using fisher's exact test, "hypergeo" for using hypergeometric test, or "binomial" for using binomial test. Fisher's exact test is to test the independence between gene group (genes belonging to a group or not) and gene annotation (genes annotated by a term or not), and thus compare sampling to the left part of background (after sampling without replacement). Hypergeometric test is to sample at random (without replacement) from the background containing annotated and non-annotated genes, and thus compare sampling to background. Unlike hypergeometric test, binomial test is to sample at random (with replacement) from the background with the constant probability. In terms of the ease of finding the significance, they are in order: hypergeometric test > fisher's exact test > binomial test. In other words, in terms of the calculated p-value, hypergeometric test < fisher's exact test < binomial test

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

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

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

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

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

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

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

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

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

Value

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

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

Note

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

- "none": does not consider the ontology hierarchy at all.
- "lea": estimates the significance of a term in terms of the significance of its children at the maximum depth (e.g. 2). Precisely, once snps/genes are already annotated to any children terms with a more signficance than itself, then all these snps/genes are eliminated from the use for the recalculation of the signifance at that term. The final p-values takes the maximum of the original p-value and the recalculated p-value.
- "elim": estimates the significance of a term in terms of the significance of its all children. Precisely, once snps/genes are already annotated to a signficantly enriched term under the cutoff of e.g. pvalue<1e-2, all these snps/genes are eliminated from the ancestors of that term).

- "pc": requires the significance of a term not only using the whole snps/genes as background but also using snps/genes annotated to all its direct parents/ancestors as background. The final p-value takes the maximum of both p-values in these two calculations.
- "Notes": the order of the number of significant terms is: "none" > "lea" > "elim" > "pc".

See Also

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

Arguments

data	an input vector containing gene symbols
background	a background vector containing gene symbols as the test background. If NULL, by default all annotatable are used as background
check.symbol.identity	logical to indicate whether to match the input data/background via Synonyms for those unmatchable by official gene symbols. By default, it sets to false
ontology	the ontology supported currently. By default, it is 'NA' to disable this option. Pre-built ontology and annotation data are detailed in xDefineOntology.
size.range	the minimum and maximum size of members of each term in consideration. By default, it sets to a minimum of 10 but no more than 2000
min.overlap	the minimum number of overlaps. Only those terms with members that overlap with input data at least min.overlap (3 by default) will be processed

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

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

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

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

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

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

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

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

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

true.path.rule	logical to indicate whether the true-path rule should be applied to propagate annotations. By default, it sets to false
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to false for no display
silent	logical to indicate whether the messages will be silent completely. By default, it sets to false. If true, verbose will be forced to be false
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details

Value

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

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

Note

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

- "none": does not consider the ontology hierarchy at all.

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

See Also

xDefineOntology, xEnricher

Examples

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

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

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

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

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

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

# e) barplot of significant enrichment results
```

```
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"), graph.node.attrs=list(fontsize=25))
# color-code terms according to the z-scores
xEnrichDAGplot(eTerm, top_num=10, displayBy="zscore",
node.info=c("full_term_name"), graph.node.attrs=list(fontsize=25))

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

## End(Not run)
```

---

xEnricherGenesAdv *Function to conduct enrichment analysis given a list of gene sets and a list of ontologies*

Description

xEnricherGenesAdv is supposed to conduct enrichment analysis given a list of gene sets and a list of ontologies. It is an advanced version of xEnricherGenes, returning an object of the class 'ls_eTerm'.

Usage

```
xEnricherGenesAdv(list_vec, background = NULL, check.symbol.identity =
F,
ontologies = NA, size.range = c(10, 2000), min.overlap = 3,
which.distance = NULL, test = c("hypergeo", "fisher", "binomial"),
background.annotatable.only = NULL, p.tail = c("one-tail",
"two-tails"),
p.adjust.method = c("BH", "BY", "bonferroni", "holm", "hochberg",
"hommel"),
ontology.algorithm = c("none", "pc", "elim", "lea"), elim.pvalue =
0.01,
lea.depth = 2, path.mode = c("all_paths", "shortest_paths",
"all_shortest_paths"), true.path.rule = F, verbose = T, silent = FALSE,
plot = TRUE, fdr.cutoff = 0.05, displayBy = c("zscore", "fdr",
```

```
"pvalue",
"fc", "or"), RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

list_vec	an input vector containing gene symbols. Alternatively it can be a list of vectors, representing multiple groups of genes
background	a background vector containing gene symbols as the test background. If NULL, by default all annotatable are used as background
check.symbol.identity	logical to indicate whether to match the input data/background via Synonyms for those unmatchable by official gene symbols. By default, it sets to false
ontologies	the ontologies supported currently. Pre-built ontology and annotation data are detailed in xDefineOntology.
size.range	the minimum and maximum size of members of each term in consideration. By default, it sets to a minimum of 10 but no more than 2000
min.overlap	the minimum number of overlaps. Only those terms with members that overlap with input data at least min.overlap (3 by default) will be processed
which.distance	which terms with the distance away from the ontology root (if any) is used to restrict terms in consideration. By default, it sets to 'NULL' to consider all distances
test	the test statistic used. It can be "fisher" for using fisher's exact test, "hypergeo" for using hypergeometric test, or "binomial" for using binomial test. Fisher's exact test is to test the independence between gene group (genes belonging to a group or not) and gene annotation (genes annotated by a term or not), and thus compare sampling to the left part of background (after sampling without replacement). Hypergeometric test is to sample at random (without replacement) from the background containing annotated and non-annotated genes, and thus compare sampling to background. Unlike hypergeometric test, binomial test is to sample at random (with replacement) from the background with the constant probability. In terms of the ease of finding the significance, they are in order: hypergeometric test > fisher's exact test > binomial test. In other words, in terms of the calculated p-value, hypergeometric test < fisher's exact test < binomial test
background.annotatable.only	logical to indicate whether the background is further restricted to the annotatable. By default, it is NULL: if ontology.algorithm is not 'none', it is always TRUE; otherwise, it depends on the background (if not provided, it will be TRUE; otherwise FALSE). Surely, it can be explicitly stated
p.tail	the tail used to calculate p-values. It can be either "two-tails" for the significance based on two-tails (ie both over- and under-overrepresentation) or "one-tail" (by default) for the significance based on one tail (ie only over-representation)
p.adjust.method	the method used to adjust p-values. It can be one of "BH", "BY", "bonferroni", "holm", "hochberg" and "hommel". The first two methods "BH" (widely used) and "BY" control the false discovery rate (FDR: the expected proportion of false

discoveries amongst the rejected hypotheses); the last four methods "bonferroni", "holm", "hochberg" and "hommel" are designed to give strong control of the family-wise error rate (FWER). Notes: FDR is a less stringent condition than FWER

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

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

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

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

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

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

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

`plot`
: logical to indicate whether heatmap plot is drawn

`fdr.cutoff`
: fdr cutoff used to declare the significant terms. By default, it is set to 0.05. This option only works when setting plot (see above) is TRUE

`displayBy`
: which statistics will be used for drawing heatmap. It can be "fc" for enrichment fold change, "fdr" for adjusted p value (or FDR), "pvalue" for p value, "zscore" for enrichment z-score (by default), "or" for odds ratio. This option only works when setting plot (see above) is TRUE

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

Value

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

- `df`: a data frame of n x 12, where the 12 columns are "group" (the input group names), "ontology" (input ontologies), "id" (term ID), "name" (term name), "nAnno" (number in members annotated by a term), "nOverlap" (number in overlaps), "fc" (enrichment fold changes), "zscore" (enrichment z-score), "pvalue" (nominal p value), "adjp" (adjusted p value (FDR)), "or" (odds ratio), "CIl" (lower bound confidence interval for the odds ratio), "CIu" (upper bound confidence interval for the odds ratio), "distance" (term distance or other information), "members" (members (represented as Gene Symbols) in overlaps)

- mat: NULL if the plot is not drawn; otherwise, a matrix of term names X groups with numeric values for the signficant enrichment, NA for the insignificant ones
- gp: NULL if the plot is not drawn; otherwise, a 'ggplot' object

Note

none

See Also

xRDataLoader, xEnricherGenes, xEnrichViewer, xHeatmap

Examples

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

# Gene-based enrichment analysis using ontologies (REACTOME and GOMF)
# a) provide the input Genes of interest (eg 100 randomly chosen human genes)
## load human genes
org.Hs.eg <- xRDataLoader(RData='org.Hs.eg',
RData.location=RData.location)
set.seed(825)
data <- as.character(sample(org.Hs.eg$gene_info$Symbol, 100))
data

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

# b) perform enrichment analysis
ls_eTerm <- xEnricherGenesAdv(data, ontologies=c("REACTOME","GOMF"),
RData.location=RData.location)
ls_eTerm
## forest plot of enrichment results
gp <- xEnrichForest(ls_eTerm, top_num=10)
## heatmap plot of enrichment results
gp <- xEnrichHeatmap(ls_eTerm, fdr.cutoff=0.1, displayBy="or")

## 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"),
background.annotatable.only = NULL, p.tail = c("one-tail",
"two-tails"),
p.adjust.method = c("BH", "BY", "bonferroni", "holm", "hochberg",
"hommel"),
ontology.algorithm = c("none", "pc", "elim", "lea"), elim.pvalue =
0.01,
lea.depth = 2, path.mode = c("all_paths", "shortest_paths",
"all_shortest_paths"), true.path.rule = T, verbose = T, silent = FALSE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

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

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

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

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

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

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

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

path.mode	the mode of paths induced by vertices/nodes with input annotation data. It can be "all_paths" for all possible paths to the root, "shortest_paths" for only one path to the root (for each node in query), "all_shortest_paths" for all shortest paths to the root (i.e. for each node, find all shortest paths with the equal lengths)
true.path.rule	logical to indicate whether the true-path rule should be applied to propagate annotations. By default, it sets to true
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to false for no display
silent	logical to indicate whether the messages will be silent completely. By default, it sets to false. If true, verbose will be forced to be false
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details

Value

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

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

Note

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

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

See Also

xRDataLoader, xEnricher

Examples

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

# 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', RData.location=RData.location)
ind <- which(colnames(anno)=='EFO:0002690')
data <- rownames(anno)[anno[,ind]!=0]
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"), 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
#eTerm <- xEnricherSNPs(data=data, include.LD="EUR", LD.r2=0.8, RData.location=RData.location)

# 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"), background.annotatable.only = NULL,
p.tail = c("one-tail", "two-tails"), p.adjust.method = c("BH", "BY",
"bonferroni", "holm", "hochberg", "hommel"), verbose = T, silent =
FALSE)
```

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

`background.annotatable.only`
: logical to indicate whether the background is further restricted to the annotatable (covered by 'annotation.file'). By default, it is NULL: if the background not provided, it will be TRUE; otherwise FALSE. Surely, it can be explicitly stated. Notably, if only one annotation is provided in 'annotation.file', it should be false (also the background.file should be provided)

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

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

verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to false for no display
silent	logical to indicate whether the messages will be silent completely. By default, it sets to false. If true, verbose will be forced to be false

Value

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

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

Note

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)

# Using ImmunoBase SNPs and associations/annotations with disease traits
## get ImmunoBase
RData.location <- "http://galahad.well.ox.ac.uk/bigdata/"
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
## 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',
RData.location=RData.location)
```

```
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)

## End(Not run)
```

---

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

Description

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

Usage

```
xEnrichForest(eTerm, top_num = 10, FDR.cutoff = 0.05, CI.one = T,
colormap = "ggplot2.top", ncolors = 64, zlim = NULL, barwidth = 0.5,
barheight = NULL, wrap.width = NULL, font.family = "sans",
signature = TRUE, drop = F)
```

Arguments

eTerm	an object of class "eTerm" or "ls_eTerm". Alterntively, it can be a data frame having all these columns (named as 'group','ontology','name','adjp','or','CIl','CIu')
top_num	the number of the top terms (sorted according to OR). If it is 'auto', only the significant terms (see below FDR.cutoff) will be displayed
FDR.cutoff	FDR cutoff used to declare the significant terms. By default, it is set to 0.05
CI.one	logical to indicate whether to allow the inclusion of one in CI. By default, it is TURE (allowed)
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
ncolors	the number of colors specified over the colormap

zlim	the minimum and maximum z values for which colors should be plotted, defaulting to the range of the -log10(FDR)
barwidth	the width of the colorbar. Default value is 'legend.key.width' or 'legend.key.size' in 'theme' or theme
barheight	the height of the colorbar. Default value is 'legend.key.height' or 'legend.key.size' in 'theme' or theme
wrap.width	a positive integer specifying wrap width of name
font.family	the font family for texts
signature	logical to indicate whether the signature is assigned to the plot caption. By default, it sets TRUE showing which function is used to draw this graph
drop	logical to indicate whether all factor levels not used in the data will automatically be dropped. If FALSE (by default), all factor levels will be shown, regardless of whether or not they appear in the data

Value

an object of class "ggplot"

Note

none

See Also

xEnricherGenes, xEnricherSNPs, xEnrichViewer

Examples

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

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

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

# 1) Gene-based enrichment analysis using REACTOME pathways
# perform enrichment analysis
eTerm <- xEnricherGenes(data, ontology="REACTOME",
RData.location=RData.location)
## forest plot of enrichment results
```

```
gp <- xEnrichForest(eTerm, top_num="auto", FDR.cutoff=0.05)

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

## End(Not run)
```

---

xEnrichHeatmap *Function to visualise enrichment results using heatmap*

Description

xEnrichHeatmap is supposed to visualise enrichment results using heatmap. It returns an object of class "ggplot".

Usage

```
xEnrichHeatmap(list_eTerm, fdr.cutoff = 0.05, displayBy = c("zscore",
"fdr",
"pvalue", "fc", "or"), colormap = NULL, zlim = NULL, reorder =
c("none",
"row", "col", "both"))
```

Arguments

list_eTerm	an object of class "ls_eTerm". Alterntively, it can be a data frame having all these columns (named as 'group','ontology','name','adjp') and one of these columns ("zscore","fdr","pvalue","fc","or"). Note, the column 'fdr' can be inferred from the column 'adjp'
fdr.cutoff	FDR cutoff used to declare the significant terms. By default, it is set to 0.05
displayBy	which statistics will be used for comparison. It can be "fc" for enrichment fold change (by default), "adjp" for adjusted p value (or FDR), "pvalue" for p value, "zscore" for enrichment z-score, "or" for enrichment odd ratio
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
zlim	the minimum and maximum z values for which colors should be plotted, defaulting to the range of the -log10(FDR)

reorder	how to reorder rows and columns. It can be "none" for no reordering, "row" for reordering rows according to number of sharings (by default), "col" for reordering columns, and "both" for reordering rows and columns

Value

an object of class "ggplot"

Note

none

See Also

xHeatmap

Examples

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

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

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

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

## End(Not run)
```

---

xEnrichLadder *Function to visualise enrichment results using ladder-like plot*

Description

xEnrichLadder is supposed to visualise enrichment results using ladder-like plot in which rows for terms and columns for its members. The members are sorted first by sharings and then by individual terms. It returns an object of class "ggplot".

Usage

```
xEnrichLadder(eTerm, sortBy = c("or", "adjp", "fdr", "pvalue",
"zscore", "fc",
"nAnno", "nOverlap", "none"), top_num = 10, FDR.cutoff = 0.05,
CI.one = T, colormap = "skyblue-darkblue", x.rotate = 60,
x.text.size = 6, y.text.size = 6, shape = 19, size = 2,
label = c("concise", "full"), verbose = T)
```

Arguments

eTerm	an object of class "eTerm"
sortBy	which statistics will be used for sorting and viewing gene sets (terms). It can be "adjp" or "fdr" for adjusted p value (FDR), "pvalue" for p value, "zscore" for enrichment z-score, "fc" for enrichment fold change, "nAnno" for the number of sets (terms), "nOverlap" for the number in overlaps, "or" for the odds ratio, and "none" for ordering according to ID of terms
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
FDR.cutoff	FDR cutoff used to declare the significant terms. By default, it is set to 0.05
CI.one	logical to indicate whether to allow the inclusion of one in CI. By default, it is TURE (allowed)
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
x.rotate	the angle to rotate the x tick labelings. By default, it is 60
x.text.size	the text size of the x tick labelings. By default, it is 6
y.text.size	the text size of the y tick labelings. By default, it is 6
shape	the number specifying the shape. By default, it is 19
size	the number specifying the shape size. By default, it is 2
label	how to label gene sets (terms). It can be "concise" or "full"
verbose	logical to indicate whether the messages will be displayed in the screen

Value

an object of class "ggplot"

Note

none

See Also

xEnrichViewer, xHeatmap

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata/"

data(Haploid_regulators)
## only IRF1 positive regulators
data <- subset(Haploid_regulators, Phenotype=='IRF1' &
MI<0)[,c('Gene')]

# 1) KEGGenvironmental
eTerm <- xEnricherGenes(data, ontology="KEGGenvironmental",
size.range=c(10,2000), min.overlap=5, RData.location=RData.location)
gp_ladder <- xEnrichLadder(eTerm)

# 2) PSG
eTerm <- xEnricherGenes(data,
ontology=c("PSG","Approved","GWAS","CGL")[1], size.range=c(1,20000),
min.overlap=0, RData.location=RData.location)
gp_ladder <- xEnrichLadder(eTerm, sortBy="none", top_num="auto",
FDR.cutoff=1)
gp_ladder+ coord_flip()

# 3) save into the file "xEnrichLadder.pdf"
mat <- xSparseMatrix(gp_ladder$data)
pdf("xEnrichLadder.pdf", width=2+ncol(mat)*0.075,
height=2+nrow(mat)*0.1, compress=T)
print(gp_ladder)
dev.off()

# 4) SIFTS2GOMF
## df_fpocket
SIFTS_fpocket <-
xRDataLoader(RData='SIFTS_fpocket',RData.location=RData.location)
df_fpocket <- as.data.frame(SIFTS_fpocket %>%
dplyr::filter(druggable=='Y') %>% dplyr::group_by(Symbol,PDB_code)
%>% dplyr::summarise(num_pockets=n()) %>%
dplyr::arrange(Symbol,desc(num_pockets),PDB_code))
df_fpocket <- df_fpocket[!duplicated(df_fpocket$Symbol), ]
## mat_fpocket
mat_fpocket <- df_fpocket %>% tidyr::spread(Symbol, num_pockets)
rownames(mat_fpocket) <- mat_fpocket[,1]
mat_fpocket <- mat_fpocket[,-1]
## gp_ladder
set.seed(825)
data <- as.character(sample(unique(df_fpocket$Symbol), 100))
eTerm <- xEnricherGenes(data=data, ontology="SIFTS2GOMF",
```

```
RData.location=RData.location)
gp_ladder <- xEnrichLadder(eTerm, sortBy="none", top_num=5,
FDR.cutoff=0.01, x.rotate=90)
#gp_ladder + coord_flip()
## data_matrix
ind <- match(colnames(gp_ladder$matrix), colnames(mat_fpocket))
data_matrix <- mat_fpocket[,ind[!is.na(ind)]]
ind <- which(apply(!is.na(data_matrix), 1, sum)!=0)
data_matrix <- data_matrix[ind,]
ind <- match(data, colnames(data_matrix))
data_matrix <- data_matrix[,ind[!is.na(ind)]]
## gp_pdb
gp_pdb <- xHeatmap(t(data_matrix), reorder="row", colormap="jet.top",
x.rotate=90, shape=19, size=1, x.text.size=6,y.text.size=5,
na.color='transparent', legend.title='# pockets')
#gp_pdb + coord_flip()
## plot_combined
#plot_combined <- cowplot::plot_grid(gp_ladder, gp_pdb, align="h", ncol=1, rel_heights=c(2,3))

## enrichment analysis
SIFTS_fpocket <-
xRDataLoader(RData='SIFTS_fpocket',RData.location=RData.location)
annotation.file <- SIFTS_fpocket[!duplicated(SIFTS_fpocket$Symbol),
c('Symbol','druggable')]
### 100 randomly chosen human genes
org.Hs.eg <- xRDataLoader(RData='org.Hs.eg',
RData.location=RData.location)
set.seed(825)
data <- as.character(sample(org.Hs.eg$gene_info$Symbol, 100))
### optionally, provide the test background (if not provided, all human genes)
background <- as.character(org.Hs.eg$gene_info$Symbol)
### perform enrichment analysis
eTerm <- xEnricherYours(data.file=data,
annotation.file=annotation.file, background.file=background,
size.range=c(10,20000))

## End(Not run)
```

---

xEnrichMatrix *Function to compare enrichment results using matrix plots*

Description

xEnrichMatrix is supposed to compare enrichment results using matrix plots.

Usage

```
xEnrichMatrix(list_eTerm, method = c("ggplot2", "circle", "square",
"color",
"pie"), displayBy = c("zscore", "fc", "adjp", "pvalue"),
```

```
FDR.cutoff = 0.05, wrap.width = NULL, sharings = NULL,
reorder = c("row", "none", "col", "both"), colormap = "jet",
ncolors = 20, zlim = NULL, slim = NULL, title = NULL, flip = FALSE,
y.rotate = 45, shape = 19, font.family = "sans", ...)
```

Arguments

list_eTerm	a list of "eTerm" objects, or a data frame (with at least 3 columns "group", "name" and "adjp")
method	which method will be used for plotting. It can be "circle" (by default), "square", "color" and "pie"
displayBy	which statistics will be used for comparison. It can be "fc" for enrichment fold change (by default), "adjp" 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
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
reorder	how to reorder rows and columns. It can be "none" for no reordering, "row" for reordering rows according to number of sharings (by default), "col" for reordering columns, and "both" for reordering rows and columns
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
ncolors	the number of colors specified over the colormap
zlim	the minimum and maximum z values for which colors should be plotted, defaulting to the range of the finite values of displayed matrix
slim	the minimum and maximum displaying values for which sizes should be plotted
title	the title of the plot. By default, it is NULL
flip	logical to indicate whether to flip the coordiate. By default, it sets to false
y.rotate	the angle to rotate the y tick labelings. By default, it is 45
shape	the number specifying the shape. By default, it is 19
font.family	the font family for texts
...	additional graphic parameters for corrplot::corrplot

Value

If the method is 'ggplot2', it returns a ggplot object. Otherwise, it is a data frame

Note

none

See Also

xEnricherGenes, xEnricherSNPs, xEnrichViewer

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata/"
xEnrichMatrix(list_eTerm, method="circle", displayBy="adjp",
FDR.cutoff=0.05, wrap.width=50, sharings=NULL, reorder="row",
colormap="black-yellow-red", ncolors=16, zlim=c(0,8), cl.pos="b",
cl.ratio=0.1, cl.align.text="c", tl.col="black", tl.cex=0.7, tl.srt=90,
title=paste0(ontology,": log10(FDR)"))
xEnrichMatrix(list_eTerm, method="pie", displayBy="adjp",
FDR.cutoff=0.05, wrap.width=50, sharings=NULL, reorder="row",
colormap="grey-grey", ncolors=1, zlim=c(0,8), cl.pos="n", cl.ratio=0.1,
cl.align.text="c", tl.col="black", tl.cex=0.7, tl.srt=90,
title=paste0(ontology,": log10(FDR)"))
gp <- xEnrichMatrix(list_eTerm, method="ggplot2", displayBy="zscore",
FDR.cutoff=0.05, wrap.width=40, sharings=NULL, reorder="row",
colormap="yellow-red", flip=T, y.rotate=45, font.family=font.family)

## End(Not run)
```

---

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 XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/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)
```

---

xEnrichTreemap *Function to visualise enrichment results using a treemap*

Description

xEnrichTreemap is supposed to visualise enrichment results using a treemap. The area is proportional to odds ratio, colored by the significance level. It returns an object of class "ggplot".

Usage

```
xEnrichTreemap(eTerm, top_num = 10, FDR.cutoff = 0.05, CI.one = T,
colormap = "spectral.top", ncolors = 64, zlim = NULL, barwidth = NULL,
barheight = 0.5, wrap.width = NULL, font.family = "sans", drop = F,
details = c("name", "name_FDR", "name_FDR_members"), caption = T,
treemap.grow = F, treemap.reflow = F, treemap.place = "topleft",
treemap.color = "black", treemap.fontface = "bold.italic",
treemap.min.size = 4)
```

Arguments

eTerm	an object of class "eTerm" or "ls_eTerm". Alterntively, it can be a data frame having all these columns (named as 'group','ontology','name','adjp','or','CIl','CIu','nOverlap','members
top_num	the number of the top terms (sorted according to OR). If it is 'auto', only the significant terms (see below FDR.cutoff) will be displayed

FDR.cutoff	FDR cutoff used to declare the significant terms. By default, it is set to 0.05
CI.one	logical to indicate whether to allow the inclusion of one in CI. By default, it is TURE (allowed)
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
ncolors	the number of colors specified over the colormap
zlim	the minimum and maximum z values for which colors should be plotted, defaulting to the range of the -log10(FDR)
barwidth	the width of the colorbar. Default value is 'legend.key.width' or 'legend.key.size' in 'theme' or theme
barheight	the height of the colorbar. Default value is 'legend.key.height' or 'legend.key.size' in 'theme' or theme
wrap.width	a positive integer specifying wrap width of name
font.family	the font family for texts
drop	logical to indicate whether all factor levels not used in the data will automatically be dropped. If FALSE (by default), all factor levels will be shown, regardless of whether or not they appear in the data
details	how to label. It can be one of 'name', 'name_FDR' (FDR/OR also appended to the name), and 'name_FDR_members' (FDR/OR plus members appended to the name; in this case, treemap.grow and treemap.reflow is forced to be true)
caption	logical to indicate whether the caption is shown on the bottom-right
treemap.grow	logical to indicate whether text will be grown as well as shrunk to fill the box
treemap.reflow	logical to indicate whether text will be reflowed (wrapped) to better fit the box
treemap.place	where inside the box to place the text. Default is "centre"; other options are "bottom", "top", "topleft", "topright", etc
treemap.color	the color of the text
treemap.fontface	the fontface of the text
treemap.min.size	the minimum font size, in points. If provided, text that would need to be shrunk below this size to fit the box will not be drawn. Defaults to 4 pt

Value

an object of class "ggplot"

Note

none

See Also

xEnricherGenes, xEnricherSNPs, xEnrichViewer

Examples

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

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

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

# 1) Gene-based enrichment analysis using REACTOME pathways
# perform enrichment analysis
eTerm <- xEnricherGenes(data, ontology="REACTOME",
RData.location=RData.location)
## forest plot of enrichment results
gp <- xEnrichTreemap(eTerm, top_num=20, FDR.cutoff=0.05,
treemap.reflow=F, treemap.place="topleft")

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

## End(Not run)
```

---

xEnrichViewer *Function to view enrichment results*

Description

xEnrichViewer is supposed to view results of enrichment analysis.

Usage

```
xEnrichViewer(eTerm, top_num = 10, sortBy = c("adjp", "fdr", "pvalue",
"zscore", "fc", "nAnno", "nOverlap", "or", "none"), decreasing = NULL,
details = 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, "or" for the odds ratio, and "none" for ordering according to ID of terms
decreasing	logical to indicate whether to sort in a decreasing order. If it is null, it would be true for "zscore", "nAnno" or "nOverlap"; otherwise it would be false
details	logical to indicate whether the detailed information of gene sets (terms) is also viewed. By default, it sets to false for no inclusion

Value

a data frame with following components:

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

xGGnetwork *Function to visualise an igraph object using ggnetwork*

Description

xGGnetwork is supposed to visualise an igraph object using ggnetwork.

Usage

```
xGGnetwork(g, node.label = NULL, label.wrap.width = NULL,
label.wrap.lineheight = 0.8, node.label.size = NULL,
node.label.fontface = "plain", node.label.color = "darkblue",
node.label.alpha = 0.8, node.label.padding = 1, node.label.arrow =
0.01,
node.label.force = 1, node.shape = 19, node.shape.title = NULL,
node.xcoord = NULL, node.ycoord = NULL, node.color = NULL,
node.color.title = NULL, colormap = "grey-orange-darkred", ncolors =
64,
zlim = NULL, na.color = "grey80", node.size = NULL,
node.size.title = NULL, node.size.range = c(1, 4), slim = NULL,
title = "", edge.size = 0.5, edge.color = "black",
edge.color.alpha = 0.5, edge.curve = 0.1, edge.arrow = 2,
edge.arrow.gap = 0.02)
```

Arguments

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

`node.label.arrow`
: the arrow pointing to the labeled node

`node.label.force`
: the repelling force between overlapping labels

`node.shape`
: an integer specifying node shape or a character specifying which node attribute used for the node shape (no matter whether it is numeric or character)

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

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

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

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

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

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

`ncolors`
: the number of colors specified over the colormap

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

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

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

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

`node.size.range`
: the range of actual node size

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

`title`
: a character specifying the title for the plot

`edge.size`
: a numeric value specifying the edge size. By default, it is 0.5

`edge.color`
: a character specifying which edge attribute defining the the edge colors

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

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

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

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

Value

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

Note

none

See Also

xGGnetwork

Examples

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

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

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

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

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

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

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

```
ncolors=64, zlim=c(0,5), node.size.range=5,
edge.color="orange",edge.color.alpha=0.3,edge.curve=0,edge.arrow.gap=0.02,
title='')

## End(Not run)
```

---

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

Description

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

Usage

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

Arguments

data	input genomic regions (GR). If formatted as "chr:start-end" (see the next parameter 'format' below), GR should be provided as a vector in the format of 'chrN:start-end', where N is either 1-22 or X, start (or end) is genomic positional number; for example, 'chr1:13-20'. If formatted as a 'data.frame', the first three columns correspond to the chromosome (1st column), the starting chromosome position (2nd column), and the ending chromosome position (3rd column). If the format is indicated as 'bed' (browser extensible data), the same as 'data.frame' format but the position is 0-based offset from chromomose position. If the genomic regions provided are not ranged but only the single position, the ending chromosome position (3rd column) is allowed not to be provided. The data could also be an object of 'GRanges' (in this case, formatted as 'GRanges')
format	the format of the input data. It can be one of "chr:start-end", "data.frame", "bed" or "GRanges"
build.conversion	the conversion from one genome build to another. The conversions supported are "hg38.to.hg19" and "hg18.to.hg19". By default it is NA (no need to do so)
add.name	logical to add names. By default, it sets to true
remove.mcol	logical to remove meta-columns. By default, it sets to false
include.strand	logical to include strand. By default, it sets to false. It only works when the format is "data.frame" or "bed" and the input data has 4 columns
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details

Value

a GenomicRanges object

See Also

xRDataLoader

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

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

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

## End(Not run)
```

---

xGR2GeneScores — *Function to identify likely modulated seed genes given a list of genomic regions together with the significance level*

Description

xGR2GeneScores is supposed to identify likely modulated seed genes from a list of genomic regions (GR) together with the significance level (measured as p-values or fdr). To do so, it defines seed genes and their scores that take into account the distance to and the significance of input SNPs. It returns an object of class "mSeed".

Usage

```
xGR2GeneScores(data, significance.threshold = 5e-05, score.cap = 10,
build.conversion = c(NA, "hg38.to.hg19", "hg18.to.hg19"),
distance.max = 50000, decay.kernel = c("slow", "linear", "rapid",
"constant"), decay.exponent = 2, GR.Gene = c("UCSC_knownGene",
```

```
"UCSC_knownCanonical"), scoring.scheme = c("max", "sum", "sequential"),
verbose = T, RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

data	a named input vector containing the sinificance level for genomic regions (GR). For this named vector, the element names are GR, in the format of 'chrN:start-end', where N is either 1-22 or X, start (or end) is genomic positional number; for example, 'chr1:13-20'. The element values for the significance level (measured as p-value or fdr). Alternatively, it can be a matrix or data frame with two columns: 1st column for GR, 2nd column for the significance level.
significance.threshold	the given significance threshold. By default, it is set to NULL, meaning there is no constraint on the significance level when transforming the significance level of GR into scores. If given, those GR below this are considered significant and thus scored positively. Instead, those above this are considered insignificant and thus receive no score
score.cap	the maximum score being capped. By default, it is set to 10. If NULL, no capping is applied
build.conversion	the conversion from one genome build to another. The conversions supported are "hg38.to.hg19" and "hg18.to.hg19". By default it is NA (no need to do so)
distance.max	the maximum distance between genes and GR. Only those genes no far way from this distance will be considered as seed genes. This parameter will influence the distance-component weights calculated for nearby GR per gene
decay.kernel	a character specifying a decay kernel function. It can be one of 'slow' for slow decay, 'linear' for linear decay, and 'rapid' for rapid decay. If no distance weight is used, please select 'constant'
decay.exponent	a numeric specifying a decay exponent. By default, it sets to 2
GR.Gene	the genomic regions of genes. By default, it is 'UCSC_knownGene', that is, UCSC known genes (together with genomic locations) based on human genome assembly hg19. It can be 'UCSC_knownCanonical', that is, UCSC known canonical genes (together with genomic locations) based on human genome assembly hg19. Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to Gene Symbols. Then, tell "GR.Gene" with your RData file name (with or without extension), plus specify your file RData path in "RData.location". Note: you can also load your customised GR object directly
scoring.scheme	the method used to calculate seed gene scores under a set of GR. It can be one of "sum" for adding up, "max" for the maximum, and "sequential" for the sequential weighting. The sequential weighting is done via: $\sum_{i=1} \frac{R_i}{i}$, where R_i is the i^{th} rank (in a descreasing order)
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:

- `GR`: a matrix of nGR X 3 containing GR information, where nGR is the number of GR, and the 3 columns are "GR" (genomic regions), "Score" (the scores for GR calculated based on p-values taking into account the given threshold of the significant level), "Pval" (the input p-values for GR)
- `Gene`: a matrix of nGene X 3 containing Gene information, where nGene is the number of seed genes, and the 3 columns are "Gene" (gene symbol), "Score" (the scores for seed genes), "Pval" (pvalue-like significance level transformed from gene scores)
- `call`: the call that produced this result

Note

This function uses `xGRscores` and `xGR2nGenes` to define and score nearby genes that are located within distance window of input genomic regions.

See Also

`xGRscores`, `xGR2nGenes`, `xSparseMatrix`

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

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

# b) define and score seed geens
mSeed <- xGR2GeneScores(data=data, RData.location=RData.location)

## End(Not run)
```

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

Description

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

Usage

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

Arguments

data	input genomic regions (GR). If formatted as "chr:start-end" (see the next parameter 'format' below), GR should be provided as a vector in the format of 'chrN:start-end', where N is either 1-22 or X, start (or end) is genomic positional number; for example, 'chr1:13-20'. If formatted as a 'data.frame', the first three columns correspond to the chromosome (1st column), the starting chromosome position (2nd column), and the ending chromosome position (3rd column). If the format is indicated as 'bed' (browser extensible data), the same as 'data.frame' format but the position is 0-based offset from chromomose position. If the genomic regions provided are not ranged but only the single position, the ending chromosome position (3rd column) is allowed not to be provided. The data could also be an object of 'GRanges' (in this case, formatted as 'GRanges')
format	the format of the input data. It can be one of "data.frame", "chr:start-end", "bed" or "GRanges"
build.conversion	the conversion from one genome build to another. The conversions supported are "hg38.to.hg19" and "hg18.to.hg19". By default it is NA (no need to do so)
distance.max	the maximum distance between genes and GR. Only those genes no far way from this distance will be considered as seed genes. This parameter will influence the distance-component weights calculated for nearby GR per gene
decay.kernel	a character specifying a decay kernel function. It can be one of 'slow' for slow decay, 'linear' for linear decay, and 'rapid' for rapid decay. If no distance weight is used, please select 'constant'
decay.exponent	a numeric specifying a decay exponent. By default, it sets to 2

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

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

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

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

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

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

Value

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

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

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

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

Note

For details on the decay kernels, please refer to xVisKernels

See Also

xRDataLoader, xGR

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

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

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

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

## End(Not run)
```

---

xGR2xGeneAnno *Function to conduct region-based enrichment analysis via crosslinked genes*

Description

xGR2xGeneAnno is supposed to conduct region-based enrichment analysis for the input genomic region data (genome build h19), using crosslinked gene annotations. To do so, crosslinked genes are first defined. Currently supported built-in crosslink info is enhancer genes, eQTL genes, conformation genes and nearby genes (purely), though the user can customise it via 'crosslink.customised';

if so, it has priority over the built-in data. Enrichment analysis is then based on either Fisher's exact test or Hypergeometric test for estimating the significance of overlapped crosslinked genes. Test background can be provided; by default, the annotatable genes will be used.

Usage

```
xGR2xGeneAnno(data, background = NULL, format = c("data.frame", "bed",
"chr:start-end", "GRanges"), build.conversion = c(NA, "hg38.to.hg19",
"hg18.to.hg19"), crosslink = c("genehancer", "PCHiC_combined",
"GTEx_V6p_combined", "nearby"), crosslink.customised = NULL,
crosslink.top = NULL, nearby.distance.max = 50000,
nearby.decay.kernel = c("rapid", "slow", "linear", "constant"),
nearby.decay.exponent = 2, ontology = NA, size.range = c(10, 2000),
min.overlap = 3, which.distance = NULL, test = c("hypergeo", "fisher",
"binomial"), background.annotatable.only = NULL, p.tail = c("one-tail",
"two-tails"), p.adjust.method = c("BH", "BY", "bonferroni", "holm",
"hochberg", "hommel"), ontology.algorithm = c("none", "pc", "elim",
"lea"),
elim.pvalue = 0.01, lea.depth = 2, path.mode = c("all_paths",
"shortest_paths", "all_shortest_paths"), true.path.rule = F, verbose =
T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

data	input genomic regions (GR). If formatted as "chr:start-end" (see the next parameter 'format' below), GR should be provided as a vector in the format of 'chrN:start-end', where N is either 1-22 or X, start (or end) is genomic positional number; for example, 'chr1:13-20'. If formatted as a 'data.frame', the first three columns correspond to the chromosome (1st column), the starting chromosome position (2nd column), and the ending chromosome position (3rd column). If the format is indicated as 'bed' (browser extensible data), the same as 'data.frame' format but the position is 0-based offset from chromomose position. If the genomic regions provided are not ranged but only the single position, the ending chromosome position (3rd column) is allowed not to be provided. The data could also be an object of 'GRanges' (in this case, formatted as 'GRanges')
background	an input background containing a list of genomic regions as the test background. The file format is the same as 'data' above. By default, it is NULL meaning all annotatable genes are used as background
format	the format of the input data. It can be one of "data.frame", "chr:start-end", "bed" or "GRanges"
build.conversion	the conversion from one genome build to another. The conversions supported are "hg38.to.hg19" and "hg18.to.hg19". By default it is NA (no need to do so)
crosslink	the built-in crosslink info with a score quantifying the link of a GR to a gene. See xGR2xGenes for details

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

`crosslink.top`
: the number of the top genes defined by 'data' will be used for test. By default, it is NULL

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

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

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

`ontology`
: the ontology supported currently. By default, it is 'NA' to disable this option. Pre-built ontology and annotation data are detailed in `xDefineOntology`.

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

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

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

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

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

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

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

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

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

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

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

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

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

`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
- `or`: a vector containing odds ratio
- `CIl`: a vector containing lower bound confidence interval for the odds ratio
- `CIu`: a vector containing upper bound confidence interval for the odds ratio
- `cross`: a matrix of nTerm X nTerm, with an on-diagnal cell for the overlapped-members observed in an individaul term, and off-diagnal cell for the overlapped-members shared betwene two terms
- `call`: the call that produced this result
- `evidence`: a data frame with 3 columns ('GR' for genomic regions, 'Gene' for crosslinked genes, and 'Score' for the score between the gene and the GR)
- `gp_evidence`: a ggplot object for evidence data

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

`xGR`, `xGR2xGenes`, `xEnricherGenes`

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

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

## b) perform DO enrichment analysis
## enhancer genes
eTerm <- xGR2xGeneAnno(data=dGR, format="GRanges",
crosslink="genehancer", ontology="DO", RData.location=RData.location)
## nearby genes (50kb, decaying rapidly)
eTerm <- xGR2xGeneAnno(data=dGR, format="GRanges", crosslink="nearby",
ontology="DO", nearby.distance.max=50000, nearby.decay.kernel="rapid",
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=10, displayBy="fc")
print(bp)

## f) forest of significant enrichment results
gp <- xEnrichForest(eTerm, top_num=10)

## End(Not run)
```

---

xGR2xGeneAnnoAdv *Function to conduct region-based enrichment analysis via crosslinked genes given a list of genomic region sets and a list of ontologies*

Description

xGR2xGeneAnnoAdv is supposed to conduct enrichment analysis given a list of gene sets and a list of ontologies, using crosslinked gene annotations. It is an advanced version of xGR2xGeneAnno, returning an object of the class 'ls_eTerm'.

Usage

```
xGR2xGeneAnnoAdv(list_vec, background = NULL, build.conversion = c(NA,
"hg38.to.hg19", "hg18.to.hg19"), crosslink = c("genehancer",
"PCHiC_combined", "GTEx_V6p_combined", "nearby"),
crosslink.customised = NULL, crosslink.top = NULL,
nearby.distance.max = 50000, nearby.decay.kernel = c("rapid", "slow",
"linear", "constant"), nearby.decay.exponent = 2, ontologies = NA,
size.range = c(10, 2000), min.overlap = 3, which.distance = NULL,
test = c("hypergeo", "fisher", "binomial"),
background.annotatable.only = NULL, p.tail = c("one-tail",
"two-tails"),
p.adjust.method = c("BH", "BY", "bonferroni", "holm", "hochberg",
"hommel"),
ontology.algorithm = c("none", "pc", "elim", "lea"), elim.pvalue =
0.01,
lea.depth = 2, path.mode = c("all_paths", "shortest_paths",
"all_shortest_paths"), true.path.rule = F, verbose = T, silent = FALSE,
plot = TRUE, fdr.cutoff = 0.05, displayBy = c("zscore", "fdr",
"pvalue",
"fc", "or"), RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

list_vec	an input vector containing genomic regions. Alternatively it can be a list of vectors, representing multiple groups of genomic regions. Formatted as "chr:start-end" are genomic regions
background	a background vector containing genomic regions (formatted as "chr:start-end") as the test background. If NULL, by default all annotatable are used as background
build.conversion	the conversion from one genome build to another. The conversions supported are "hg38.to.hg19" and "hg18.to.hg19". By default it is NA (no need to do so)
crosslink	the built-in crosslink info with a score quantifying the link of a GR to a gene. See xGR2xGenes for details
crosslink.customised	the crosslink info with a score quantifying the link of a GR to a gene. A user-input matrix or data frame with 4 columns: 1st column for genomic regions (formatted as "chr:start-end", genome build 19), 2nd column for Genes, 3rd for crosslink score (crosslinking a genomic region to a gene, such as -log10 significance level), and 4th for contexts (optional; if not provided, it will be added as 'C'). Alternatively, it can be a file containing these 4 columns. Required, otherwise it will return NULL
crosslink.top	the number of the top genes defined by 'data' will be used for test. By default, it is NULL
nearby.distance.max	the maximum distance between genes and GR. Only those genes no far way from this distance will be considered as seed genes. This parameter will influence the distance-component weights calculated for nearby GR per gene

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

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

`ontologies`
: the ontologies supported currently. By default, it is 'NA' to disable this option. Pre-built ontology and annotation data are detailed in `xDefineOntology`.

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

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

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

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

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

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

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

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

elim.pvalue	the parameter only used when "ontology.algorithm" is "elim". It is used to control how to declare a signficantly enriched term (and subsequently all genes in this term are eliminated from all its ancestors)
lea.depth	the parameter only used when "ontology.algorithm" is "lea". It is used to control how many maximum depth is used to consider the children of a term (and subsequently all genes in these children term are eliminated from the use for the recalculation of the signifance at this term)
path.mode	the mode of paths induced by vertices/nodes with input annotation data. It can be "all_paths" for all possible paths to the root, "shortest_paths" for only one path to the root (for each node in query), "all_shortest_paths" for all shortest paths to the root (i.e. for each node, find all shortest paths with the equal lengths)
true.path.rule	logical to indicate whether the true-path rule should be applied to propagate annotations. By default, it sets to false
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to false for no display
silent	logical to indicate whether the messages will be silent completely. By default, it sets to false. If true, verbose will be forced to be false
plot	logical to indicate whether heatmap plot is drawn
fdr.cutoff	fdr cutoff used to declare the significant terms. By default, it is set to 0.05. This option only works when setting plot (see above) is TRUE
displayBy	which statistics will be used for drawing heatmap. It can be "fc" for enrichment fold change, "fdr" for adjusted p value (or FDR), "pvalue" for p value, "zscore" for enrichment z-score (by default), "or" for odds ratio. This option only works when setting plot (see above) is TRUE
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details

Value

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

- df: a data frame of n x 12, where the 12 columns are "group" (the input group names), "ontology" (input ontologies), "id" (term ID), "name" (term name), "nAnno" (number in members annotated by a term), "nOverlap" (number in overlaps), "fc" (enrichment fold changes), "zscore" (enrichment z-score), "pvalue" (nominal p value), "adjp" (adjusted p value (FDR)), "or" (odds ratio), "CIl" (lower bound confidence interval for the odds ratio), "CIu" (upper bound confidence interval for the odds ratio), "distance" (term distance or other information), "members" (members (represented as Gene Symbols) in overlaps)
- mat: NULL if the plot is not drawn; otherwise, a matrix of term names X groups with numeric values for the signficant enrichment, NA for the insignificant ones
- gp: NULL if the plot is not drawn; otherwise, a 'ggplot' object

Note

none

See Also

xGR2xGeneAnno, xEnrichViewer, xHeatmap

Examples

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

# Enrichment analysis for GWAS SNPs from ImmunoBase
## a) provide input data (bed-formatted)
data.file <- "http://galahad.well.ox.ac.uk/bigdata/ImmunoBase_GWAS.bed"
input <- read.delim(file=data.file, header=T, stringsAsFactors=F)
data <- paste0(input$chrom, ':', (input$chromStart+1), '-',
input$chromEnd)

# b) perform enrichment analysis
## overlap with gene body
ls_eTerm <- xGR2xGeneAnnoAdv(data, crosslink="genehancer",
ontologies=c("REACTOME_ImmuneSystem","REACTOME_SignalTransduction"),
RData.location=RData.location)
ls_eTerm
## forest plot of enrichment results
gp <- xEnrichForest(ls_eTerm, top_num=10, CI.one=F)
gp

## End(Not run)
```

---

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

Description

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

Usage

```
xGR2xGenes(data, format = c("chr:start-end", "data.frame", "bed",
"GRanges"),
build.conversion = c(NA, "hg38.to.hg19", "hg18.to.hg19"),
crosslink = c("genehancer", "PCHiC_combined", "GTEx_V6p_combined",
"nearby"), crosslink.customised = NULL, cdf.function = c("original",
"empirical"), scoring = F, scoring.scheme = c("max", "sum",
```

```
"sequential"),
scoring.rescale = F, nearby.distance.max = 50000,
nearby.decay.kernel = c("rapid", "slow", "linear", "constant"),
nearby.decay.exponent = 2, verbose = T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

data	input genomic regions (GR). If formatted as "chr:start-end" (see the next parameter 'format' below), GR should be provided as a vector in the format of 'chrN:start-end', where N is either 1-22 or X, start (or end) is genomic positional number; for example, 'chr1:13-20'. If formatted as a 'data.frame', the first three columns correspond to the chromosome (1st column), the starting chromosome position (2nd column), and the ending chromosome position (3rd column). If the format is indicated as 'bed' (browser extensible data), the same as 'data.frame' format but the position is 0-based offset from chromomose position. If the genomic regions provided are not ranged but only the single position, the ending chromosome position (3rd column) is allowed not to be provided. The data could also be an object of 'GRanges' (in this case, formatted as 'GRanges')
format	the format of the input data. It can be one of "data.frame", "chr:start-end", "bed" or "GRanges"
build.conversion	the conversion from one genome build to another. The conversions supported are "hg38.to.hg19" and "hg18.to.hg19". By default it is NA (no need to do so)
crosslink	the built-in crosslink info with a score quantifying the link of a GR to a gene. It can be one of 'genehancer' (enhancer genes; PMID:28605766), 'nearby' (nearby genes; if so, please also specify the relevant parameters 'nearby.distance.max', 'nearby.decay.kernel' and 'nearby.decay.exponent' below), 'PCHiC_combined' (conformation genes; PMID:27863249), 'GTEx_V6p_combined' (eQTL genes; PMID:29022597), 'eQTL_scRNAseq_combined' (eQTL genes; PMID:29610479), 'eQTL_ImmuneCells_combined' (eQTL genes; PMID:29610479)
crosslink.customised	the crosslink info with a score quantifying the link of a GR to a gene. A user-input matrix or data frame with 4 columns: 1st column for genomic regions (formatted as "chr:start-end", genome build 19), 2nd column for Genes, 3rd for crosslink score (crosslinking a genomic region to a gene, such as -log10 significance level), and 4th for contexts (optional; if not provided, it will be added as 'C'). Alternatively, it can be a file containing these 4 columns. Required, otherwise it will return NULL
cdf.function	a character specifying how to transform the input crosslink score. It can be one of 'original' (no such transformation), and 'empirical' for looking at empirical Cumulative Distribution Function (cdf; as such it is converted into pvalue-like values [0,1])
scoring	logical to indicate whether gene-level scoring will be further calculated. By default, it sets to false
scoring.scheme	the method used to calculate seed gene scores under a set of GR. It can be one of "sum" for adding up, "max" for the maximum, and "sequential" for the

sequential weighting. The sequential weighting is done via: $\sum_{i=1} \frac{R_i}{i}$, where R_i is the i^{th} rank (in a descreasing order)

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

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

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

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

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

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

Value

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

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

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

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

See Also

`xRDataLoader`, `xGR`

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
```

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

# 2) using built-in crosslink info
## enhancer genes
df_xGenes <- xGR2xGenes(dGR, format="GRanges", crosslink="genehancer",
RData.location=RData.location)
## conformation genes
df_xGenes <- xGR2xGenes(dGR, format="GRanges",
crosslink="PCHiC_combined", RData.location=RData.location)
## eQTL genes
df_xGenes <- xGR2xGenes(dGR, format="GRanges",
crosslink="GTEx_V6p_combined", RData.location=RData.location)
## nearby genes (50kb, decaying rapidly)
df_xGenes <- xGR2xGenes(dGR, format="GRanges", crosslink="nearby",
nearby.distance.max=50000, nearby.decay.kernel="rapid",
RData.location=RData.location)

# 3) advanced use
# 3a) provide crosslink.customised
## illustration purpose only (see the content of 'crosslink.customised')
df <- xGR2nGenes(dGR, format="GRanges", RData.location=RData.location)
crosslink.customised <- data.frame(GR=df$GR, Gene=df$Gene,
Score=df$Weight, Context=rep('C',nrow(df)), stringsAsFactors=F)
#crosslink.customised <- data.frame(GR=df$GR, Gene=df$Gene, Score=df$Weight, stringsAsFactors=F)
# 3b) define crosslinking genes
# without gene scoring
df_xGenes <- xGR2xGenes(dGR, format="GRanges",
crosslink.customised=crosslink.customised,
RData.location=RData.location)
# with gene scoring
df_xGenes <- xGR2xGenes(dGR, format="GRanges",
crosslink.customised=crosslink.customised, scoring=T,
scoring.scheme="max", RData.location=RData.location)

## End(Not run)
```

---

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

Description

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

Usage

```
xGR2xGeneScores(data, significance.threshold = NULL, score.cap = NULL,
build.conversion = c(NA, "hg38.to.hg19", "hg18.to.hg19"),
crosslink = c("genehancer", "PCHiC_combined", "GTEx_V6p_combined",
"nearby"), crosslink.customised = NULL, cdf.function = c("original",
"empirical"), scoring.scheme = c("max", "sum", "sequential"),
nearby.distance.max = 50000, nearby.decay.kernel = c("rapid", "slow",
"linear", "constant"), nearby.decay.exponent = 2, verbose = T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

data	a named input vector containing the sinificance level for genomic regions (GR). For this named vector, the element names are GR, in the format of 'chrN:start-end', where N is either 1-22 or X, start (or end) is genomic positional number; for example, 'chr1:13-20'. The element values for the significance level (measured as p-value or fdr). Alternatively, it can be a matrix or data frame with two columns: 1st column for GR, 2nd column for the significance level.
significance.threshold	the given significance threshold. By default, it is set to NULL, meaning there is no constraint on the significance level when transforming the significance level of GR into scores. If given, those GR below this are considered significant and thus scored positively. Instead, those above this are considered insignificant and thus receive no score
score.cap	the maximum score being capped. By default, it is set to NULL, meaning that no capping is applied
build.conversion	the conversion from one genome build to another. The conversions supported are "hg38.to.hg19" and "hg18.to.hg19". By default it is NA (no need to do so)
crosslink	the built-in crosslink info with a score quantifying the link of a GR to a gene. See xGR2xGenes for details
crosslink.customised	the crosslink info with a score quantifying the link of a GR to a gene. A user-input matrix or data frame with 4 columns: 1st column for genomic regions (formatted as "chr:start-end", genome build 19), 2nd column for Genes, 3rd for crosslink score (crosslinking a genomic region to a gene, such as -log10 significance level), and 4th for contexts (optional; if nor provided, it will be added as 'C'). Alternatively, it can be a file containing these 4 columns. Required, otherwise it will return NULL

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

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

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

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

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

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

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

Value

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

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

Note

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

See Also

`xGRscores`, `xGR2xGenes`, `xSparseMatrix`

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

# a) provide the seed SNPs with the significance info
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
## get lead SNPs reported in AS GWAS and their significance info (p-values)
gr <- ImmunoBase$AS$variant
df <- as.data.frame(gr, row.names=NULL)
GR <- paste0(df$seqnames,':',df$start,'-',df$end)
data <- cbind(GR=GR, Sig=df$Pvalue)

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

## End(Not run)
```

---

xGR2xNet *Function to identify a gene network from an input network given a list of genomic regions*

Description

xGR2xNet is supposed to identify maximum-scoring gene subnetwork from an input graph with the node information (genomic regions with or without the significance). To do so, it defines seed genes and their scores that take into account the distance to and the significance of input genomic regions (GR). It returns an object of class "igraph".

Usage

```
xGR2xNet(data, significance.threshold = NULL, score.cap = NULL,
build.conversion = c(NA, "hg38.to.hg19", "hg18.to.hg19"),
crosslink = c("genehancer", "PCHiC_combined", "GTEx_V6p_combined",
"nearby"), crosslink.customised = NULL, cdf.function = c("original",
"empirical"), scoring.scheme = c("max", "sum", "sequential"),
nearby.distance.max = 50000, nearby.decay.kernel = c("rapid", "slow",
"linear", "constant"), nearby.decay.exponent = 2,
network = c("STRING_highest", "STRING_high", "STRING_medium",
"STRING_low",
"PCommonsUN_high", "PCommonsUN_medium", "PCommonsDN_high",
"PCommonsDN_medium", "PCommonsDN_Reactome", "PCommonsDN_KEGG",
"PCommonsDN_HumanCyc", "PCommonsDN_PID", "PCommonsDN_PANTHER",
"PCommonsDN_ReconX", "PCommonsDN_TRANSFAC", "PCommonsDN_PhosphoSite",
```

```
"PCommonsDN_CTD", "KEGG", "KEGG_metabolism", "KEGG_genetic",
"KEGG_environmental", "KEGG_cellular", "KEGG_organismal",
"KEGG_disease",
"REACTOME"), network.customised = NULL, seed.genes = T,
subnet.significance = 5e-05, subnet.size = NULL, verbose = T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

`data`	a named input vector containing the significance level for genomic regions (GR). For this named vector, the element names are GR, in the format of 'chrN:start-end', where N is either 1-22 or X, start (or end) is genomic positional number; for example, 'chr1:13-20', the element values for the significance level (measured as p-value or fdr). Alternatively, it can be a matrix or data frame with two columns: 1st column for GR, 2nd column for the significance level. Also supported is the input with GR only (without the significance level)
`significance.threshold`	the given significance threshold. By default, it is set to NULL, meaning there is no constraint on the significance level when transforming the significance level of GR into scores. If given, those GR below this are considered significant and thus scored positively. Instead, those above this are considered insignificant and thus receive no score
`score.cap`	the maximum score being capped. By default, it is set to NULL, meaning that no capping is applied
`build.conversion`	the conversion from one genome build to another. The conversions supported are "hg38.to.hg19" and "hg18.to.hg19". By default it is NA (no need to do so)
`crosslink`	the built-in crosslink info with a score quantifying the link of a GR to a gene. See `xGR2xGenes` for details
`crosslink.customised`	the crosslink info with a score quantifying the link of a GR to a gene. A user-input matrix or data frame with 4 columns: 1st column for genomic regions (formatted as "chr:start-end", genome build 19), 2nd column for Genes, 3rd for crosslink score (crosslinking a genomic region to a gene, such as -log10 significance level), and 4th for contexts (optional; if nor provided, it will be added as 'C'). Alternatively, it can be a file containing these 4 columns. Required, otherwise it will return NULL
`cdf.function`	a character specifying how to transform the input crosslink score. It can be one of 'original' (no such transformation), and 'empirical' for looking at empirical Cumulative Distribution Function (cdf; as such it is converted into pvalue-like values [0,1])
`scoring.scheme`	the method used to calculate seed gene scores under a set of GR (also over Contexts if many). It can be one of "sum" for adding up, "max" for the maximum, and "sequential" for the sequential weighting. The sequential weighting is done via: $\sum_{i=1} \frac{R_i}{i}$, where R_i is the i^{th} rank (in a descreasing order)

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

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

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

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

`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 graph attributes (evidence, gp_evidence) and node attributes (significance, score).

Note

The algorithm identifying a gene subnetwork that is likely modulated by input genomic regions (GR) includes two major steps. The first step is to use `xGR2xGeneScores` for defining and scoring nearby genes that are located within distance window of input GR. 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

`xGR2xGeneScores`, `xSubneterGenes`

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata/"

# a) provide the seed SNPs with the significance info
data(ImmunoBase)
## only AS GWAS SNPs and their significance info (p-values)
df <- as.data.frame(ImmunoBase$AS$variant, row.names=NULL)
GR <- paste0(df$seqnames,':',df$start,'-',df$end)
```

```
data <- cbind(GR=GR, Sig=df$Pvalue)

# b) perform network analysis
# b1) find maximum-scoring subnet based on the given significance threshold
subnet <- xGR2xNet(data=data, crosslink="genehancer",
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 <- xGR2xNet(data=data, crosslink="genehancer",
network="STRING_high", seed.genes=F, subnet.size=30,
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
xVisNet(g=subnet, pattern=-log10(as.numeric(V(subnet)$significance)),
vertex.shape="sphere", colormap="wyr")
## do visualisation with nodes colored according to transformed scores
xVisNet(g=subnet, pattern=as.numeric(V(subnet)$score),
vertex.shape="sphere")

# e) visualise the identified subnet as a circos plot
library(RCircos)
xCircos(g=subnet, entity="Gene", colormap="orange-darkred", ideogram=F,
entity.label.side="out", chr.exclude=NULL,
RData.location=RData.location)

## End(Not run)
```

---

xGraphML *Function to generate a graphml file from a graph object of class "igraph"*

Description

xGraphML is supposed to generate a graphml file from a graph object of class "igraph".

Usage

```
xGraphML(g, node.label = NULL, label.wrap.width = NULL,
node.label.size = 12, node.label.color = "#000000", node.tooltip =
NULL,
node.link = NULL, node.xcoord = "xcoord", node.ycoord = "ycoord",
```

```
node.color.na = "#dddddd", node.color = NULL,
colormap = "grey-orange-darkred", ncolors = 64, nlegend = 11,
legend.label.size = 10, legend.interval = 0.05, zlim = NULL,
node.size = 30, node.coord.scale = 300, edge.color = "#00000033",
edge.width = 1, filename = "xGraphML")
```

Arguments

g	an object of class "igraph"
node.label	either a vector labelling nodes or a character specifying which node attribute used for the labelling. If NULL (by default), no node labelling. If provided as a vector, a node with 'NA' will be not labelled
label.wrap.width	a positive integer specifying wrap width of name
node.label.size	the node label size
node.label.color	the node label color
node.tooltip	either a vector used for node tooltips or a character specifying which node attribute used for the tooltips. If NULL (by default), node attribute 'name' will be used node lab
node.link	a string specifying hyperlink address. By default, it is NULL meaning no hyperlink
node.xcoord	a vector specifying x coordinates. If NULL, it will be created using igraph::layout_with_kk
node.ycoord	a vector specifying y coordinates. If NULL, it will be created using igraph::layout_with_kk
node.color.na	the color for nodes with NA. By default, it is '#dddddd'
node.color	a character specifying which node attribute used for node coloring. If NULL (by default), it is '#BFFFBF'
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta), and "ggplot2" (emulating ggplot2 default color palette). Alternatively, any hyphen-separated HTML color names, e.g. "lightyellow-orange" (by default), "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
ncolors	the number of colors specified over the colormap
nlegend	the number of colors specified in the legend. By default, it is 11
legend.label.size	the legend label size. By default, it is 10
legend.interval	the interval between legends. By default, it is 0.05

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
node.size	either a vector specifying node size or a character specifying which node attribute used for the node size. If NULL (by default), it will be 30
node.coord.scale	the node coord (-1,1) subjected to be rescaled. By default, it is 300
edge.color	a character specifying the edge colors. By default, it is #00000033
edge.width	the edge width. By default, it is 1
filename	the without-extension part of the name of the output file. By default, it is 'xGraphML'

Value

invisible (a string storing graphml-formatted content). If the filename is not NULL, a graphml-formatted file is also output.

Note

none

See Also

xGraphML

Examples

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

# 1) load REACTOME
# 1a) restricted to Immune System ('R-HSA-168256') or Signal Transduction ('R-HSA-162582')
g <- xRDataLoader(RData.customised='ig.REACTOME',
RData.location=RData.location)
neighs.out <- igraph::neighborhood(g, order=vcount(g),
nodes="R-HSA-168256", mode="out")
nodeInduced <- V(g)[unique(unlist(neighs.out))]$name
ig <- igraph::induced.subgraph(g, vids=nodeInduced)
# visualise the graph with vertices being color-coded by the pattern
V(ig)$pattern <- runif(vcount(ig))
xGraphML(g=ig, node.label="name", node.color="pattern", colormap="wyr",
node.size=10, node.label.size=6)

# 1b) restricted to Signal Transduction ('R-HSA-162582')
g <- xRDataLoader(RData.customised='ig.REACTOME',
RData.location=RData.location)
neighs.out <- igraph::neighborhood(g, order=vcount(g),
```

```
nodes="R-HSA-162582", mode="out")
nodeInduced <- V(g)[unique(unlist(neighs.out))]$name
ig <- igraph::induced.subgraph(g, vids=nodeInduced)
# visualise the graph with vertices being color-coded by the pattern
V(ig)$pattern <- runif(vcount(ig))
xGraphML(g=ig, node.label="name", node.color="pattern", colormap="wyr",
node.size=8, node.label.size=4)

###########################
# visualise gene network
glayout <- igraph::layout_with_kk(ig)
V(ig)$xcoord <- glayout[,1]
V(ig)$ycoord <- glayout[,2]
xGraphML(g=ig, node.label="name", node.tooltip="description",
node.xcoord="xcoord", node.ycoord="ycoord", node.color="pattern",
colormap="grey-orange-darkred",
node.link="http://www.genecards.org/cgi-bin/carddisp.pl?gene=",
nlegend=11, node.size=30, node.coord.scale=300)

## End(Not run)
```

`xGraphSplit` *Function to split a graph according to a node attribute*

Description

`xGraphSplit` is supposed to split a graph according to a node attribute such as community or comp.

Usage

```
xGraphSplit(g, node.attr = NULL, verbose = TRUE)
```

Arguments

`g`	an object of class "igraph" (or "graphNEL") for a graph with such as a 'community' node attribute
`node.attr`	a charatter specifying a node attribute. If NULL or no match, it returns NULL
`verbose`	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

It returns a list of igraph objects.

See Also

`xGraphSplit`

Examples

```
# 1) generate a random bipartite graph
set.seed(123)
g <- sample_bipartite(100, 50, p=0.1)
V(g)$name <- V(g)

## Not run:
# 2) obtain and append the community
cs <- igraph::cluster_louvain(g)
V(g)$community <- cs$membership
ls_ig <- xGraphSplit(g, node.attr="community")

## End(Not run)
```

---

xGRcse *Function to create a vector for genomic regions*

Description

xGRcse is supposed to create genomic regions in the format of 'chr:start-end'.

Usage

```
xGRcse(data, format = c("GRanges", "data.frame", "bed"))
```

Arguments

data	input genomic regions (GR). If formatted as a 'data.frame', the first three columns correspond to the chromosome (1st column), the starting chromosome position (2nd column), and the ending chromosome position (3rd column). If the format is indicated as 'bed' (browser extensible data), the same as 'data.frame' format but the position is 0-based offset from chromomose position. If the genomic regions provided are not ranged but only the single position, the ending chromosome position (3rd column) is allowed not to be provided. The data could also be an object of 'GRanges' (in this case, formatted as 'GRanges')
format	the format of the input data. It can be one of "data.frame", "bed" or "GRanges"

Value

a vector for genomic regions the format of 'chrN:start-end'

See Also

xGRcse

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

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

# b) create a GRanges object
cse <- xGRcse(gr)

## End(Not run)
```

---

xGRkaryogram *Function to visualise genomic regions using karyogram plot*

Description

xGRkaryogram is supposed to visualise genomic regions using manhattan plot. It returns an object of class "ggplot".

Usage

```
xGRkaryogram(gr, cytoband = F, color = "royalblue", size = 0.5,
label = F, label.size = 2, label.col = "magenta", label.force = 0.05,
label.query = NULL, verbose = T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

gr	a GenomicRange object. If the meta-column 'label' is not provided, it will the name of this object
cytoband	logical to indicate whether cytoband will be displayed. By default, it sets to false
color	the rect color. By default it is 'royalblue'
size	the rect size
label	logical to indicate whether to label the rect. By default, it sets to false
label.size	the label size
label.col	the label color ('magenta' by default)
label.force	the repelling force between overlapping labels
label.query	only query will be labelled. By default, it sets to NULL meaning all will be displayed. If labels in query can not be found, then all will be displayed

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 ggplot object.

Note

none

See Also

xGRkaryogram

Examples

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

## End(Not run)

RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
### GWAS catalog
GWAScatalog <- xRDataLoader('GWAScatalog',
RData.location=RData.location)
gwas <- xGR(GWAScatalog$cse_hg19, format="chr:start-end")
ind <- match(names(gwas), GWAScatalog$cse_hg19)
names(gwas) <- GWAScatalog$snp_id_current[ind]
gwas$label <- names(gwas)
gp <- xGRkaryogram(gwas)
gp

## End(Not run)
```

---

xGRmanhattan *Function to visualise genomic regions using manhattan plot*

Description

xGRmanhattan is supposed to visualise genomic regions using manhattan plot. It returns an object of class "ggplot".

Usage

```
xGRmanhattan(gr, chromosome.only = TRUE, color = c("royalblue",
"sandybrown"), y.scale = c("normal", "sqrt", "log"), y.lab = NULL,
top = NULL, top.label.type = c("text", "box"), top.label.size = 2,
top.label.col = "black", top.label.force = 0.05, top.label.query =
NULL,
label.query.only = FALSE, top.label.chr = T, verbose = TRUE)
```

Arguments

gr	a GenomicRange object with a meta-column 'value'. If the meta-column 'label' is not provided, it will the name of this object
chromosome.only	logical to indicate whether only those from input data will be displayed. By default, it sets to TRUE
color	a character vector for colors to alternate chromosome colorings. If NULL, ggplot2 default colors will be used. If a single character is provided, it can be "jet" (jet colormap) or "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta)
y.scale	how to transform the y scale. It can be "normal" for no transformation, "sqrt" for square root transformation, and "log" for log-based transformation
y.lab	the y labelling. If NULL (by default), it shows the column of input data
top	the number of the top targets to be labelled/highlighted
top.label.type	how to label the top targets. It can be "box" drawing a box around the labels , and "text" for the text only
top.label.size	the highlight label size
top.label.col	the highlight label color
top.label.force	the repelling force between overlapping labels
top.label.query	which top genes in query will be labelled. By default, it sets to NULL meaning all top genes will be displayed. If labels in query can not be found, then all will be displayed
label.query.only	logical to indicate whether only those in query will be displayed. By default, it sets to FALSE. It only works when labels in query are enabled/found
top.label.chr	logical to indicate whether the top hit per chromosome will be displayed. By default, it sets to TRUE. It only works when the parameter 'top' is null
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to false for no display

Value

a ggplot object.

Note

none

See Also

xGRmanhattan

Examples

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

## End(Not run)

RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
### GWAS catalog
GWAScatalog <- xRDataLoader('GWAScatalog',
RData.location=RData.location)
gwas <- xGR(GWAScatalog$cse_hg19, format="chr:start-end")
ind <- match(names(gwas), GWAScatalog$cse_hg19)
gwas$value <- -log10(GWAScatalog$pvalue[ind])
names(gwas) <- GWAScatalog$snp_id_current[ind]
gwas$label <- names(gwas)
gp <- xGRmanhattan(gwas)
gp

## End(Not run)
```

---

xGRoverlap *Function to extract overlap-based scores given a list of genomic regions*

Description

xGRoverlap is supposed to extract overlap-based scores given a list of genomic regions. Scores are extracted for overlapped sub-regions only, valued at the mean per base; otherwise NA. It returns a GR object.

Usage

```
xGRoverlap(data, format = c("chr:start-end", "data.frame", "bed",
"GRanges"),
build.conversion = c(NA, "hg38.to.hg19", "hg18.to.hg19"), GR.score =
c(NA,
"RecombinationRate", "phastCons100way", "phyloP100way", "GERP"),
verbose = T, RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

data	input genomic regions (GR). If formatted as "chr:start-end" (see the next parameter 'format' below), GR should be provided as a vector in the format of 'chrN:start-end', where N is either 1-22 or X, start (or end) is genomic positional number; for example, 'chr1:13-20'. If formatted as a 'data.frame', the first three columns correspond to the chromosome (1st column), the starting chromosome position (2nd column), and the ending chromosome position (3rd column). If the format is indicated as 'bed' (browser extensible data), the same as 'data.frame' format but the position is 0-based offset from chromomose position. If the genomic regions provided are not ranged but only the single position, the ending chromosome position (3rd column) is allowed not to be provided. The data could also be an object of 'GRanges' (in this case, formatted as 'GRanges')
format	the format of the input data. It can be one of "data.frame", "chr:start-end", "bed" or "GRanges"
build.conversion	the conversion from one genome build to another. The conversions supported are "hg38.to.hg19" and "hg18.to.hg19". By default it is NA (no need to do so)
GR.score	the genomic regions together with score data. By default, it is 'NA' to disable this option. Pre-built genomic score data: 'RecombinationRate' (recombintion rate, http://www.ncbi.nlm.nih.gov/pubmed/17943122)), 'phastCons100way', 'phyloP100way', 'GERP'. Beyond pre-built data, the user can specify the customised input: load your customised GR object directly (with the first meta column for scores; if not provided, it will be valued at 1)
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 GenomicRanges object, appended with a meta-column 'GScore'. If input data contains only a genomic region, then outputs are all overlapped regions from GR.score; otherwise all overlapped regions from input data will be output.

See Also

xRDataLoader

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

# a) provide the genomic regions
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
```

```
RData.location=RData.location)
## get lead SNPs reported in AS GWAS
data <- ImmunoBase$AS$variant

# b) extract recombination rate
gr <- xGRoverlap(data=data, format="GRanges",
GR.score="RecombinationRate", RData.location=RData.location)

##########################################
# gene-centric genomic score (per base)
gr_Gene <- xRDataLoader('UCSC_knownGene',
RData.location=RData.location)
## recombination rate
gr_rr <- xGRoverlap(data=gr_Gene, format="GRanges",
GR.score="RecombinationRate", RData.location=RData.location)
## phastCons100way
gr_phast <- xGRoverlap(data=gr_Gene, format="GRanges",
GR.score="phastCons100way", RData.location=RData.location)
## phyloP100way
gr_phylo <- xGRoverlap(data=gr_Gene, format="GRanges",
GR.score="phyloP100way", RData.location=RData.location)

## End(Not run)
```

`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 = "http://galahad.well.ox.ac.uk/bigdata")
```

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 XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/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)
```

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

Description

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

Usage

```
xGRscores(data, significance.threshold = 0.05, score.cap = 10,
verbose = T, RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

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

Value

a data frame with following columns:

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

Note

None

See Also

xRDataLoader

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

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

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

## End(Not run)
```

---

xGRsep *Function to obtain separator index.*

Description

xGRsep is supposed to obtain separator index.

Usage

```
xGRsep(data)
```

Arguments

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

Value

a vector for separator index

See Also

xGRsep

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

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

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

# c) get separator index
vec_sep <- xGRsep(data)

## End(Not run)
```

---

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

Description

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

Usage

```
xGRsort(data)
```

Arguments

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

Value

index

See Also

xGRsort

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

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

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

## End(Not run)
```

---

xGRtrack *Function to visualise genes within a genomic region using track plot*

Description

xGRtrack is supposed to visualise genes within a genomic region using track plot. Genes in query within a genomic region are displayed on the gene model track along with nearby genes of desired window or number. If scores for genomic region are also provided, the genomic score track will be also displayed at the top.

Usage

```
xGRtrack(cse.query = NULL, gene.query = NULL, window = 1e+05,
nearby = NULL, name.scoretrack = "Genomic scores",
gene.model = c("UCSC_knownGene_model", "UCSC_knownCanonical_model"),
GR.score = c(NA, "RecombinationRate", "phastCons100way",
"phyloP100way",
"GERP"), GR.score.customised = NULL, name.customised = "Customised",
type.customised = c("point", "line"), label.size = 2,
label.col = "black", label.force = 0.05, verbose = TRUE,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

cse.query	a genomic region in query. By default it is NULL; otherwise provided as 'chrN:start-end', where N is either 1-22 or X, start (or end) is genomic positional number; for example, 'chr1:13-20'. If provided, it will overwrite the parameter 'gene.query' below
gene.query	which gene in query will be visualised. By default it is NULL
window	the maximum distance defining nearby genes around the gene in query. By default it is 1e5
nearby	the maximum number defining nearby genes around the gene in query. By default it is NULL. If not NULL, it will overwrite the parameter 'window' above
name.scoretrack	the name for the score track. By default, it is "Genomic scores"
gene.model	the genomic regions of the gene model. By default, it is 'UCSC_knownGene_model', that is, UCSC known genes (together with genomic locations) based on human genome assembly hg19. It can be 'UCSC_knownCanonical_model', 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"
GR.score	the genomic regions together with score data. By default, it is 'NA' to disable this option. Pre-built genomic score data: 'RecombinationRate' (recombintion rate, http://www.ncbi.nlm.nih.gov/pubmed/17943122)), 'phastCons100way', 'phyloP100way', 'GERP'.
GR.score.customised	the customised genomic score data. By default, it is NA to disable this option; otherwise load your customised GR object directly (with the first meta column for scores; if not provided, it will be valued at 1). If provided, it will be appended to 'GR.score' above. Also supported is the labelling by providing a meta-column called 'Label'. It can be also a list of GR objects
name.customised	the name for customised genomic score data. By default, it is "Customised"
type.customised	how to plot customised genomic score data. It can be "point" or "line"
label.size	label size
label.col	label color
label.force	the repelling force between overlapping labels
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 Tracks object.

Note

none

See Also

xGRoverlap

Examples

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

## End(Not run)

RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
## given a query gene
tks <- xGRtrack(gene.query='TNF', nearby=10,
gene.model="UCSC_knownGene_model",
GR.score=c("RecombinationRate","phastCons100way"),
RData.location=RData.location)
tks
## given a query genomic region
tks <- xGRtrack(cse.query='chr6:31497996-31584798',
gene.model="UCSC_knownGene_model",
GR.score=c("RecombinationRate","phastCons100way"),
RData.location=RData.location)

### GWAS catalog
GWAScatalog <- xRDataLoader('GWAScatalog',
RData.location=RData.location)
gwas <- xGR(GWAScatalog$cse_hg19, format="chr:start-end")
ind <- match(names(gwas), GWAScatalog$cse_hg19)
gwas$pvalue <- -log10(GWAScatalog$pvalue[ind])
tks <- xGRtrack(gene.query='TNF', nearby=10,
gene.model="UCSC_knownGene_model", GR.score="RecombinationRate",
GR.score.customised=gwas, RData.location=RData.location)
tks

##########################
## Advanced use: customised GR.score
##########################
gene.model <- xRDataLoader("UCSC_knownGene_model",
RData.location=RData.location)

### LDblock_GR
gr <- xRDataLoader("LDblock_GR", RData.location=RData.location)
maf <- gr[,'maf']
distance <- gr[,'distance']
cadd <- gr[,'cadd']
### GR.score.customised as a list of GR objects
```

```
GR.score.customised <- list(maf=maf, distance=distance, cadd=cadd)
tks <- xGRtrack(gene.query='TNF', window=1e0, gene.model=gene.model,
GR.score=NA, GR.score.customised=GR.score.customised,
type.customised='point', RData.location=RData.location)
tks

### the built-in provided as the customised
customised <-
c("RecombinationRate","phastCons100way","phyloP100way","GERP","dbSNP_GWAS")
GR.score.customised <- lapply(customised, function(x) xRDataLoader(x,
RData.location=RData.location))
tks <- xGRtrack(gene.query='TNF', nearby=10, gene.model=gene.model,
GR.score=NA, GR.score.customised=GR.score.customised,
type.customised='line', RData.location=RData.location)
tks

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

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 genes are used as background
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. By default, it is 'NA' to disable this option. Pre-built ontology and annotation data are detailed in xDefineOntology.
size.range	the minimum and maximum size of members of each term in consideration. By default, it sets to a minimum of 10 but no more than 2000
min.overlap	the minimum number of overlaps. Only those terms with members that overlap with input data at least min.overlap (3 by default) will be processed
which.distance	which terms with the distance away from the ontology root (if any) is used to restrict terms in consideration. By default, it sets to 'NULL' to consider all distances
test	the test statistic used. It can be "fisher" for using fisher's exact test, "hypergeo" for using hypergeometric test, or "binomial" for using binomial test. Fisher's

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

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

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

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

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

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

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

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

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

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

`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
- `or`: a vector containing odds ratio
- `CIl`: a vector containing lower bound confidence interval for the odds ratio
- `CIu`: a vector containing upper bound confidence interval for the odds ratio
- `cross`: a matrix of nTerm X nTerm, with an on-diagnal cell for the overlapped-members observed in an individaul term, and off-diagnal cell for the overlapped-members shared betwene two terms
- `call`: the call that produced this result

Note

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

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

See Also

xEnrichViewer, xEnricherGenes

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

# Enrichment analysis for GWAS SNPs from ImmunoBase
## a) provide input data
data.file <- "http://galahad.well.ox.ac.uk/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=10, displayBy="fc")
print(bp)

## End(Not run)
```

---

xGRviaGeneAnnoAdv *Function to conduct region-based enrichment analysis given a list of genomic region sets and a list of ontologies*

Description

xGRviaGeneAnnoAdv is supposed to conduct enrichment analysis given a list of gene sets and a list of ontologies. It is an advanced version of xGRviaGeneAnno, returning an object of the class 'ls_eTerm'.

Usage

```
xGRviaGeneAnnoAdv(list_vec, background = NULL, build.conversion = c(NA,
"hg38.to.hg19", "hg18.to.hg19"), gap.max = 0,
GR.Gene = c("UCSC_knownGene", "UCSC_knownCanonical"), ontologies = NA,
size.range = c(10, 2000), min.overlap = 3, which.distance = NULL,
```

```
test = c("hypergeo", "fisher", "binomial"),
background.annotatable.only = NULL, p.tail = c("one-tail",
"two-tails"),
p.adjust.method = c("BH", "BY", "bonferroni", "holm", "hochberg",
"hommel"),
ontology.algorithm = c("none", "pc", "elim", "lea"), elim.pvalue =
0.01,
lea.depth = 2, path.mode = c("all_paths", "shortest_paths",
"all_shortest_paths"), true.path.rule = F, verbose = T, silent = FALSE,
plot = TRUE, fdr.cutoff = 0.05, displayBy = c("zscore", "fdr",
"pvalue",
"fc", "or"), RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

list_vec	an input vector containing genomic regions. Alternatively it can be a list of vectors, representing multiple groups of genomic regions. Formatted as "chr:start-end" are genomic regions
background	a background vector containing genomic regions (formatted as "chr:start-end") as the test background. If NULL, by default all annotatable are used as background
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"
ontologies	the ontologies supported currently. By default, it is 'NA' to disable this option. Pre-built ontology and annotation data are detailed in xDefineOntology.
size.range	the minimum and maximum size of members of each term in consideration. By default, it sets to a minimum of 10 but no more than 2000
min.overlap	the minimum number of overlaps. Only those terms with members that overlap with input data at least min.overlap (3 by default) will be processed
which.distance	which terms with the distance away from the ontology root (if any) is used to restrict terms in consideration. By default, it sets to 'NULL' to consider all distances

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

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

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

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

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

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

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

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

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

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

silent	logical to indicate whether the messages will be silent completely. By default, it sets to false. If true, verbose will be forced to be false
plot	logical to indicate whether heatmap plot is drawn
fdr.cutoff	fdr cutoff used to declare the significant terms. By default, it is set to 0.05. This option only works when setting plot (see above) is TRUE
displayBy	which statistics will be used for drawing heatmap. It can be "fc" for enrichment fold change, "fdr" for adjusted p value (or FDR), "pvalue" for p value, "zscore" for enrichment z-score (by default), "or" for odds ratio. This option only works when setting plot (see above) is TRUE
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details

Value

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

- df: a data frame of n x 12, where the 12 columns are "group" (the input group names), "ontology" (input ontologies), "id" (term ID), "name" (term name), "nAnno" (number in members annotated by a term), "nOverlap" (number in overlaps), "fc" (enrichment fold changes), "zscore" (enrichment z-score), "pvalue" (nominal p value), "adjp" (adjusted p value (FDR)), "or" (odds ratio), "CIl" (lower bound confidence interval for the odds ratio), "CIu" (upper bound confidence interval for the odds ratio), "distance" (term distance or other information), "members" (members (represented as Gene Symbols) in overlaps)
- mat: NULL if the plot is not drawn; otherwise, a matrix of term names X groups with numeric values for the signficant enrichment, NA for the insignificant ones
- gp: NULL if the plot is not drawn; otherwise, a 'ggplot' object

Note

none

See Also

xRDataLoader, xGRviaGeneAnno, xEnrichViewer, xHeatmap

Examples

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

# Enrichment analysis for GWAS SNPs from ImmunoBase
## a) provide input data (bed-formatted)
data.file <- "http://galahad.well.ox.ac.uk/bigdata/ImmunoBase_GWAS.bed"
input <- read.delim(file=data.file, header=T, stringsAsFactors=F)
data <- paste0(input$chrom, ':', (input$chromStart+1), '-',
input$chromEnd)
```

```
# b) perform enrichment analysis
## overlap with gene body
ls_eTerm <- xGRviaGeneAnnoAdv(data, gap.max=0,
ontologies=c("REACTOME_ImmuneSystem","REACTOME_SignalTransduction"),
RData.location=RData.location)
ls_eTerm
## forest plot of enrichment results
gp <- xEnrichForest(ls_eTerm, top_num=10, CI.one=F)
gp

## 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 either at the base resolution, at the region resolution or at the hybrid 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"),
resolution = c("bases", "regions", "hybrid"),
background.annotatable.only = T, p.tail = c("one-tail", "two-tails"),
p.adjust.method = c("BH", "BY", "bonferroni", "holm", "hochberg",
"hommel"),
GR.annotation = NA, verbose = T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

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)

`resolution` the resolution of overlaps being tested. It can be one of "bases" at the base resolution (by default), "regions" at the region resolution, and "hybrid" at the base-region hybrid resolution (that is, data at the region resolution but annotation/background at the base resolution). If regions being analysed are SNPs themselves, then the results are the same even when choosing this parameter as either 'bases' or 'hybrid' or 'regions'

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

`p.adjust.method`

the method used to adjust p-values. It can be one of "BH", "BY", "bonferroni", "holm", "hochberg" and "hommel". The first two methods "BH" (widely used)

	and "BY" control the false discovery rate (FDR: the expected proportion of false discoveries amongst the rejected hypotheses); the last four methods "bonferroni", "holm", "hochberg" and "hommel" are designed to give strong control of the family-wise error rate (FWER). Notes: FDR is a less stringent condition than FWER
GR.annotation	the genomic regions of annotation data. By default, it is 'NA' to disable this option. Pre-built genomic annotation data are detailed in the section 'Note'. Alternatively, the user can also directly provide a customised GR object (or a list of GR objects)
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 following columns (below explanations are based on results at the 'hybrid' resolution):

- 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
- or: a vector containing odds ratio
- CIl: a vector containing lower bound confidence interval for the odds ratio
- CIu: a vector containing upper bound confidence interval for the odds ratio
- 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

Pre-built genomic annotation data are detailed in xDefineGenomicAnno.

See Also

xDefineGenomicAnno

Examples

```
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

## Not run:
# Enrichment analysis for GWAS SNPs from ImmunoBase
## a) provide input data
data.file <- "http://galahad.well.ox.ac.uk/bigdata/ImmunoBase_GWAS.bed"

## b) perform enrichment analysis using FANTOM expressed enhancers
### one-tail p-value calculation (by default)
eTerm <- xGRviaGenomicAnno(data.file, format.file="bed",
GR.annotation="FANTOM5_Enhancer_Cell", RData.location=RData.location)
### alternatively: two-tails p-value calculation (useful to identify depletions)
eTerm_2 <- xGRviaGenomicAnno(data.file, format.file="bed",
GR.annotation="FANTOM5_Enhancer_Cell", p.tail="two-tails",
RData.location=RData.location)

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

## d) barplot of enriched terms
bp <- xEnrichBarplot(eTerm, top_num='auto', displayBy="fc")
bp

## e) forest plot of enriched terms
gp <- xEnrichForest(eTerm)
gp

## f) 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)

########################################
### Advanced use: customised GR.annotation
########################################
FANTOM5_CAT_Cell <- xRDataLoader('FANTOM5_CAT_Cell',
RData.location=RData.location)
ls_gr_lncRNA <- lapply(FANTOM5_CAT_Cell, function(x)
x[grep('lncRNA',x$Category)])
ls_gr_mRNA <- lapply(FANTOM5_CAT_Cell, function(x)
x[grep('coding_mRNA',x$Category)])
GR.annotations <- c("ls_gr_lncRNA","ls_gr_mRNA","FANTOM5_CAT_Cell")
ls_df <- lapply(1:length(GR.annotations), function(i){
GR.annotation <- get(GR.annotations[i])
df <- xGRviaGenomicAnno(data.file=data.file, format.file="bed",
GR.annotation=GR.annotation, RData.location=RData.location)
df$group <- GR.annotations[i]
return(df)
```

```
})
df <- do.call(rbind, ls_df)
gp <- xEnrichHeatmap(df, fdr.cutoff=0.05, displayBy="zscore")

########################################
### Advanced use: customised EpigenomeAtlas_15Segments
########################################
info <- xRDataLoader('EpigenomeAtlas_15Segments_info',
RData.location=RData.location)
GR.annotations <- paste0('EpigenomeAtlas_15Segments_',names(info))
names(GR.annotations) <- info
ls_df <- lapply(1:length(GR.annotations), function(i){
GR.annotation <- GR.annotations[i]
message(sprintf("Analysing '%s' (%s) ...", names(GR.annotation),
as.character(Sys.time())), appendLF=T)
df <- xGRviaGenomicAnno(data.file=data.file, format.file="bed",
GR.annotation=GR.annotation, RData.location=RData.location, verbose=F)
df$group <- names(GR.annotation)
return(df)
})
df <- do.call(rbind, ls_df)
gp <- xEnrichHeatmap(df, fdr.cutoff=0.05, displayBy="fdr",
reorder="both")


## 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 overlaps 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 = NA,
```

```
parallel = TRUE, multicores = NULL, verbose = T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

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 in `xDefineGenomicAnno`. Alternatively, the user can also directly provide a customised GR object (or a list of GR objects)

`parallel` logical to indicate whether parallel computation with multicores is used. By default, it sets to true, but not necessarily does so. It will depend on whether these two packages "foreach" and "doParallel" have been installed. It can be installed via: `source("http://bioconductor.org/biocLite.R"); biocLite(c("foreach","doParallel")`
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

Pre-built genomic annotation data are detailed in xDefineGenomicAnno.

See Also

xDefineGenomicAnno

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

# Enrichment analysis for GWAS SNPs from ImmunoBase
## a) provide input data
data.file <- "http://galahad.well.ox.ac.uk/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) barplot of enriched terms
bp <- xEnrichBarplot(eTerm, top_num='auto', displayBy="fdr")
bp

## e) 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)
```

xGScore *Function to extract scores given a list of genomic regions*

Description

xGScore is supposed to extract scores given a list of genomic regions. Scores for genomic regions/variants can be constraint/conservation or impact/pathogenicity. It returns a GR object.

Usage

```
xGScore(data, format = c("chr:start-end", "data.frame", "bed",
"GRanges"),
build.conversion = c(NA, "hg38.to.hg19", "hg18.to.hg19"),
GS.annotation = c("fitCons", "phastCons", "phyloP", "mcap", "cadd"),
scoring.scheme = c("mean", "median", "max", "min", "sum"), verbose = T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

data	input genomic regions (GR). If formatted as "chr:start-end" (see the next parameter 'format' below), GR should be provided as a vector in the format of 'chrN:start-end', where N is either 1-22 or X, start (or end) is genomic positional number; for example, 'chr1:13-20'. If formatted as a 'data.frame', the first three columns correspond to the chromosome (1st column), the starting chromosome position (2nd column), and the ending chromosome position (3rd column). If the format is indicated as 'bed' (browser extensible data), the same as 'data.frame' format but the position is 0-based offset from chromomose position. If the genomic regions provided are not ranged but only the single position, the ending chromosome position (3rd column) is allowed not to be provided. The data could also be an object of 'GRanges' (in this case, formatted as 'GRanges')
format	the format of the input data. It can be one of "data.frame", "chr:start-end", "bed" or "GRanges"
build.conversion	the conversion from one genome build to another. The conversions supported are "hg38.to.hg19" and "hg18.to.hg19". By default it is NA (no need to do so)
GS.annotation	which genomic scores (GS) annotaions used. It can be 'fitCons' (the probability of fitness consequences for point mutations; http://www.ncbi.nlm.nih.gov/pubmed/25599402), 'phastCons' (the probability that each nucleotide belongs to a conserved element/negative selection [0,1]), 'phyloP' (conservation at individual sites representing -log p-values under a null hypothesis of neutral evolution, positive scores for conservation and negative scores for acceleration), 'mcap' (eliminating a majority of variants with uncertain significance in clinical exomes at high sensitivity: http://www.ncbi.nlm.nih.gov/pubmed/27776117), and 'cadd' (combined annotation dependent depletion for estimating relative levels of pathogenicity of potential human variants: http://www.ncbi.nlm.nih.gov/pubmed/24487276)

scoring.scheme	the method used to calculate scores spanning a set of GR. It can be one of "mean", "median", "max", "min" and "sum"
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 GenomicRanges object

See Also

xRDataLoader

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

# a) provide the genomic regions
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
## get lead SNPs reported in AS GWAS
data <- ImmunoBase$AS$variant

# b) extract fitness consequence score
gr <- xGScore(data=data, format="GRanges", GS.annotation="fitCons",
scoring.scheme="mean", RData.location=RData.location)

## End(Not run)
```

---

xGScoreAdv *Function to calculate per base scores given a list of genomic regions in terms of overlaps with genomic annotations*

Description

xGScoreAdv is supposed to calculate per base scores for an input list of genomic regions (genome build 19), using genomic annotations (eg genomic segments, active chromatin, transcription factor binding sites/motifs, conserved sites). The per base scores are calculated for overlaps with each genomic annotation. Scores for genomic regions/variants can be constraint/conservation or impact/pathogenicity.

Usage

```
xGScoreAdv(data, format = c("data.frame", "bed", "chr:start-end",
"GRanges"),
build.conversion = c(NA, "hg38.to.hg19", "hg18.to.hg19"),
GS.annotation = c("fitCons", "phastCons", "phyloP", "mcap", "cadd"),
GR.annotation = NA, details = F, verbose = T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

data	input genomic regions (GR). If formatted as "chr:start-end" (see the next parameter 'format' below), GR should be provided as a vector in the format of 'chrN:start-end', where N is either 1-22 or X, start (or end) is genomic positional number; for example, 'chr1:13-20'. If formatted as a 'data.frame', the first three columns correspond to the chromosome (1st column), the starting chromosome position (2nd column), and the ending chromosome position (3rd column). If the format is indicated as 'bed' (browser extensible data), the same as 'data.frame' format but the position is 0-based offset from chromomose position. If the genomic regions provided are not ranged but only the single position, the ending chromosome position (3rd column) is allowed not to be provided. The data could also be an object of 'GRanges' (in this case, formatted as 'GRanges')
format	the format of the input data. It can be one of "data.frame", "chr:start-end", "bed" or "GRanges"
build.conversion	the conversion from one genome build to another. The conversions supported are "hg38.to.hg19" and "hg18.to.hg19". By default it is NA (no need to do so)
GS.annotation	which genomic scores (GS) annotaions used. It can be 'fitCons' (the probability of fitness consequences for point mutations; http://www.ncbi.nlm.nih.gov/pubmed/25599402), 'phastCons' (the probability that each nucleotide belongs to a conserved element/negative selection [0,1]), 'phyloP' (conservation at individual sites representing -log p-values under a null hypothesis of neutral evolution, positive scores for conservation and negative scores for acceleration), 'mcap' (eliminating a majority of variants with uncertain significance in clinical exomes at high sensitivity: http://www.ncbi.nlm.nih.gov/pubmed/27776117), and 'cadd' (combined annotation dependent depletion for estimating relative levels of pathogenicity of potential human variants: http://www.ncbi.nlm.nih.gov/pubmed/24487276)
GR.annotation	the genomic regions of annotation data. By default, it is 'NA' to disable this option. Pre-built genomic annotation data are detailed in xDefineGenomicAnno. Alternatively, the user can also directly provide a customised GR object (or a list of GR objects)
details	logical to indicate whether the detailed information (ie ratio) is returned. By default, it sets to false for no inclusion
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 6 columns:

- name: the annotation name
- o_nBase: the number of bases overlapped between input regions and annotation regions
- o_GS: the per base genomic scores for overlaps between input regions and annotation regions
- a_nBase: the number of bases covered by that annotation; optional, it is only appended when "details" is true
- a_GS: the per base genomic scores for that annotation; optional, it is only appended when "details" is true
- ratio: ratio of o_GS divided by a_GS; optional, it is only appended when "details" is true

Note

Pre-built genomic annotation data are detailed in xDefineGenomicAnno.

See Also

xGScore

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

# a) provide the genomic regions
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
## get lead SNPs reported in AS GWAS
data <- ImmunoBase$AS$variant

# b) in terms of overlaps with genomic segments (Primary monocytes from peripheral blood)
## fitness consequence score
res_df <- xGScoreAdv(data=data, format="GRanges",
GS.annotation="fitCons",
GR.annotation="EpigenomeAtlas_15Segments_E029",
RData.location=RData.location)
## phastCons conservation score
res_df <- xGScoreAdv(data=data, format="GRanges",
GS.annotation="phastCons",
GR.annotation="EpigenomeAtlas_15Segments_E029",
RData.location=RData.location)

# c) in terms of overlaps with genic annotations
## phyloP conservation score
res_df <- xGScoreAdv(data=data, format="GRanges",
GS.annotation="phyloP", GR.annotation="Genic_anno",
```

```
RData.location=RData.location)

## End(Not run)
```

---

xHeatmap *Function to draw heatmap using ggplot2*

Description

xHeatmap is supposed to draw heatmap using ggplot2.

Usage

```
xHeatmap(data, reorder = c("none", "row", "col", "both"),
colormap = "spectral", ncolors = 64, zlim = NULL, barwidth = 0.3,
barheight = NULL, nbin = 64, legend.title = "", x.rotate = 60,
x.text.size = 6, y.text.size = 6, legend.text.size = 4,
legend.title.size = 6, shape = 19, size = 2, plot.margin = unit(c(5.5,
5.5, 5.5, 5.5), "pt"), font.family = "sans", na.color = "grey80",
data.label = NULL, label.size = 1, label.color = "black", ...)
```

Arguments

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

x.rotate	the angle to rotate the x tick labelings. By default, it is 60
x.text.size	the text size of the x tick labelings. By default, it is 6
y.text.size	the text size of the y tick labelings. By default, it is 6
legend.text.size	the text size of the legend tick labelings. By default, it is 5
legend.title.size	the text size of the legend titles. By default, it is 6
shape	the number specifying the shape. By default, it is 19
size	the number specifying the shape size. By default, it is 2
plot.margin	the margin (t, r, b, l) around plot. By default, it is unit(c(5.5,5.5,5.5,5.5),"pt")
font.family	the font family for texts
na.color	the color for NAs. By default, it is 'grey80'
data.label	a data frame/matrix used for the labelling
label.size	the label size
label.color	the label color
...	additional graphic parameters for supraHex::visTreeBootstrap

Value

a ggplot2 object

Note

none

See Also

xHeatmap

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
data(mtcars)
gp <- xHeatmap(mtcars, reorder="none", colormap='jet.top', x.rotate=45,
shape=19, size=3, x.text.size=8,y.text.size=8, legend.title='mtcars')
gp + theme(legend.position="bottom",legend.direction="horizontal") +
guides(color=guide_colorbar(title="mtcars",title.position="top",barwidth=5,barheight=0.3))
gp + theme(legend.position="bottom",legend.direction="horizontal") +
guides(color=guide_legend(title="mtcars",title.position="top",barwidth=5,barheight=0.3))
gp + geom_text(aes(x, y,
label=val),size=1.8,color='black',fontface='bold',na.rm=TRUE,angle=45)

## End(Not run)
```

xHeatmapAdv *Function to draw heatmap together with sidebars on rows using ggplot2*

Description

xHeatmapAdv is supposed to draw heatmap together with sidebars on rows using ggplot2.

Usage

```
xHeatmapAdv(data.main, data.meta, reorder = c("none", "row", "col",
"both"),
colormap = "spectral", ncolors = 64, zlim = NULL, barwidth = 0.3,
barheight = 4, nbin = 64, legend.title = "Main", x.rotate = 60,
x.text.size = 6, y.text.size = 6, legend.text.size = 5,
legend.title.size = 6, shape = 19, size = 2, plot.margin = unit(c(5.5,
5.5, 5.5, 5.5), "pt"), font.family = "sans", na.color = "grey80",
data.label = NULL, label.size = 1, label.color = "black",
meta.colormap = "spectral", meta.x.rotate = 75,
meta.shape.continuous = 15, meta.shape.discrete = 95, meta.size = 2,
meta.location = c("right", "left"), meta.width = 0.5, gap.width = 0.5,
legend.width = NULL, legend.direction = c("vertical", "horizontal"),
legend.nrow = NULL, verbose = TRUE, ...)
```

Arguments

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

barheight	the height of the colorbar. Default value is 'legend.key.height' or 'legend.key.size' in 'theme' or theme
nbin	the number of bins for drawing colorbar
legend.title	the title of the colorbar. By default, it is ''
x.rotate	the angle to rotate the x tick labelings. By default, it is 60
x.text.size	the text size of the x tick labelings. By default, it is 6
y.text.size	the text size of the y tick labelings. By default, it is 6
legend.text.size	the text size of the legend tick labelings. By default, it is 5
legend.title.size	the text size of the legend titles. By default, it is 6
shape	the number specifying the shape. By default, it is 19
size	the number specifying the shape size. By default, it is 2
plot.margin	the margin (t, r, b, l) around plot. By default, it is unit(c(5.5,5.5,5.5,5.5),"pt")
font.family	the font family for texts
na.color	the color for NAs. By default, it is 'grey80'
data.label	a data frame/matrix used for the labelling
label.size	the label size
label.color	the label color
meta.colormap	the colormap for metadata
meta.x.rotate	the angle to rotate the x tick labelings for the metadata. By default, it is 90
meta.shape.continuous	the number specifying the shape for continuous metadata. By default, it is 15
meta.shape.discrete	the number specifying the shape for discrete metadata. By default, it is 95
meta.size	the number specifying the shape size for metadata. By default, it is 2
meta.location	the location of metadata. It can be "right" or "left"
meta.width	the width for each column in metadata. By default, it is 0.5 (relative to each column in main data)
gap.width	the width for the gap between panels. By default, it is 0.5 (relative to each column in main data)
legend.width	the width for the legend. By default, it is NULL automatically determined (which can be used as a reference to define later for the better visualisation)
legend.direction	the direction of the legend. It can be "vertical" or "horizontal"
legend.nrow	the row number for legends. By default, it is 3 for the vertical direction of the legend; 6 for the horizontal direction of the legend
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display
...	additional graphic parameters for supraHex::visTreeBootstrap

Value

a gtable object

Note

none

See Also

xHeatmap

Examples

```
## Not run:
# Load the XGR package
library(XGR)
data(mtcars)
data.main <- mtcars[,1:6]
data.meta <- mtcars[,7:11]
gt <- xHeatmapAdv(data.main, data.meta, barwidth=0.3, barheight=2.5,
meta.location="right", legend.nrow=3, meta.width=0.4, gap.width=0.2,
legend.width=NULL)
gt <- xHeatmapAdv(data.main, data.meta, barwidth=0.3, barheight=4,
meta.location="right", legend.nrow=6, meta.width=0.4, gap.width=0.2,
legend.width=4)
dev.new(); grid.draw(gt)

## End(Not run)
```

---

xHEB *Function to visualise a graph with communities using hierarchical edge bundling*

Description

xHEB is supposed to visualise a graph with communities using hierarchical edge bundling (HEB), an effective way to visualise connections between leaves of a hierarchical/tree graph (representing the community structure). The connections are curved and follow the tree structure (in a circular layout). It returns a ggplot object.

Usage

```
xHEB(g, leave.label.size = 3, leave.label.color = "black",
leave.size = NULL, edge.tension = 0.8, edge.alpha = 1,
edge.width = 0.5, edge.palette = NULL)
```

Arguments

g	an object of class "igraph" with node attributes 'name' and 'community'
leave.label.size	the text size of the leave labelings. By default, it is 3
leave.label.color	the color of the leave labelings. By default, it is 'black'. If NULL, the label will be colored by the community
leave.size	the size of the leave nodes. By default, it is 3 if there is no node attribute 'size'
edge.tension	the bundling strength of edges. 1 for very tight bundles, 0 for no bundle (straight lines). By defaults it is 0.8
edge.alpha	the alpha of edges
edge.width	the width of edges
edge.palette	the palette defining edge color. It is correponding to the edge attribute 'weight' for the input graph (if any). By default, it is NULL: if the edge attribute 'weight' exists for the input graph, it will be 'RdPu' (RColorBrewer::display.brewer.all()); otherwise 'skyblue'

Value

a ggplot2 object

See Also

xHEB

Examples

```
# 1) generate a random bipartite graph
set.seed(123)
g <- sample_bipartite(50, 20, p=0.1)
V(g)$name <- paste0('node_',1:vcount(g))

## Not run:
# 2) obtain its community
ig <- xBigraph(g)

# 3) HEB visualisation
library(ggraph)
E(ig)$weight <- runif(ecount(ig))
gp <- xHEB(ig)

## End(Not run)
```

xLDblock *Function to obtain LD blocks*

Description

xLDblock is supposed to obtain LD blocks for a list of Lead SNPs together with the significance level.

Usage

```
xLDblock(data, include.LD = c("AFR", "AMR", "EAS", "EUR", "SAS"),
LD.customised = NULL, LD.r2 = 0.8, GR.SNP = "LDblock_GR", verbose = T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

data	a named input vector containing the significance level for nodes (dbSNP). For this named vector, the element names are dbSNP (starting with rs or in the format of 'chrN:xxx', where N is either 1-22 or X, xxx is number; for example, 'chr16:28525386'), the element values for the significance level (measured as p-value or fdr). Alternatively, it can be a matrix or data frame with two columns: 1st column for dbSNP, 2nd column for the significance level.
include.LD	additional SNPs in LD with Lead SNPs are also included. By default, it is 'NA' to disable this option. Otherwise, LD SNPs will be included based on one or more of 26 populations and 5 super populations from 1000 Genomics Project data (phase 3). The population can be one of 5 super populations ("AFR", "AMR", "EAS", "EUR", "SAS"). 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 compulsory columns: 1st column for Lead SNPs, 2nd column for LD SNPs, and 3rd for LD r2 value. The recommended columns are 'maf', 'distance' (to the nearest gene) and 'cadd'. 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 0.1 to 1
GR.SNP	the genomic regions of SNPs. By default, it is 'LDblock_GR', that is, SNPs from dbSNP (version 150) restricted to GWAS SNPs and their LD SNPs (hg19). Beyond it, the user can also directly provide a customised GR object
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 "bLD", a list with following components:

- `best`: a GR object. It has optional meta-columns 'maf', 'distance' (to the nearest gene) and 'cadd', and compulsory meta-columns 'pval', 'score' (-log10(pval)), 'upstream' (the lower boundary away from the best SNP, non-positive value), 'downstream' (the upper boundary away from the best SNP, non-negative value) and 'num' (the number of SNPs in the block)
- `block`: a GRL object, each element corresponding to a block for the best SNP with optional meta-columns 'maf', 'distance' (to the nearest gene) and 'cadd', and compulsory meta-columns 'pval', 'score' (-log10(pval)*R2, based on pval for its lead SNP), 'best' (the best SNP) and 'distance_to_best' (to the best SNP)

Note

None

See Also

xLDblock

Examples

```
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

## Not run:
# a) provide the seed SNPs with the significance info
## load ImmunoBase
data(ImmunoBase)
## get lead SNPs reported in AS GWAS and their significance info (p-values)
gr <- ImmunoBase$AS$variant
data <- GenomicRanges::mcols(gr)[,c('Variant','Pvalue')]

# b) get LD block (EUR population)
bLD <- xLDblock(data, include.LD="EUR", LD.r2=0.8,
RData.location=RData.location)

# c1) manhattan plot of the best
best <- bLD$best
best$value <- best$score
gp <- xGRmanhattan(best, top=length(best))
gp
# c2) manhattan plot of all LD block
grl_block <- bLD$block
gr_block <- BiocGenerics::unlist(grl_block,use.names=F)
gr_block$value <- gr_block$score
top.label.query <- names(gr_block)[!is.na(gr_block$pval)]
#gr_block <- gr_block[as.character(GenomicRanges::seqnames(gr_block)) %in% c('chr1','chr2')]
gp <- xGRmanhattan(gr_block, top=length(gr_block),
top.label.query=top.label.query)
```

```
# c3) karyogram plot of the best
kp <- xGRkaryogram(gr=best,cytoband=T,label=T,
RData.location=RData.location)
kp
# c4) circle plot of the best
library(ggbio)
gr_ideo <- xRDataLoader(RData.customised="hg19_ideogram",
RData.location=RData.location)$ideogram
#cp <- ggbio() + circle(kp$gr, geom="rect", color="steelblue", size=0.5)
cp <- ggbio() + circle(kp$gr, aes(x=start, y=num), geom="point",
color="steelblue", size=0.5)
cp <- cp + circle(gr_ideo, geom="ideo", fill="gray70") +
circle(gr_ideo, geom="scale", size=1.5) + circle(gr_ideo, geom="text",
aes(label=seqnames), vjust=0, size=3)
cp

# d) track plot of 1st LD block
gr_block <- bLD$block[[1]]
cnames <- c('score','maf','cadd')
ls_gr <- lapply(cnames, function(x) gr_block[,x])
names(ls_gr) <- cnames
ls_gr$score$Label <- names(gr_block)
ls_gr$score$Label[is.na(gr_block$pval)] <-''
GR.score.customised <- ls_gr
## cse.query
df_block <- as.data.frame(gr_block)
chr <- unique(df_block$seqnames)
xlim <- range(df_block$start)
cse.query <- paste0(chr,':',xlim[1],'-',xlim[2])
#cse.query <- paste0(chr,':',xlim[1]-1e4,'-',xlim[2]+1e4)
## xGRtrack
tks <- xGRtrack(cse.query=cse.query, GR.score="RecombinationRate",
GR.score.customised=GR.score.customised, RData.location=RData.location)
tks

###############
# Advanced use: get LD block (based on customised LD and SNP data)
###############
LD.customised <- xRDataLoader('LDblock_EUR',
RData.location=RData.location)
GR.SNP <- xRDataLoader('LDblock_GR', RData.location=RData.location)
bLD <- xLDblock(data, LD.customised=LD.customised, LD.r2=0.8,
GR.SNP=GR.SNP, RData.location=RData.location)

## End(Not run)
```

`xLDenricher` *Function to conduct LD-based enrichment analysis using genomic annotations via sampling*

Description

xLDenricher is supposed to conduct LD-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 overlaps against the expected overlaps which are estimated from the null distribution. The null LD block is generated via sampling from the background (for example, all GWAS SNPs or all common SNPs), respecting the maf of the best SNP and/or the distance of the best SNP to the nearest gene, restricting the same chromosome or not.

Usage

```
xLDenricher(bLD, GR.SNP = c("dbSNP_GWAS", "dbSNP_Common",
"dbSNP_Single"),
num.samples = 2000, respect = c("maf", "distance", "both"),
restrict.chr = F, preserve = c("exact", "boundary"), seed = 825,
p.adjust.method = c("BH", "BY", "bonferroni", "holm", "hochberg",
"hommel"),
GR.annotation = NA, verbose = T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

bLD	a bLD object, containing a set of blocks based on which to generate a null distribution
GR.SNP	the genomic regions of SNPs. By default, it is 'dbSNP_GWAS', that is, SNPs from dbSNP (version 150) restricted to GWAS SNPs and their LD SNPs (hg19). It can be 'dbSNP_Common', that is, Common SNPs from dbSNP (version 150) plus GWAS SNPs and their LD SNPs (hg19). Alternatively, the user can specify the customised GR object directly
num.samples	the number of samples randomly generated
respect	how to respect the properties of to-be-sampled LD blocks. It can be one of 'maf' (respecting the maf of the best SNP), 'distance' (respecting the distance of the best SNP to the nearest gene), and 'both' (respecting the maf and distance)
restrict.chr	logical to restrict to the same chromosome. By default, it sets to false
preserve	how to preserve the resulting null LD block. It can be one of 'boundary' (preserving the boundary of the LD block), and 'exact' (exactly preserving the relative SNP locations within the LD block). Notably, no huge difference for the boundary preserving when enrichment analysis invovles region-based genomic annotations, but it may make difference when genomic annatations are largely SNP-based (such as eQTLs)
seed	an integer specifying the seed
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 in xDefineGenomicAnno. Alternatively, the user can also directly provide a customised GR object (or a list of GR objects)
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 13 columns:

- name: the annotation name
- nAnno: the number of regions from annotation data
- nOverlap: the observed number of LD blocks overlapped with annotation data
- 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
- or: a vector containing odds ratio
- CIl: a vector containing lower bound confidence interval for the odds ratio
- CIu: a vector containing upper bound confidence interval for the odds ratio
- nData: the number of input LD blocks
- nExpect: the expected number of LD blocks overlapped with annotation data
- std: the standard deviation of expected number of LD blocks overlapped with annotation data

Note

Pre-built genomic annotation data are detailed in xDefineGenomicAnno.

See Also

xDefineGenomicAnno

Examples

```
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

## Not run:
# a) provide the seed SNPs with the significance info
## load ImmunoBase
```

```
data(ImmunoBase)
## get lead SNPs reported in AS GWAS and their significance info (p-values)
gr <- ImmunoBase$AS$variant
data <- GenomicRanges::mcols(gr)[,c('Variant','Pvalue')]

# b) get LD block (EUR population)
bLD <- xLDblock(data, include.LD="EUR", LD.r2=0.8,
RData.location=RData.location)

## c) perform enrichment analysis using FANTOM expressed enhancers
eTerm <- xLDenricher(bLD, GR.annotation="ReMap_Encode_mergedTFBS",
RData.location=RData.location)

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

## e) barplot of enriched terms
bp <- xEnrichBarplot(eTerm, top_num='auto', displayBy="fdr")
bp

## f) forest plot of enrichment results
gp <- xEnrichForest(eTerm, FDR.cutoff=0.01)

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

## h) compare boundary and exact
GR.SNP <- xRDataLoader("dbSNP_GWAS", RData.location=RData.location)
GR.annotation <- xRDataLoader("FANTOM5_CAT_Cell",
RData.location=RData.location)
eTerm_boundary <- xLDenricher(bLD, GR.SNP=GR.SNP,
GR.annotation=GR.annotation, num.samples=20000, preserve="boundary",
RData.location=RData.location)
eTerm_exact <- xLDenricher(bLD, GR.SNP=GR.SNP,
GR.annotation=GR.annotation, num.samples=20000, preserve="exact",
RData.location=RData.location)
ls_eTerm <- list(boundary=eTerm_boundary, exact=eTerm_exact)
### barplot
bp <- xEnrichCompare(ls_eTerm, displayBy="zscore")
### forest plot
eTerm_boundary$group <- 'boundary'
eTerm_exact$group <- 'exact'
df <- rbind(eTerm_boundary, eTerm_exact)
gp <- xEnrichForest(df, FDR.cutoff=0.01)

## End(Not run)
```

---

xLDsampling *Function to generate randomly sampled LD blocks*

Description

xLDsampling is supposed to generate randomly sampled LD blocks. A sample block has the same boundary range as the observed, and can respect maf of the best SNP, and/or distance of the best SNP to the nearest gene. Also it can be restricted to the same chromosome. For each null LD block, it can preserve the boundary only or exactly preserve the relative SNP locations. It returns a GRL object.

Usage

```
xLDsampling(bLD, GR.SNP = c("dbSNP_GWAS", "dbSNP_Common",
"dbSNP_Single"),
num.samples = 2000, respect = c("maf", "distance", "both"),
restrict.chr = F, preserve = c("boundary", "exact"), seed = 825,
verbose = T, RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

bLD	a bLD object, containing a set of blocks based on which to generate a null distribution
GR.SNP	the genomic regions of SNPs. By default, it is 'dbSNP_GWAS', that is, SNPs from dbSNP (version 150) restricted to GWAS SNPs and their LD SNPs (hg19). It can be 'dbSNP_Common', that is, Common SNPs from dbSNP (version 150) plus GWAS SNPs and their LD SNPs (hg19). Alternatively, the user can specify the customised GR object directly
num.samples	the number of samples randomly generated
respect	how to respect the properties of to-be-sampled LD blocks. It can be one of 'maf' (respecting the maf of the best SNP), 'distance' (respecting the distance of the best SNP to the nearest gene), and 'both' (respecting the maf and distance)
restrict.chr	logical to restrict to the same chromosome. By default, it sets to false
preserve	how to preserve the resulting null LD block. It can be one of 'boundary' (preserving the boundary of the LD block), and 'exact' (exactly preserving the relative SNP locations within the LD block). Notably, no huge difference for the boundary preserving when enrichment analysis invovles region-based genomic annotations, but it may make difference when genomic annatations are largely SNP-based (such as eQTLs)
seed	an integer specifying the seed
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 GRL object, each containing an GR oject storing an instance of sampled blocks (with a meta-column 'best' for the identity, and a meta-column 'B' for the instance sequence).

See Also

xLDsampling

Examples

```
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

## Not run:
# a) provide the seed SNPs with the significance info
## load ImmunoBase
data(ImmunoBase)
## get lead SNPs reported in AS GWAS and their significance info (p-values)
gr <- ImmunoBase$AS$variant
data <- GenomicRanges::mcols(gr)[,c('Variant','Pvalue')]

# b) get LD block (EUR population)
bLD <- xLDblock(data, include.LD="EUR", LD.r2=0.8,
RData.location=RData.location)

# c) generate random samples as a GRL object
grl <- xLDsampling(bLD, GR.SNP="dbSNP_GWAS", num.samples=2000,
RData.location=RData.location)

##########################
## Advanced use: customised GR.SNP
##########################
GR.SNP <- xRDataLoader("dbSNP_GWAS", RData.location=RData.location)
grl <- xLDsampling(bLD, GR.SNP=GR.SNP, respect="both", restrict.chr=T,
preserve="exact", RData.location=RData.location)

## 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 = "http://galahad.well.ox.ac.uk/bigdata")
```

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 XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/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)
```

---

xOBOcode *Function to create codes annotating nodes in an igraph object*

Description

xOBOcode is supposed to create codes annotating nodes in an igraph object. It returns two ggplot2 objects, one for visualing the network with nodes lablelled by codes, the other for listing code meaning in a table

Usage

```
xOBOcode(g, node.level = "term_distance", node.level.value = 2,
node.label.size = 2, node.label.color = "darkblue",
node.label.alpha = 0.8, node.label.padding = 0, node.label.arrow =
0.01,
node.label.force = 0, node.shape = 19, node.xcoord = NULL,
node.ycoord = NULL, node.color = NULL, node.color.title = NULL,
colormap = "grey-grey", ncolors = 64, zlim = NULL,
node.size.range = 4, title = "", edge.size = 0.5,
edge.color = "black", edge.color.alpha = 0.4, edge.curve = 0.1,
edge.arrow = 2, edge.arrow.gap = 0.02, node.table = "term_name",
node.table.wrap = 50, table.base.size = 7, table.row.space = 2,
table.nrow = 55, table.ncol = NULL, root.code = "RT")
```

Arguments

g	an object of class "igraph"
node.level	a character specifying which node attribute defining the node level. By default, it is 'term_distance'
node.level.value	a positive integer specifying the level value as major branches. By default, it is 2
node.label.size	a character specifying which node attribute used for node label size
node.label.color	a character specifying which node attribute used for the node label color
node.label.alpha	the 0-1 value specifying transparency of node labelling
node.label.padding	the padding around the labeled node
node.label.arrow	the arrow pointing to the labeled node

node.label.force
: the repelling force between overlapping labels

node.shape
: an integer specifying node shape

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

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

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

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

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

ncolors
: the number of colors specified over the colormap

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

node.size.range
: the range of actual node size

title
: a character specifying the title for the plot

edge.size
: a numeric value specifying the edge size. By default, it is 0.5

edge.color
: a character specifying which edge attribute defining the the edge colors

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

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

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

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

node.table
: a character specifying which node attribute for coding. By default, it is 'term_name'

node.table.wrap
: a positive integer specifying wrap width of coded node labelling

table.base.size
: a positive integer specifying font size in the table

table.row.space
: a positive numeric value specifying amplying horizental space for a row with wrapped text

table.nrow
: a positive integer specifying the number of rows in the table

table.ncol
: NULL or a positive integer specifying the number of columns per page. If NULL, it will be 3 or less

root.code
: a character specifying the root code. By default, it is 'RT'

Value

a list with 3 components, two ggplot objects (code and table) and an igraph object (ig appended with node attributes 'node.code' and 'node.table')

Note

none

See Also

xGGnetwork

Examples

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

# load REACTOME
# 1a) restricted to Immune System ('R-HSA-168256') or Signal Transduction ('R-HSA-162582')
g <- xRDataLoader(RData.customised='ig.REACTOME',
RData.location=RData.location)
neighs.out <- igraph::neighborhood(g, order=vcount(g),
nodes="R-HSA-168256", mode="out")
vids <- V(g)[unique(unlist(neighs.out))]$name
ig <- igraph::induced.subgraph(g, vids=vids)

# 1b) visualise the graph with nodes coded
ls_gp <- xOBOcode(g=ig, node.level='term_distance', node.level.value=2,
node.shape=19, node.size.range=4, edge.color.alpha=0.2)
pdf('xOBOcode.pdf', useDingbats=FALSE, width=8, height=8)
print(ls_gp$code + coord_equal(ratio=1))
print(ls_gp$table)
dev.off()

# 1c) visualise the graph with nodes coded and colored by information content (IC)
V(ig)$IC <- -1*log10(V(ig)$nAnno/max(V(ig)$nAnno))
ls_gp <- xOBOcode(g=ig, node.level='term_distance', node.level.value=2,
node.shape=19, node.size.range=4, node.color='IC',
node.color.title='IC', colormap='white-cyan-darkcyan')

V(ig)$term_anno <- log10(V(ig)$nAnno)
ls_gp <- xOBOcode(g=ig, node.level='term_distance', node.level.value=2,
node.shape=19, node.size.range=4, node.color='term_anno',
node.color.title='# genes\n(log10)', colormap='white-cyan-darkcyan',
zlim=c(1,4))


# load EF (annotating GWAS reported genes)
# 2a) restricted to disease ('EFO:0000408') and annotation (>=10)
# 2a) restricted to immune system disease ('EFO:0000540') and annotation (>=10)
```

```
g <- xRDataLoader(RData.customised='ig.EF',
RData.location=RData.location)
neighs.out <- igraph::neighborhood(g, order=vcount(g),
nodes="EFO:0000540", mode="out")
nodeClan <- V(g)[unique(unlist(neighs.out))]$name
anno <- xRDataLoader(RData.customised='org.Hs.egEF',
RData.location=RData.location)
vec <- sapply(anno$gs, length)
nodeAnno <- names(vec[vec>=10])
neighs.in <- igraph::neighborhood(g, order=vcount(g), nodes=nodeAnno,
mode="in")
nodeAnno <- V(g)[unique(unlist(neighs.in))]$name
vids <- intersect(nodeClan, nodeAnno)
ig <- igraph::induced.subgraph(g, vids=vids)
V(ig)$anno <- anno$gs[V(ig)$name]
# 2b) visualise the graph with nodes coded
ls_gp <- xOBOcode(g=ig, node.level='term_distance', node.level.value=4,
node.shape=19, node.size.range=4, edge.color.alpha=0.2)
pdf('xOBOcode.pdf', useDingbats=FALSE, width=8, height=8)
print(ls_gp$code + coord_equal(ratio=1))
print(ls_gp$table)
dev.off()
# 2c) ## GWAS genes for immune system disease ('EFO:0000540')
anno <- xRDataLoader(RData.customised='org.Hs.egEF',
RData.location=RData.location)
genes <- anno$gs[['EFO:0000540']]
# 2d) ## GWAS SNPs for immune system disease ('EFO:0000540')
annotation <- xRDataLoader(RData.customised='GWAS2EF',
RData.location=RData.location)
dag <- xDAGpropagate(g, annotation, path.mode="all_paths",
propagation="min")
snps <- unlist(V(dag)[V(dag)$name=='EFO:0000540']$anno)
# 2e) ## ChEMBL targets for immune system disease ('EFO:0000540')
annotation <- xRDataLoader(RData.customised='Target2EF',
RData.location=RData.location)
dag <- xDAGpropagate(g, annotation, path.mode="all_paths",
propagation="max")
targets <- unlist(V(dag)[V(dag)$name=='EFO:0000540']$anno)


# load GOBP
# 3a) restricted to immune system process ('GO:0002376') and annotation (>=10)
g <- xRDataLoader(RData.customised='ig.GOBP',
RData.location=RData.location)
neighs.out <- igraph::neighborhood(g, order=vcount(g),
nodes="GO:0002376", mode="out")
nodeClan <- V(g)[unique(unlist(neighs.out))]$name
anno <- xRDataLoader(RData.customised='org.Hs.egGOBP',
RData.location=RData.location)
vec <- sapply(anno$gs, length)
nodeAnno <- names(vec[vec>=10])
neighs.in <- igraph::neighborhood(g, order=vcount(g), nodes=nodeAnno,
mode="in")
```

```
nodeAnno <- V(g)[unique(unlist(neighs.in))]$name
vids <- intersect(nodeClan, nodeAnno)
ig <- igraph::induced.subgraph(g, vids=vids)
V(ig)$anno <- anno$gs[V(ig)$name]
# 3b) visualise the graph with nodes coded
ls_gp <- xOBOcode(g=ig, node.level='term_distance', node.level.value=1,
node.shape=19, node.size.range=4, edge.color.alpha=0.2)
pdf('xOBOcode.pdf', useDingbats=FALSE, width=8, height=8)
print(ls_gp$code + coord_equal(ratio=1))
print(ls_gp$table)
dev.off()


# load GOMF
# 4a) restricted to molecular function ('GO:0003674') and annotation (>=50)
g <- xRDataLoader(RData.customised='ig.GOMF',
RData.location=RData.location)
neighs.out <- igraph::neighborhood(g, order=vcount(g),
nodes="GO:0003674", mode="out")
nodeClan <- V(g)[unique(unlist(neighs.out))]$name
anno <- xRDataLoader(RData.customised='org.Hs.egGOMF',
RData.location=RData.location)
vec <- sapply(anno$gs, length)
nodeAnno <- names(vec[vec>=50])
neighs.in <- igraph::neighborhood(g, order=vcount(g), nodes=nodeAnno,
mode="in")
nodeAnno <- V(g)[unique(unlist(neighs.in))]$name
vids <- intersect(nodeClan, nodeAnno)
ig <- igraph::induced.subgraph(g, vids=vids)
V(ig)$anno <- anno$gs[V(ig)$name]
# 4b) visualise the graph with nodes coded
ls_gp <- xOBOcode(g=ig, node.level='term_distance', node.level.value=1,
node.shape=19, node.size.range=4, edge.color.alpha=0.2)
pdf('xOBOcode.pdf', useDingbats=FALSE, width=8, height=8)
print(ls_gp$code + coord_equal(ratio=1))
print(ls_gp$table)
dev.off()


# load HPPA
# 5a) restricted to Abnormality of the immune system ('HP:0002715') and annotation (>=50)
g <- xRDataLoader(RData.customised='ig.HPPA',
RData.location=RData.location)
neighs.out <- igraph::neighborhood(g, order=vcount(g),
nodes="HP:0002715", mode="out")
nodeClan <- V(g)[unique(unlist(neighs.out))]$name
anno <- xRDataLoader(RData.customised='org.Hs.egHPPA',
RData.location=RData.location)
vec <- sapply(anno$gs, length)
nodeAnno <- names(vec[vec>=50])
neighs.in <- igraph::neighborhood(g, order=vcount(g), nodes=nodeAnno,
mode="in")
nodeAnno <- V(g)[unique(unlist(neighs.in))]$name
```

```
vids <- intersect(nodeClan, nodeAnno)
ig <- igraph::induced.subgraph(g, vids=vids)
V(ig)$anno <- anno$gs[V(ig)$name]
# 5b) visualise the graph with nodes coded
ls_gp <- xOBOcode(g=ig, node.level='term_distance', node.level.value=1,
node.shape=19, node.size.range=4, edge.color.alpha=0.2)
pdf('xOBOcode.pdf', useDingbats=FALSE, width=8, height=8)
print(ls_gp$code + coord_equal(ratio=1))
print(ls_gp$table)
dev.off()


# load DO
# 6a) restricted to immune system disease ('DOID:2914') and annotation (>=10)
g <- xRDataLoader(RData.customised='ig.DO',
RData.location=RData.location)
neighs.out <- igraph::neighborhood(g, order=vcount(g),
nodes="DOID:2914", mode="out")
nodeClan <- V(g)[unique(unlist(neighs.out))]$name
anno <- xRDataLoader(RData.customised='org.Hs.egDO',
RData.location=RData.location)
vec <- sapply(anno$gs, length)
nodeAnno <- names(vec[vec>=10])
neighs.in <- igraph::neighborhood(g, order=vcount(g), nodes=nodeAnno,
mode="in")
nodeAnno <- V(g)[unique(unlist(neighs.in))]$name
vids <- intersect(nodeClan, nodeAnno)
ig <- igraph::induced.subgraph(g, vids=vids)
V(ig)$anno <- anno$gs[V(ig)$name]
# 6b) visualise the graph with nodes coded
ls_gp <- xOBOcode(g=ig, node.level='term_distance', node.level.value=2,
node.shape=19, node.size.range=4, edge.color.alpha=0.2)
pdf('xOBOcode.pdf', useDingbats=FALSE, width=8, height=8)
print(ls_gp$code + coord_equal(ratio=1))
print(ls_gp$table)
dev.off()


# load MP
# 7a) restricted to immune system phenotype ('MP:0005387') and annotation (>=50)
# 7a) restricted to abnormal immune system physiology ('MP:0001790') and annotation (>=50)
g <- xRDataLoader(RData.customised='ig.MP',
RData.location=RData.location)
neighs.out <- igraph::neighborhood(g, order=vcount(g),
nodes="MP:0001790", mode="out")
nodeClan <- V(g)[unique(unlist(neighs.out))]$name
anno <- xRDataLoader(RData.customised='org.Hs.egMP',
RData.location=RData.location)
vec <- sapply(anno$gs, length)
nodeAnno <- names(vec[vec>=50])
neighs.in <- igraph::neighborhood(g, order=vcount(g), nodes=nodeAnno,
mode="in")
nodeAnno <- V(g)[unique(unlist(neighs.in))]$name
```

```
vids <- intersect(nodeClan, nodeAnno)
ig <- igraph::induced.subgraph(g, vids=vids)
V(ig)$anno <- anno$gs[V(ig)$name]
# 7b) visualise the graph with nodes coded
ls_gp <- xOBOcode(g=ig, node.level='term_distance', node.level.value=3,
node.shape=19, node.size.range=4, edge.color.alpha=0.2)
pdf('xOBOcode.pdf', useDingbats=FALSE, width=8, height=8)
print(ls_gp$code + coord_equal(ratio=1))
print(ls_gp$table)
dev.off()

## End(Not run)
```

`xPCHiCplot` *Function to visualise promoter capture HiC data using different network layouts*

Description

`xPCHiCplot` is supposed to visualise promoter capture HiC data using different network layouts.

Usage

```
xPCHiCplot(g, node.info = c("smart", "none", "GR", "GR_SNP",
"GR_SNP_target",
"SNP_target"), node.colors = c("skyblue", "pink1"), nodes.query = NULL,
newpage = TRUE, signature = TRUE, glayout = layout_with_kk,
vertex.frame.color = NA, vertex.size = NULL, vertex.color = NULL,
vertex.shape = "sphere", vertex.label = NULL, vertex.label.cex = NULL,
vertex.label.font = 2, vertex.label.dist = 0.3,
vertex.label.color = "black", edge.arrow.size = 0.5, edge.width = NULL,
edge.color = "grey", ...)
```

Arguments

`g`	an object of both classes "igraph" and "PCHiC" (part of the results from `xDefineHIC`)
`node.info`	tells the information used to label nodes. It can be one of "none" for no node labeling, "GR" for only using genomic regions (GR), "GR_SNP" for using GR and SNP (if any), "GR_SNP_target" for using GR and SNP (if any) and target genes (if any), "SNP_target" for using SNP (if any) and target genes (if any), and "smart" (by default) for only using GR if both SNP and target genes are not available (otherwise GR will be hidden)
`node.colors`	colors used to flag which nodes contain SNP or not. By default, a node harboring an SNP will be colored in 'skyblue' and the node without an SNP in 'pink'
`nodes.query`	nodes in query for which edges attached to them will be displayed. By default, it sets to NULL meaning no such restriction

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

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

`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

`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"

`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.font` the font of vertex labels. It is interpreted the same way as the the 'font' graphical parameter: 1 is plain text, 2 is bold face, 3 is italic, 4 is bold and italic and 5 specifies the symbol font.

`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 0.5.

`edge.width` the width of the directed edge. If NULL, the width edge is proportional to CHiCAGO scores (quantifying the strength of physical interactions).

`edge.color` the color of the directed edge. The default value is 'grey'.

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

Value

an igraph object

Note

- `edge arrow`: interactions are represented as a direct graph (bait-prey)
- `edge thickness`: the thickness in an edge is proportional to the interaction strength
- `node color`: a node is colored in pink if it harbors SNPs in query; otherwise skyblue
- `node label`: a node is labelled with three pieces of information (if any): genomic regions, SNPs in query, genes associated (marked by an @ icon)

See Also

xDefineHIC

Examples

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

## End(Not run)

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

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

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

# d) control node labelling info
xPCHiCplot(g, node.info="GR_SNP_target")
xPCHiCplot(g, node.info="GR_SNP")
xPCHiCplot(g, node.info="SNP_target")
xPCHiCplot(g, node.info='SNP_target', vertex.label.cex=0.5)

## End(Not run)
```

xPolarBar *Function to visualise a data frame using a polar barplot*

Description

xPolarBar is supposed to visualise a data frame using a polar dotplot. It returns an object of class "ggplot".

Usage

```
xPolarBar(df, colormap = "spectral", size.name = 10, size.value = 3,
parallel = FALSE, font.family = "sans", signature = TRUE)
```

Arguments

df	a data frame with two columns ('name' and 'value')
colormap	either NULL or color names ('spectral' by default) for bars according to the name column
size.name	an integer specifying the text size for the name column. By default, it sets to 10
size.value	an integer specifying the text size for the value column. By default, it sets to 3
parallel	logical to indicate whether the label is parallel to polar coordinate. By default, it sets FALSE
font.family	the font family for texts
signature	logical to indicate whether the signature is assigned to the plot caption. By default, it sets TRUE showing which function is used to draw this graph

Value

an object of class "ggplot"

Note

none

See Also

xPolarBar

Examples

```
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata/"

## Not run:
# a) provide the seed nodes/genes with the weight info
```

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

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

# c) do correlation
gp <- xPolarBar(df[1:20,], parallel=TRUE)
gp

## End(Not run)
```

---

xPolarDot *Function to visualise a data frame using a polar dotplot*

Description

xPolarDot is supposed to visualise a data frame using a polar dotplot. It returns an object of class "ggplot".

Usage

```
xPolarDot(df, colormap = "blue-yellow-red", shape = 19, size = 2,
parallel = FALSE, font.family = "sans", signature = TRUE)
```

Arguments

df	a data frame with two columns ('name' and 'value')
colormap	either NULL or color names ('blue-yellow-red' by default) for points according to the value column
shape	an integer specifying point shape
size	an integer specifying the point size. By default, it sets to 2
parallel	logical to indicate whether the label is parallel to polar coordinate. By default, it sets FALSE
font.family	the font family for texts
signature	logical to indicate whether the signature is assigned to the plot caption. By default, it sets TRUE showing which function is used to draw this graph

Value

an object of class "ggplot"

Note

none

See Also

xPolarDot

Examples

```
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata/"

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

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

# c) do correlation
gp <- xPolarDot(df[1:50,])
gp

## 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.egPfam", "org.Hs.string", "org.Hs.PCommons_DN",
"org.Hs.PCommons_UN"),
RData.customised = NULL, verbose = T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

RData	which built-in RData to load. It can be one of "GWAS2EF", "GWAS_LD", "IlluminaHumanHT", "IlluminaOmniExpress", "ig.DO", "ig.EF", "ig.GOBP", "ig.GOCC", "ig.GOMF", "ig.HPCM", "ig.HPMA", "ig.HPMI", "ig.HPPA", "ig.MP", "org.Hs.eg", "org.Hs.egDGIdb", "org.Hs.egDO", "org.Hs.egGOBP", "org.Hs.egGOCC", "org.Hs.egGOMF", "org.Hs.egHPCM", "org.Hs.egHPMA", "org.Hs.egHPMI", "org.Hs.egHPPA", "org.Hs.egMP", "org.Hs.egMsigdbC1", "org.Hs.egMsigdbC2BIOCARTA", "org.Hs.egMsigdbC2CGP", "org.Hs.egMsigdbC2CPall", "org.Hs.egMsigdbC2CP", "org.Hs.egMsigdbC2KEGG", "org.Hs.egMsigdbC2REACTOME", "org.Hs.egMsigdbC3MIR", "org.Hs.egMsigdbC3TFT", "org.Hs.egMsigdbC4CGN", "org.Hs.egMsigdbC4CM", "org.Hs.egMsigdbC5BP", "org.Hs.egMsigdbC5CC", "org.Hs.egMsigdbC5MF", "org.Hs.egMsigdbC6", "org.Hs.egMsigdbC7", "org.Hs.egMsigdbH", "org.Hs.egPS", "org.Hs.egSF", "org.Hs.egPfam", "org.Hs.string", "org.Hs.PCommons_DN", "org.Hs.PCommons_UN", "org.Hs.egGTExV4", "org.Hs.egGTExV6"
RData.customised	a file name for RData-formatted file. By default, it is NULL. It is designed when the user wants to import customised RData that are not listed in the above argument 'RData'. However, this argument can be always used even for those RData that are listed in the argument 'RData'
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to TRUE for display
RData.location	the characters to tell the location of built-in RData files. By default, it remotely locates at http://galahad.well.ox.ac.uk/bigdata; the development version locates at http://galahad.well.ox.ac.uk/bigdata. For the user equipped with fast internet connection, this option can be just left as default. But it is always advisable to download these files locally. Especially when the user needs to run this function many times, there is no need to ask the function to remotely download every time (also it will unnecessarily increase the runtime). For examples, these files (as a whole or part of them) can be first downloaded into your current working directory, and then set this option as: $RData.location = "."$. Surely, the location can be anywhere as long as the user provides the correct path pointing to (otherwise, the script will have to remotely download each time)

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

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

## 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)
```

xRegress *Function to regress data according to principle components (PCs)*

Description

xRegress is supposed to regress data according to principle components (PCs).

Usage

```
xRegress(data, center = TRUE, scale = TRUE, which.PCs = NULL)
```

Arguments

data	a data matrix/frame with, for exampe, genes in rows and samples in columns
center	logical to indicate whether the input data columns should be shifted to be zero centered when calculating PCs
scale	logical to indicate whether the input data columns should have unit variance when calculating PCs
which.PCs	a vector specifying which PCs are used for being regressed out. If NULL (by default), no gression is done

Value

a list with three componets:

- regressed: the regressed data with the same dimension as the input data
- PCs: a data matrix of PCs X samples
- Ss: a vector storing the square roots of the eigenvalues

Note

none

See Also

xRegress

Examples

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

## End(Not run)

## Not run:
data(Fang)
```

```
ls_res <- xRegress(Fang, which.PCs=1)
gp <- xHeatmap(ls_res$PCs)
gp

## End(Not run)
```

---

xRepurpose *Function to obtain repurposing matrix*

Description

xRepurpose is supposed to obtain repurposing matrix given a query list of genes. It returns an object of the class 'DR'.

Usage

```
xRepurpose(data, phase.min = 3, target.max = 5, plot = TRUE,
verbose = T, DTT = c("ChEMBL_v24", "ChEMBL_v23"), restricted = NULL,
excluded = NULL, RData.location =
"http://galahad.well.ox.ac.uk/bigdata",
...)
```

Arguments

data	an input vector containing gene symbols
phase.min	the minumum phase of drugs allowed. By default it is 3 defining target genes of drugs reaching development phase 3 and above
target.max	the maximum number of targets per drug allowed. By default it is 5. It is used to define non-promoscuous drug target genes
plot	logical to indicate whether heatmap plot is drawn
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display
DTT	the drug therapeutic targets. It can be "ChEMBL_v24" for the version 24 (by default), and the version 23. Note: you can also load your customised object directly with columns ('target_number','efo_term','phase','pref_name_drug','Symbol')
restricted	the disease areas restricted to. By default it is NULL
excluded	the disease areas that are excluded. By default it is NULL
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details
...	additional graphic parameters for xHeatmap

Value

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

- df: a data frame of n x 5, where the 5 columns are "Target", "Disease", "Phase", "Drug", "Drug_index"
- index: a data frame of n x 2, where the 2 columns are "Drug_index", "Drug"
- gp: NULL if the plot is not drawn; otherwise, a 'ggplot' object

Note

none

See Also

xRDataLoader, xHeatmap

Examples

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

# a) provide the input Genes of interest (eg 1000 randomly chosen human genes)
## load human genes
org.Hs.eg <- xRDataLoader(RData='org.Hs.eg',
RData.location=RData.location)
set.seed(825)
data <- as.character(sample(org.Hs.eg$gene_info$Symbol, 1000))

# b) obtain repurposing matrix
DR <- xRepurpose(data, RData.location=RData.location, reorder="none",
colormap="ggplot2.top", zlim=c(1,4), na.color='transparent',
label.size=1.5, label.color="white")
DR$gp
write.table(DR$df, file="xRepurpose.txt", sep="\t", row.names=F,
quote=F)
write.table(DR$index, file="xRepurpose_index.txt", sep="\t",
row.names=F, quote=F)

## End(Not run)
```

---

xRPS *Function to calculate regulatory potential scores for genomic regions using genomic annotations*

Description

xRPS is supposed to calculate regulatory potential scores for genomic regions using genomic annotations.

Usage

```
xRPS(data, format = c("data.frame", "bed", "chr:start-end", "GRanges"),
build.conversion = c(NA, "hg38.to.hg19", "hg18.to.hg19"),
GR.annotation = c("FANTOM5_Enhancer_Cell", "FANTOM5_Enhancer_Tissue",
"FANTOM5_CAT_Cell", "FANTOM5_CAT_Tissue", "GWAScatalog_alltraits",
"ENCODE_DNaseI_ClusteredV3", "ENCODE_TFBS_ClusteredV3",
"EpigenomeAtlas_15Segments", "RecombinationRate", "phastCons100way",
"phyloP100way"), verbose = T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

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

Note

The genomic annotation data are described below according to the data sources and data types.
1. 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.

2. FANTOM5 sample-ontology-enriched CAT genes
   - `FANTOM5_CAT_Cell`: a list (173 cell types) of GenomicRanges objects; each is an GR object containing CAT genes specifically expressed in a cell type.
   - `FANTOM5_CAT_Tissue`: a list (174 tissues) of GenomicRanges objects; each is an GR object containing CAT genes specifically expressed in a tissue.
3. GWAS Catalog trait-associated SNPs
   - `GWAScatalog_alltraits`: a list (390 traits grouped by EFO) of GenomicRanges objects; each is an GR object containing trait-associated SNPs.
4. ENCODE DNaseI Hypersensitivity site data
   - `ENCODE_DNaseI_ClusteredV3`: a GR object containing clustered peaks, along with a meta-column 'num_cells' telling how many cell types associated with a clustered peak.
5. ENCODE Transcription Factor ChIP-seq data
   - `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 associated with a clustered peak.
6. Roadmap Epigenomics Core 15-state Genome Segmentation data for 127 cell types
   - `EpigenomeAtlas_15Segments`: a list (127 cell types) of a list (15 categories of segments) of GenomicRanges objects; each is an GR object containing segments per category in the reference epigenome.
7. Genomic scores
   - `RecombinationRate`: a GR object containing a meta-column for recombination rates.
   - `phastCons100way`: a GR object containing a meta-column for phastCons100way.
   - `phyloP100way`: a GR object containing a meta-column for phyloP100way.

See Also

`xEnrichViewer`

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"

# transcribed lncRNAs
FANTOM5_CAT <- xRDataLoader('FANTOM5_CAT',
RData.location=RData.location)
GR_lncRNA <- FANTOM5_CAT[grepl('lncRNA',FANTOM5_CAT$Class)]
names(GR_lncRNA) <- NULL
data <- GR_lncRNA
# RPS calculation
```

```
dGR <- xRPS(data, format="GRanges",
GR.annotation=c("FANTOM5_CAT_Cell","FANTOM5_CAT_Tissue"),
RData.location=RData.location)

## End(Not run)
```

`xRWenricher` *Function to perform connectivity enrichment analysis on the input graph*

Description

`xRWenricher` is supposed to perform connectivity enrichment analysis on the input graph given a list of nodes of interest. The test statistic is the average affinity between the given nodes. The pairwise affinity between two nodes is estimated via short random walks. The null distribution of the test statistic is generated by permuting node labels on the graph (fixed) in a centrality-preserving manner. In brief, all nodes are equally binned by centrality (defined as the mean affinity to all other nodes), and a permuted instance is generated by randomly sampling (a same number of) nodes from the same bin. The connectivity ratio is the observed divided by the expected (the median across the permutations), together with the empirical p-value.

Usage

```
xRWenricher(data, g, Amatrix = NULL, num.permutation = 2000, nbin = 10,
steps = 4, chance = 2, seed = 825, verbose = TRUE)
```

Arguments

`data`	a vector containing node names
`g`	an object of class "igraph" or "graphNEL". It will be a weighted graph if having an edge attribute 'weight'. The edge directions are ignored for directed graphs
`Amatrix`	an affinity matrix pre-computed from the input graph. It is symmetric
`num.permutation`	the number of permutations generating the null distribution
`nbin`	the number of bins dividing all nodes into the equal number of nodes
`steps`	an integer specifying the number of steps that random walk performs. By default, it is 4
`chance`	an integer specifying the chance of remaining at the same vertex. By default, it is 2, the higher the higher chance
`seed`	an integer specifying the seed
`verbose`	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

a data frame with 9 columns:

- `Obs`: the observed affinity between the given nodes
- `Exp`: the expected affinity between the permuted nodes
- `std`: the standard deviation of the expected
- `fc`: fold change
- `zscore`: z-score
- `pvalue`: p-value
- `or`: a vector containing odds ratio
- `CIl`: a vector containing lower bound confidence interval for the odds ratio
- `CIu`: a vector containing upper bound confidence interval for the odds ratio

Note

The input graph will treat as an unweighted graph if there is no 'weight' edge attribute associated with. The edge direction is not considered for the purpose of defining pairwise affinity; that is, adjacency matrix and its laplacian version are both symmetric.

See Also

`xRWenricher`

Examples

```
# 1) generate a random graph according to the ER model
set.seed(825)
g <- erdos.renyi.game(10, 3/10)
V(g)$name <- paste0('n',1:vcount(g))

## Not run:
# 2) perform connectivity enrichment analysis
data <- V(g)$name[1:3]
res <- xRWenricher(data, g, nbin=2)

## End(Not run)
```

---

xRWkernel *Function to calculate random walk kernel on the input graph*

Description

`xRWkernel` is supposed to calculate a weighted random walk kernel (at a predefined number of steps) for estimating pairwise affinity between nodes.

Usage

```
xRWkernel(g, steps = 4, chance = 2, verbose = TRUE)
```

Arguments

g	an object of class "igraph" or "graphNEL". It will be a weighted graph if having an edge attribute 'weight'. The edge directions are ignored for directed graphs
steps	an integer specifying the number of steps that random walk performs. By default, it is 4
chance	an integer specifying the chance of remaining at the same vertex. By default, it is 2, the higher the higher chance
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

It returns a sparse matrix for pairwise affinity between nodes via short random walks

Note

The input graph will treat as an unweighted graph if there is no 'weight' edge attribute associated with. The edge direction is not considered for the purpose of defining pairwise affinity; that is, adjacency matrix and its laplacian version are both symmetric.

See Also

xRWkernel

Examples

```
# 1) generate a random graph according to the ER model
set.seed(825)
g <- erdos.renyi.game(10, 3/10)
V(g)$name <- paste0('n',1:vcount(g))

## Not run:
# 2) pre-computate affinity matrix between all nodes
Amatrix <- xRWkernel(g)
# visualise affinity matrix
visHeatmapAdv(as.matrix(Amatrix), colormap="wyr",
KeyValueName="Affinity")

## End(Not run)
```

`xSimplifyNet` *Function to simplify networks from an igraph object*

Description

`xSimplifyNet` is supposed to simplify networks from an igraph object by keeping root-tip shortest paths only.

Usage

```
xSimplifyNet(g, verbose = TRUE)
```

Arguments

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

Value

an object of class "igraph"

Note

none

See Also

`xSimplifyNet`

Examples

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

## End(Not run)

RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
g <- xRDataLoader(RData.customised='ig.DO',
RData.location=RData.location)
ig <- xSimplifyNet(g)

## End(Not run)
```

xSM2DF *Function to create a data frame (with three columns) from a (sparse) matrix*

Description

xSM2DF is supposed to create a data frame (with three columns) from a (sparse) matrix. Only nonzero entries from the matrix will be kept in the resulting data frame.

Usage

```
xSM2DF(data, verbose = TRUE)
```

Arguments

data	a matrix or an object of the dgCMatrix class (a sparse matrix)
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to TRUE for display

Value

a data frame containing three columns: 1st column for row names, 2nd for column names, and 3rd for numeric values

Note

none None

See Also

xSM2DF

Examples

```
# create a sparse matrix of 4 X 2
input.file <- rbind(c('R1','C1',1), c('R2','C1',1), c('R2','C2',1),
c('R3','C2',2), c('R4','C1',1))
data <- xSparseMatrix(input.file)
# convert into a data frame
res_df <- xSM2DF(data)
res_df
```

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, score.cap = 10, distance.max = 50000,
decay.kernel = c("slow", "linear", "rapid", "constant"),
decay.exponent = 2, GR.SNP = c("dbSNP_GWAS", "dbSNP_Common",
"dbSNP_Single"), GR.Gene = c("UCSC_knownGene", "UCSC_knownCanonical"),
include.TAD = c("none", "GM12878", "IMR90", "MSC", "TRO", "H1", "MES",
"NPC"), scoring.scheme = c("max", "sum", "sequential"), verbose = T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

data	a named input vector containing the sinificance level for nodes (dbSNP). For this named vector, the element names are dbSNP ID (or in the format such as 'chr16:28525386'), the element values for the significance level (measured as p-value or fdr). Alternatively, it can be a matrix or data frame with two columns: 1st column for dbSNP, 2nd column for the significance level
include.LD	additional SNPs in LD with Lead SNPs are also included. By default, it is 'NA' to disable this option. Otherwise, LD SNPs will be included based on one or more of 26 populations and 5 super populations from 1000 Genomics Project data (phase 3). The population can be one of 5 super populations ("AFR", "AMR", "EAS", "EUR", "SAS"), or one of 26 populations ("ACB", "ASW", "BEB", "CDX", "CEU", "CHB", "CHS", "CLM", "ESN", "FIN", "GBR", "GIH", "GWD", "IBS", "ITU", "JPT", "KHV", "LWK", "MSL", "MXL", "PEL", "PJL", "PUR", "STU", "TSI", "YRI"). Explanations for population code can be found at http://www.1000genomes.org/faq/which-populations-are-part-your-study
LD.customised	a user-input matrix or data frame with 3 columns: 1st column for Lead SNPs, 2nd column for LD SNPs, and 3rd for LD r2 value. It is designed to allow the user analysing their precalcuated LD info. This customisation (if provided) has the high priority over built-in LD SNPs
LD.r2	the LD r2 value. By default, it is 0.8, meaning that SNPs in LD (r2>=0.8) with input SNPs will be considered as LD SNPs. It can be any value from 0.8 to 1

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

`score.cap`
: the maximum score being capped. By default, it is set to 10. If NULL, no capping is applied

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

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

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

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

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

`include.TAD`
: TAD boundary regions are also included. By default, it is 'NA' to disable this option. Otherwise, inclusion of a TAD dataset to pre-filter SNP-nGene pairs (i.e. only those within a TAD region will be kept). TAD datasets can be one of "GM12878" (lymphoblast), "IMR90" (fibroblast), "MSC" (mesenchymal stem cell) ,"TRO" (trophoblasts-like cell), "H1" (embryonic stem cell), "MES" (mesendoderm) and "NPC" (neural progenitor cell). Explanations can be found at http://dx.doi.org/10.1016/j.celrep.2016.10.061

`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 4 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), "Flag" (indicating as Lead or 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 XGR package and specify the location of built-in data
library(XGR)

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

## Not run:
# a) provide the seed SNPs with the significance info
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
## get lead SNPs reported in AS GWAS and their significance info (p-values)
gr <- ImmunoBase$AS$variant
data <- GenomicRanges::mcols(gr)[,c(1,3)]

# b) define and score seed geens
mSeed <- xSNP2GeneScores(data=data, include.TAD="GM12878",
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", "constant"), decay.exponent = 2, GR.SNP = c("dbSNP_GWAS",
"dbSNP_Common", "dbSNP_Single"), GR.Gene = c("UCSC_knownGene",
"UCSC_knownCanonical"), include.TAD = c("none", "GM12878", "IMR90",
"MSC",
"TRO", "H1", "MES", "NPC"), verbose = T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

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

GR.Gene	the genomic regions of genes. By default, it is 'UCSC_knownGene', that is, UCSC known genes (together with genomic locations) based on human genome assembly hg19. It can be 'UCSC_knownCanonical', that is, UCSC known canonical genes (together with genomic locations) based on human genome assembly hg19. Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to Gene Symbols. Then, tell "GR.Gene" with your RData file name (with or without extension), plus specify your file RData path in "RData.location". Note: you can also load your customised GR object directly
include.TAD	TAD boundary regions are also included. By default, it is 'none' to disable this option. Otherwise, inclusion of a TAD dataset to pre-filter SNP-nGene pairs (i.e. only those within a TAD region will be kept). TAD datasets can be one of "GM12878" (lymphoblast), "IMR90" (fibroblast), "MSC" (mesenchymal stem cell) ,"TRO" (trophoblasts-like cell), "H1" (embryonic stem cell), "MES" (mesendoderm) and "NPC" (neural progenitor cell). Explanations can be found at http://dx.doi.org/10.1016/j.celrep.2016.10.061
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details

Value

a data frame with following columns:

- Gene: nearby genes
- SNP: SNPs
- Dist: the genomic distance between the gene and the SNP
- Weight: the distance weight based on the genomic distance
- Gap: the genomic gap between the gene and the SNP (in the form of 'chr:start-end')
- TAD: if applied, it can be 'Excluded' or the TAD boundary region (in the form of 'chr:start-end') that the genomic interval falls into

Note

For details on the decay kernels, please refer to xVisKernels

See Also

xRDataLoader, xVisKernels

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
```

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

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

# c) define nearby genes (considering TAD boundary regions in GM12878)
df_nGenes <- xSNP2nGenes(data=data, distance.max=200000,
decay.kernel="slow", decay.exponent=2, include.TAD='GM12878',
RData.location=RData.location)

## End(Not run)
```

`xSNPlocations` *Function to extract genomic locations given a list of SNPs*

Description

`xSNPlocations` is supposed to extract genomic locations given a list of SNPs.

Usage

```
xSNPlocations(data, GR.SNP = c("dbSNP_GWAS", "dbSNP_Common",
"dbSNP_Single"),
verbose = T, RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

`data`	a input vector containing SNPs. SNPs should be provided as dbSNP ID (ie starting with rs). Alternatively, they can be in the format of 'chrN:xxx', where N is either 1-22 or X, xxx is genomic positional number; for example, 'chr16:28525386'
`GR.SNP`	the genomic regions of SNPs. By default, it is 'dbSNP_GWAS', that is, SNPs from dbSNP (version 146) restricted to GWAS SNPs and their LD SNPs (hg19). It can be 'dbSNP_Common', that is, Common SNPs from dbSNP (version 146) plus GWAS SNPs and their LD SNPs (hg19). Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to dbSNP IDs. Then, tell "GR.SNP" with your RData file name (with or without extension), plus specify your file RData path in "RData.location". Note: you can also load your customised GR object directly

verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details

Value

an GR oject, with an additional metadata column called 'variant_id' storing SNP location in the format of 'chrN:xxx', where N is either 1-22 or X, xxx is genomic positional number.

Note

none

See Also

xRDataLoader

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)

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

## Not run:
# a) provide the seed SNPs with the significance info
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
## get lead SNPs reported in AS GWAS and their significance info (p-values)
gr <- ImmunoBase$AS$variant
data <- names(gr)

# b) define nearby genes
snp_gr <- xSNPlocations(data=data, RData.location=RData.location)

## End(Not run)
```

---

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

Description

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

Usage

```
xSNPscores(data, include.LD = NA, LD.customised = NULL, LD.r2 = 0.8,
significance.threshold = 5e-05, score.cap = 10, verbose = T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

data	a named input vector containing the significance level for nodes (dbSNP). For this named vector, the element names are dbSNP (starting with rs or in the format of 'chrN:xxx', where N is either 1-22 or X, xxx is number; for example, 'chr16:28525386'), the element values for the significance level (measured as p-value or fdr). Alternatively, it can be a matrix or data frame with two columns: 1st column for dbSNP, 2nd column for the significance level.
include.LD	additional SNPs in LD with Lead SNPs are also included. By default, it is 'NA' to disable this option. Otherwise, LD SNPs will be included based on one or more of 26 populations and 5 super populations from 1000 Genomics Project data (phase 3). The population can be one of 5 super populations ("AFR", "AMR", "EAS", "EUR", "SAS"), or one of 26 populations ("ACB", "ASW", "BEB", "CDX", "CEU", "CHB", "CHS", "CLM", "ESN", "FIN", "GBR", "GIH", "GWD", "IBS", "ITU", "JPT", "KHV", "LWK", "MSL", "MXL", "PEL", "PJL", "PUR", "STU", "TSI", "YRI"). Explanations for population code can be found at http://www.1000genomes.org/faq/which-populations-are-part-your-study
LD.customised	a user-input matrix or data frame with 3 columns: 1st column for Lead SNPs, 2nd column for LD SNPs, and 3rd for LD r2 value. It is designed to allow the user analysing their precalcuated LD info. This customisation (if provided) has the high priority over built-in LD SNPs
LD.r2	the LD r2 value. By default, it is 0.8, meaning that SNPs in LD (r2>=0.8) with input SNPs will be considered as LD SNPs. It can be any value from 0.8 to 1
significance.threshold	the given significance threshold. By default, it is set to NULL, meaning there is no constraint on the significance level when transforming the significance level of SNPs into scores. If given, those SNPs below this are considered significant and thus scored positively. Instead, those above this are considered insigificant and thus receive no score
score.cap	the maximum score being capped. By default, it is set to 10. If NULL, no capping is applied
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display
RData.location	the characters to tell the location of built-in RData files. See xRDataLoader for details

Value

a data frame with following columns:

- SNP: Lead and/or LD SNPs

- `Score`: the scores for SNPs calculated based on p-values taking into account the given threshold of the significant level
- `Pval`: the input p-values for Lead SNPs or R2-adjusted p-values for LD SNPs
- `Flag`: the flag to indicate whether the resulting SNPs are Lead SNPs or LD SNPs

Note

None

See Also

xRDataLoader

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)

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

## Not run:
# a) provide the seed SNPs with the significance info
## load ImmunoBase
ImmunoBase <- xRDataLoader(RData.customised='ImmunoBase',
RData.location=RData.location)
## get lead SNPs reported in AS GWAS and their significance info (p-values)
gr <- ImmunoBase$AS$variant
data <- GenomicRanges::mcols(gr)[,c(1,3)]

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

## End(Not run)
```

`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.

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 https://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. It will depend on whether these two packages "foreach" and "doParallel" have been installed. It can be installed via: `source("http://bioconductor.org/biocLite.R"); biocLite(c("foreach","doParallel"))` 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.

Usage

```
xSocialiserGenes(data, check.symbol.identity = F, 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 = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

data	an input vector containing gene symbols
check.symbol.identity	logical to indicate whether to match the input data via Synonyms for those unmatchable by official gene symbols. By default, it sets to false
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 https://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. It will depend on whether these two packages "foreach" and "doParallel" have been installed. It can be installed via: `source("http://bioconductor.org/biocLite.R"); biocLite(c("foreach","doParallel")`) 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 XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata/"

# 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',
RData.location=RData.location)
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.

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 = "http://galahad.well.ox.ac.uk/bigdata")
```

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 https://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. It will depend on whether these two packages "foreach" and "doParallel" have been installed. It can be installed via: `source("http://bioconductor.org/biocLite.R"); biocLite(c("foreach","doParallel")`) 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 XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata/"

# 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', RData.location=RData.location)
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", "STRING_low", "PCommonsUN_high", "PCommonsUN_medium",
"PCommonsDN_high", "PCommonsDN_medium", "PCommonsDN_Reactome",
"PCommonsDN_KEGG", "PCommonsDN_HumanCyc", "PCommonsDN_PID",
"PCommonsDN_PANTHER", "PCommonsDN_ReconX", "PCommonsDN_TRANSFAC",
"PCommonsDN_PhosphoSite", "PCommonsDN_CTD", "KEGG", "KEGG_metabolism",
"KEGG_genetic", "KEGG_environmental", "KEGG_cellular",
"KEGG_organismal",
"KEGG_disease", "REACTOME"), STRING.only = c(NA, "neighborhood_score",
"fusion_score", "cooccurence_score", "coexpression_score",
"experimental_score", "database_score", "textmining_score")[1],
network.customised = NULL, seed.genes = T, subnet.significance = 0.01,
subnet.size = NULL, test.permutation = F, num.permutation = 100,
respect = c("none", "degree"), aggregateBy = c("Ztransform", "fishers",
"logistic", "orderStatistic"), verbose = T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

data	a named input vector containing the significance level for nodes (gene symbols). For this named vector, the element names are gene symbols, the element values for the significance level (measured as p-value or fdr). Alternatively, it can be a matrix or data frame with two columns: 1st column for gene symbols, 2nd column for the significance level
network	the built-in network. Currently two sources of network information are supported: the STRING database (version 10) and the Pathway Commons database (version 7). STRING is a meta-integration of undirect interactions from the functional aspect, while Pathways Commons mainly contains both undirect and direct interactions from the physical/pathway aspect. Both have scores to control the confidence of interactions. Therefore, the user can choose the different quality of the interactions. In STRING, "STRING_highest" indicates interactions with highest confidence (confidence scores>=900), "STRING_high" for interactions with high confidence (confidence scores>=700), "STRING_medium" for interactions with medium confidence (confidence scores>=400), and "STRING_low" for interactions with low confidence (confidence scores>=150). For undirect/physical

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

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

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

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

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

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

`test.permutation`
: logical to indicate whether the permutation test is perform to estimate the significance of identified network with the same number of nodes. By default, it sets to false

`num.permutation`
: the number of permutations generating the null distribution of the identified network

`respect`
: how to respect nodes to be sampled. It can be one of 'none' (randomly sampling) and 'degree' (degree-preserving sampling)

`aggregateBy`
: the aggregate method used to aggregate edge confidence p-values. It can be either "orderStatistic" for the method based on the order statistics of p-values, or "fishers" for Fisher's method, "Ztransform" for Z-transform method, "logistic" for the logistic method. Without loss of generality, the Z-transform method does well in problems where evidence against the combined null is spread widely (equal footings) or when the total evidence is weak; Fisher's method does best in problems where the evidence is concentrated in a relatively small fraction of the individual tests or when the evidence is at least moderately strong; the logistic method provides a compromise between these two. Notably, the aggregate methods 'Ztransform' and 'logistic' are preferred here

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

`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 node attributes: significance, score, type. If permutation test is enabled, it also has a graph attribute (combinedP) and an edge attribute (edgeConfidence)

Note

The algorithm identifying a subnetwork is implemented in the dnet package (http://genomemedicine.biomedcentral.com/artic
014-0064-8). In brief, from an input network with input node/gene information (the significant level; p-values or FDR), the way of searching for a maximum-scoring subnetwork is done as follows. Given the threshold of tolerable p-value, it gives positive scores for nodes with p-values below the threshold (nodes of interest), and negative scores for nodes with threshold-above p-values (intolerable). After score transformation, the search for a maximum scoring subnetwork is deduced to find the connected subnetwork that is enriched with positive-score nodes, allowing for a few negative-score nodes as linkers. This objective is met through minimum spanning tree finding and post-processing, previously used as a heuristic solver of prize-collecting Steiner tree problem. The solver is deterministic, only determined by the given tolerable p-value threshold. For identification of the subnetwork with a desired number of nodes, an iterative procedure is also developed to fine-tune tolerable thresholds. This explicit control over the node size may be necessary for guiding follow-up experiments.

See Also

`xRDataLoader`, `xDefineNet`

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata/"

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

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

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

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

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

# g) visualise the subnet using the same layout_with_kk
df_tmp <- df[match(V(subnet)$name,df$Symbol),]
vec_tmp <- colnames(df_tmp)
names(vec_tmp) <- vec_tmp
glayout <- igraph::layout_with_kk(subnet)
V(subnet)$xcoord <- glayout[,1]
V(subnet)$xcoord <- glayout[,2]
# g1) colored according to FDR
ls_ig <- lapply(vec_tmp, function(x){
ig <- subnet
```

```
V(ig)$fdr <- -log10(as.numeric(df_tmp[,x]))
ig
})
gp_FDR <- xA2Net(g=ls_g, node.label='name', node.label.size=2,
node.label.color='blue', node.label.alpha=0.8, node.label.padding=0.25,
node.label.arrow=0, node.label.force=0.1, node.shape=19,
node.xcoord='xcoord', node.ycoord='ycoord', node.color='fdr',
node.color.title=expression(-log[10]('FDR')),
colormap='grey-yellow-orange', ncolors=64, zlim=c(0,3),
node.size.range=4,
edge.color="black",edge.color.alpha=0.3,edge.curve=0.1,edge.arrow.gap=0.025)
# g2) colored according to FC
ls_ig <- lapply(vec_tmp, function(x){
ig <- subnet
V(ig)$lfc <- as.numeric(df_tmp[,x])
ig
})
gp_FC <- xA2Net(g=ls_g, node.label='name', node.label.size=2,
node.label.color='blue', node.label.alpha=0.8, node.label.padding=0.25,
node.label.arrow=0, node.label.force=0.1, node.shape=19,
node.xcoord='xcoord', node.ycoord='ycoord', node.color='lfc',
node.color.title=expression(log[2]('FC')), colormap='cyan1-grey-pink1',
ncolors=64, zlim=c(-3,3), node.size.range=4,
edge.color="black",edge.color.alpha=0.3,edge.curve=0.1,edge.arrow.gap=0.025)
# g3) colored according to FC
gridExtra::grid.arrange(grobs=list(gp_FDR, gp_FC), ncol=2,
as.table=TRUE)

## End(Not run)
```

---

xSubneterGR *Function to identify a gene network from an input network given a list of genomic regions together with the significance level*

Description

xSubneterGR 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 genomic regions (GR). It returns an object of class "igraph".

Usage

```
xSubneterGR(data, significance.threshold = 5e-05, score.cap = 10,
build.conversion = c(NA, "hg38.to.hg19", "hg18.to.hg19"),
distance.max = 50000, decay.kernel = c("slow", "linear", "rapid",
"constant"), decay.exponent = 2, GR.Gene = c("UCSC_knownGene",
"UCSC_knownCanonical"), scoring.scheme = c("max", "sum", "sequential"),
network = c("STRING_highest", "STRING_high", "STRING_medium",
```

```
"STRING_low",
"PCommonsUN_high", "PCommonsUN_medium", "PCommonsDN_high",
"PCommonsDN_medium", "PCommonsDN_Reactome", "PCommonsDN_KEGG",
"PCommonsDN_HumanCyc", "PCommonsDN_PID", "PCommonsDN_PANTHER",
"PCommonsDN_ReconX", "PCommonsDN_TRANSFAC", "PCommonsDN_PhosphoSite",
"PCommonsDN_CTD", "KEGG", "KEGG_metabolism", "KEGG_genetic",
"KEGG_environmental", "KEGG_cellular", "KEGG_organismal",
"KEGG_disease",
"REACTOME"), network.customised = NULL, seed.genes = T,
subnet.significance = 5e-05, subnet.size = NULL, verbose = T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

data	a named input vector containing the sinificance level for genomic regions (GR). For this named vector, the element names are GR, in the format of 'chrN:start-end', where N is either 1-22 or X, start (or end) is genomic positional number; for example, 'chr1:13-20'. The element values for the significance level (measured as p-value or fdr). Alternatively, it can be a matrix or data frame with two columns: 1st column for GR, 2nd column for the significance level.
significance.threshold	the given significance threshold. By default, it is set to NULL, meaning there is no constraint on the significance level when transforming the significance level of GR into scores. If given, those GR below this are considered significant and thus scored positively. Instead, those above this are considered insigificant and thus receive no score
score.cap	the maximum score being capped. By default, it is set to 10. If NULL, no capping is applied
build.conversion	the conversion from one genome build to another. The conversions supported are "hg38.to.hg19" and "hg18.to.hg19". By default it is NA (no need to do so)
distance.max	the maximum distance between genes and GR. Only those genes no far way from this distance will be considered as seed genes. This parameter will influence the distance-component weights calculated for nearby GR per gene
decay.kernel	a character specifying a decay kernel function. It can be one of 'slow' for slow decay, 'linear' for linear decay, and 'rapid' for rapid decay. If no distance weight is used, please select 'constant'
decay.exponent	a numeric specifying a decay exponent. By default, it sets to 2
GR.Gene	the genomic regions of genes. By default, it is 'UCSC_knownGene', that is, UCSC known genes (together with genomic locations) based on human genome assembly hg19. It can be 'UCSC_knownCanonical', that is, UCSC known canonical genes (together with genomic locations) based on human genome assembly hg19. Alternatively, the user can specify the customised input. To do so, first save your RData file (containing an GR object) into your local computer, and make sure the GR object content names refer to Gene Symbols. Then, tell "GR.Gene" with your RData file name (with or without extension), plus specify

your file RData path in "RData.location". Note: you can also load your customised GR object directly

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

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

`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, type

Note

The algorithm identifying a gene subnetwork that is likely modulated by input genomic regions (GR) includes two major steps. The first step is to use `xGR2GeneScores` for defining and scoring nearby genes that are located within distance window of input GR. 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

`xGR2GeneScores`, `xSubneterGenes`

Examples

```
## Not run:
# Load the XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata/"

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

```
end <- df$end
sig <- df$Pvalue
GR <- paste(chr,':',start,'-',end, sep='')
data <- cbind(GR=GR, Sig=sig)

# b) perform network analysis
# b1) find maximum-scoring subnet based on the given significance threshold
subnet <- xSubneterGR(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 <- xSubneterGR(data=data, network="STRING_high", seed.genes=F,
subnet.size=30, 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
xVisNet(g=subnet, pattern=-log10(as.numeric(V(subnet)$significance)),
vertex.shape="sphere", colormap="wyr")
## do visualisation with nodes colored according to transformed scores
xVisNet(g=subnet, pattern=as.numeric(V(subnet)$score),
vertex.shape="sphere")

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

## 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, score.cap = 10, distance.max = 2e+05,
decay.kernel = c("slow", "linear", "rapid", "constant"),
decay.exponent = 2, GR.SNP = c("dbSNP_GWAS", "dbSNP_Common",
"dbSNP_Single"), GR.Gene = c("UCSC_knownGene", "UCSC_knownCanonical"),
scoring.scheme = c("max", "sum", "sequential"),
network = c("STRING_highest", "STRING_high", "STRING_medium",
"STRING_low",
"PCommonsUN_high", "PCommonsUN_medium", "PCommonsDN_high",
"PCommonsDN_medium", "PCommonsDN_Reactome", "PCommonsDN_KEGG",
"PCommonsDN_HumanCyc", "PCommonsDN_PID", "PCommonsDN_PANTHER",
"PCommonsDN_ReconX", "PCommonsDN_TRANSFAC", "PCommonsDN_PhosphoSite",
"PCommonsDN_CTD", "KEGG", "KEGG_metabolism", "KEGG_genetic",
"KEGG_environmental", "KEGG_cellular", "KEGG_organismal",
"KEGG_disease",
"REACTOME"), network.customised = NULL, seed.genes = T,
subnet.significance = 5e-05, subnet.size = NULL, verbose = T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

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

and thus scored positively. Instead, those above this are considered insigificant and thus receive no score

`score.cap` the maximum score being capped. By default, it is set to 10. If NULL, no capping is applied

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

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

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

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

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

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

`network` the built-in network. Currently two sources of network information are supported: the STRING database (version 10) and the Pathway Commons database (version 7). STRING is a meta-integration of undirect interactions from the functional aspect, while Pathways Commons mainly contains both undirect and direct interactions from the physical/pathway aspect. Both have scores to control the confidence of interactions. Therefore, the user can choose the different quality of the interactions. In STRING, "STRING_highest" indicates interactions with highest confidence (confidence scores>=900), "STRING_high" for interactions with high confidence (confidence scores>=700), "STRING_medium" for interactions with medium confidence (confidence scores>=400), and "STRING_low" for interactions with low confidence (confidence scores>=150). For undirect/physical interactions from Pathways Commons, "PCommonsUN_high" indicates undirect interactions with high confidence (supported with the PubMed references

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

`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, type

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 XGR package and specify the location of built-in data
library(XGR)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata/"

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

# b) 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, 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
xVisNet(g=subnet, pattern=-log10(as.numeric(V(subnet)$significance)),
vertex.shape="sphere", colormap="wyr")
```

```
## do visualisation with nodes colored according to transformed scores
xVisNet(g=subnet, pattern=as.numeric(V(subnet)$score),
vertex.shape="sphere")

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

## End(Not run)
```

`xSymbol2GeneID` *Function to convert gene symbols to entrez geneid*

Description

`xSymbol2GeneID` is supposed to convert gene symbols to entrez geneid.

Usage

```
xSymbol2GeneID(data, check.symbol.identity = F, details = F, verbose =
T,
RData.location = "http://galahad.well.ox.ac.uk/bigdata")
```

Arguments

`data`	an input vector containing gene symbols
`check.symbol.identity`	logical to indicate whether to match the input data via Synonyms for those unmatchable by official gene symbols. By default, it sets to false
`details`	logical to indicate whether to result in a data frame (in great details). By default, it sets to false
`verbose`	logical to indicate whether the messages will be displayed in the screen. By default, it sets to false for no display
`RData.location`	the characters to tell the location of built-in RData files. See `xRDataLoader` for details

Value

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

Note

If a symbol mapped many times, the one assiged as the "protein-coding" type of gene is preferred.

See Also

xEnricherGenes, xSocialiserGenes

Examples

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

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

# b) convert into GeneID
GeneID <- xSymbol2GeneID(Symbol)

# c) convert into a data frame
df <- xSymbol2GeneID(Symbol, details=TRUE)

## End(Not run)
```

---

xVisInterp *Function to visualise interpolated irregular data*

Description

xVisInterp is supposed to visualise irregular data after bilinear or bicubic spline interpolation onto a grid.

Usage

```
xVisInterp(ls_xyz, interpolation = c("spline", "linear"), nx = 100,
ny = 100, zlim = NULL, nD = c("auto", "2D", "3D"), colkey = TRUE,
contour = FALSE, image = FALSE, clab = c("Value", ""), nlevels = 20,
colormap = "terrain", label.pch = 17, label.text.cex = 0.8,
label.text.adj = -0.4, label.text.adj.z = 0.01,
label.font.family = "sans", xy.swap = FALSE, theta.3D = 40,
phi.3D = 20, verbose = TRUE)
```

Arguments

ls_xyz	a list with 3 required components (x, y and z) and an optional component (label)
interpolation	the method for the interpolation. It can be "linear" or "spline" interpolation
nx	the dimension of output grid in x direction
ny	the dimension of output grid in y direction

`zlim` the minimum and maximum z values, defaulting to the range of the finite values of z

`nD` an integer specifying the dimension of the visualisation. It can be one of '2D', '3D', and 'auto' (to display the input raw data as well in 2D)

`colkey` a logical (TRUE by default) or a 'list' with parameters for the color key (legend). List parameters should be one of 'side, plot, length, width, dist, shift, addlines, col.clab, cex.clab, side.clab, line.clab, adj.clab, font.clab'. The defaults for the parameters are 'side=4, plot=TRUE, length=1, width=1, dist=0, shift=0, addlines=FALSE, col.clab=NULL, cex.clab=par("cex.lab"), side.clab=NULL, line.clab=NULL, adj.clab=NULL, font.clab=NULL'. colkey=list(side=4,length=0.15,width=0.5,shift=0.3 0.15,cex.axis=0.6,cex.clab=0.8,side.clab=3)

`contour` a logical (FALSE by default) or a 'list' with parameters for the contour function. An optional parameter to this 'list' is the 'side' where the image should be plotted. Allowed values for 'side' are a z-value, or 'side = "zmin", "zmax"', for positioning at bottom or top respectively. The default is to put the image at the bottom

`image` a logical (FALSE by default) or a 'list' with parameters for the image2D function. An optional parameter to this 'list' is the 'side' where the image should be plotted. Allowed values for 'side' are a z-value, or 'side = "zmin", "zmax"', for positioning at bottom or top respectively. The default is to put the image at the bottom

`clab` a title for the colorbar. the label to be written on top of the color key; to lower it, 'clab' can be made a vector, with the first values empty strings.

`nlevels` the number of levels to partition the input matrix values. The same level has the same color mapped to

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

`label.pch` a numeric value specifying the graphiics symbol (by default, 17 for upward triangle). This argument only works when the labelling is enabled

`label.text.cex` a numeric value specifying the text size. This argument only works when the labelling is enabled

`label.text.adj` a numeric value adjusting the text location in xy-plane. This argument only works when the labelling is enabled

`label.text.adj.z` a numeric value adjusting the text locaion in z-axis. This argument only works when the labelling is enabled

`label.font.family` the font family for texts. This argument only works when the labelling is enabled

`xy.swap` logical to indicate whether to wrap x and y. By default, it sets to false

theta.3D	the azimuthal direction. By default, it is 40
phi.3D	the colatitude direction. By default, it is 20
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display

Value

invisible

Note

none

See Also

xVisInterp

Examples

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

## End(Not run)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
g <- erdos.renyi.game(20, 1/10)
glayout <- layout_with_kk(g)
ls_xyz <- data.frame(x=glayout[,1], y=glayout[,2], z=degree(g),
label=degree(g))

# auto
ls_xyz.smooth <- xVisInterp(ls_xyz, nD="auto")
# 2D
ls_xyz.smooth <- xVisInterp(ls_xyz, nD="2D")
# 3D
xVisInterp(ls_xyz, nD="3D", theta.3D=40, phi.3D=20, clab="Value\n")
# 3D views of different angles
pdf("xVisInterp.pdf")
for(theta.3D in seq(0,360,10)){ xVisInterp(ls_xyz, nD="3D",
contour=TRUE, image=TRUE, clab=paste0("theta:",theta.3D,"\n"),
theta.3D=theta.3D, phi.3D=20)}
dev.off()

## End(Not run)
```

`xVisInterpAnimate` *Function to animate the visualisation of interpolated irregular data*

Description

`xVisInterpAnimate` is supposed to animate the visualisation of interpolated irregular data. The output can be a pdf file containing a list of frames/images, a mp4 video file or a gif file. To support video output file, the software 'ffmpeg' must be first installed (also put its path into the system PATH variable; see Note). To support gif output file, the software 'ImageMagick' must be first installed (also put its path into the system PATH variable; see Note).

Usage

```
xVisInterpAnimate(ls_xyz, interpolation = c("spline", "linear"), nx =
100,
ny = 100, zlim = NULL, colkey = TRUE, contour = FALSE,
image = FALSE, clab = c("Value"), nlevels = 20, colormap = "terrain",
label.pch = 17, label.text.cex = 0.8, label.text.adj = -0.4,
label.text.adj.z = 0.01, label.font.family = "sans", xy.swap = FALSE,
theta.3D = 0, phi.3D = 20, verbose = TRUE,
filename = "xVisInterpAnimate", filetype = c("pdf", "mp4", "gif"),
image.type = c("jpg", "png"), image.bg = "transparent",
height.device = NULL, num.frame = 36, sec_per_frame = 1, res = 72)
```

Arguments

`ls_xyz`	a list with 3 required components (x, y and z) and an optional component (label)
`interpolation`	the method for the interpolation. It can be "linear" or "spline" interpolation
`nx`	the dimension of output grid in x direction
`ny`	the dimension of output grid in y direction
`zlim`	the minimum and maximum z values, defaulting to the range of the finite values of z
`colkey`	a logical (TRUE by default) or a 'list' with parameters for the color key (legend). List parameters should be one of 'side, plot, length, width, dist, shift, addlines, col.clab, cex.clab, side.clab, line.clab, adj.clab, font.clab'. The defaults for the parameters are 'side=4, plot=TRUE, length=1, width=1, dist=0, shift=0, addlines=FALSE, col.clab=NULL, cex.clab=par("cex.lab"), side.clab=NULL, line.clab=NULL, adj.clab=NULL, font.clab=NULL'. colkey=list(side=4,length=0.15,width=0.5,shift=0.3 0.15,cex.axis=0.6,cex.clab=0.8,side.clab=3)
`contour`	a logical (FALSE by default) or a 'list' with parameters for the contour function. An optional parameter to this 'list' is the 'side' where the image should be plotted. Allowed values for 'side' are a z-value, or 'side = "zmin", "zmax"', for positioning at bottom or top respectively. The default is to put the image at the bottom

image	a logical (FALSE by default) or a 'list' with parameters for the image2D function. An optional parameter to this 'list' is the 'side' where the image should be plotted. Allowed values for 'side' are a z-value, or 'side = "zmin", "zmax"', for positioning at bottom or top respectively. The default is to put the image at the bottom
clab	a title for the colorbar. the label to be written on top of the color key; to lower it, 'clab' can be made a vector, with the first values empty strings.
nlevels	the number of levels to partition the input matrix values. The same level has the same color mapped to
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), and "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta). Alternatively, any hyphen-separated HTML color names, e.g. "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
label.pch	a numeric value specifying the graphiics symbol (by default, 17 for upward triangle). This argument only works when the labelling is enabled
label.text.cex	a numeric value specifying the text size. This argument only works when the labelling is enabled
label.text.adj	a numeric value adjusting the text location in xy-plane. This argument only works when the labelling is enabled
label.text.adj.z	a numeric value adjusting the text locaion in z-axis. This argument only works when the labelling is enabled
label.font.family	the font family for texts. This argument only works when the labelling is enabled
xy.swap	logical to indicate whether to wrap x and y. By default, it sets to false
theta.3D	the starting azimuthal direction. By default, it is 0
phi.3D	the colatitude direction. By default, it is 20
verbose	logical to indicate whether the messages will be displayed in the screen. By default, it sets to true for display
filename	the without-extension part of the name of the output file. By default, it is 'xVisInterpAnimate'
filetype	the type of the output file, i.e. the extension of the output file name. It can be one of either 'pdf' for the pdf file, 'mp4' for the mp4 video file, 'gif' for the gif file
image.type	the type of the image files temporarily generated. It can be one of either 'jpg' or 'png'. These temporary image files are used for producing mp4/gif output file. The reason doing so is to accommodate that sometimes only one of image types is supported so that you can choose the right one
image.bg	the background color for each frame/image. This argument only works when producing mp4 video or gif file

height.device	a numeric value specifying the height (or width) of device/frame/image.
num.frame	a numeric value specifying the number of frames/images. By default, it sets to the number of columns in the input data matrix
sec_per_frame	a numeric value specifying how long (seconds) it takes to stream a frame/image. This argument only works when producing mp4 video or gif file
res	the resolution for each frame/image. This argument only works when producing mp4 video or gif file

Value

If specifying the output file name (see argument 'filename' above), the output file is either 'filename.pdf' or 'filename.mp4' or 'filename.gif' in the current working directory. If no output file name specified, by default the output file is either 'xVisInterpAnimate.pdf' or 'xVisInterpAnimate.mp4' or 'xVisInterpAnimate.gif'

Note

When producing mp4 video, this function requires the installation of the software 'ffmpeg' at https://www.ffmpeg.org. Shell command lines for ffmpeg installation in Terminal (for both Linux and Mac) are:

- 1) `wget -O ffmpeg.tar.gz http://www.ffmpeg.org/releases/ffmpeg-2.7.1.tar.gz`
- 2) `mkdir ~/ffmpeg | tar xvfz ffmpeg.tar.gz -C ~/ffmpeg --strip-components=1`
- 3) `cd ffmpeg`
- 4a) # Assuming you want installation with a ROOT (sudo) privilege:
  `./configure --disable-yasm`
- 4b) # Assuming you want local installation without ROOT (sudo) privilege:
  `./configure --disable-yasm --prefix=$HOME/ffmpeg`
- 5) `make`
- 6) `make install`
- 7) # add the system PATH variable to your ~/.bash_profile file if you follow 4b) route:
  `export PATH=$HOME/ffmpeg:$PATH`
- 8) # make sure ffmpeg has been installed successfully:
  `ffmpeg -h`

When producing gif file, this function requires the installation of the software 'ImageMagick' at http://www.imagemagick.org. Shell command lines for ImageMagick installation in Terminal are:

- 1) `wget http://www.imagemagick.org/download/ImageMagick.tar.gz`
- 2) `mkdir ~/ImageMagick | tar xvzf ImageMagick.tar.gz -C ~/ImageMagick --strip-components=1`
- 3) `cd ImageMagick`
- 4) `./configure --prefix=$HOME/ImageMagick`
- 5) `make`
- 6) `make install`

- 7) # add the system PATH variable to your ~/.bash_profile file.
  For Linux:
  `export MAGICK_HOME=$HOME/ImageMagick`
  `export PATH=$MAGICK_HOME/bin:$PATH`
  `export LD_LIBRARY_PATH=${LD_LIBRARY_PATH:+$LD_LIBRARY_PATH:}$MAGICK_HOME/lib`
  For Mac:
  `export MAGICK_HOME=$HOME/ImageMagick`
  `export PATH=$MAGICK_HOME/bin:$PATH`
  `export DYLD_LIBRARY_PATH=$MAGICK_HOME/lib/`
- 8a) # check configuration:
  `convert -list configure`
- 8b) # check image format supported:
  `identify -list format`
- Tips:
  Prior to 4), please make sure `libjpeg` and `libpng` are installed. If NOT, for Mac try this:
  `brew install libjpeg libpng librsvg`
  To check whether ImageMagick does work, please get additional information from:
  `identify -list format`
  `convert -list configure`
  On details, please refer to http://www.imagemagick.org/script/advanced-unix-installation.php

See Also

visNetMul

Examples

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

## End(Not run)
RData.location <- "http://galahad.well.ox.ac.uk/bigdata"
## Not run:
g <- erdos.renyi.game(20, 1/10)
glayout <- layout_with_kk(g)
ls_xyz <- data.frame(x=glayout[,1], y=glayout[,2], z=degree(g),
label=degree(g))

# 3D views of different angles
# output as a pdf file
xVisInterpAnimate(ls_xyz, image=TRUE, filetype="pdf")
# output as a mp4 file
xVisInterpAnimate(ls_xyz, filetype="mp4")
# output as a gif file
xVisInterpAnimate(ls_xyz, filetype="gif", num.frame=72,
sec_per_frame=0.5)

## 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 = TRUE,
newpage = TRUE, signature = TRUE, glayout = layout_with_kk,
vertex.frame.color = NA, vertex.size = NULL, vertex.color = NULL,
vertex.shape = NULL, vertex.label = NULL, vertex.label.cex = NULL,
vertex.label.dist = 0.3, vertex.label.color = "blue",
vertex.label.family = "sans", edge.arrow.size = 0.3, ...)
```

Arguments

g	an object of class "igraph"
pattern	a numeric vector used to color-code vertices/nodes. Notably, if the input vector contains names, then these names should include all node names of input graph, i.e. V(g)$name, since there is a mapping operation. After mapping, the length of the patern vector should be the same as the number of nodes of input graph; otherwise, this input pattern will be ignored. The way of how to color-code is to map values in the pattern onto the whole colormap (see the next arguments: colormap, ncolors, zlim and colorbar)
colormap	short name for the colormap. It can be one of "jet" (jet colormap), "bwr" (blue-white-red colormap), "gbr" (green-black-red colormap), "wyr" (white-yellow-red colormap), "br" (black-red colormap), "yr" (yellow-red colormap), "wb" (white-black colormap), "rainbow" (rainbow colormap, that is, red-yellow-green-cyan-blue-magenta), and "ggplot2" (emulating ggplot2 default color palette). Alternatively, any hyphen-separated HTML color names, e.g. "lightyellow-orange" (by default), "blue-black-yellow", "royalblue-white-sandybrown", "darkgreen-white-darkviolet". A list of standard color names can be found in http://html-color-codes.info/color-names
ncolors	the number of colors specified over the colormap
zlim	the minimum and maximum z/patttern values for which colors should be plotted, defaulting to the range of the finite values of z. Each of the given colors will be used to color an equispaced interval of this range. The midpoints of the intervals cover the range, so that values just outside the range will be plotted
colorbar	logical to indicate whether to append a colorbar. If pattern is null, it always sets to false

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

signature — logical to indicate whether the signature is assigned to the plot caption. By default, it sets FALSE

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

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

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

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

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

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

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

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

vertex.label.color — the color of vertex labels

vertex.label.family — the font family of vertex labels

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

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

Value

invisible

Note

none

See Also

xSubneterGenes, xSubneterSNPs

Examples

```
## Not run:
# 1) generate a ring graph
g <- make_ring(10, directed=TRUE)

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

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

# 5) use font family (Arial)
source("http://bioconductor.org/biocLite.R"); biocLite("extrafont")
library(extrafont)
font_import()
fonttable()
## creating PDF files with fonts
loadfonts()
pdf("xVisNet_fonts.pdf", family="Arial Black")
xVisNet(g=g, newpage=FALSE, vertex.label.family="Arial Black",
signature=F)
dev.off()

## End(Not run)
```

---

xVolcano *Function to draw a volcano plot*

Description

xVolcano is supposed to draw a volcano plot

Usage

```
xVolcano(data, column.lfc = "lfc", column.fdr = "fdr", cutoff.lfc = 1,
cutoff.fdr = 0.05, colors = c("#EEEEEE", "darkgrey", "pink", "red"),
column.label = NULL, top = 10, top.direction = c("both", "up", "down"),
label.size = 2, label.color = "black", label.alpha = 0.8,
label.padding = 0.5, label.arrow = 0.01, label.force = 0.5,
xlim = NULL, ylim = NULL, y.scale = c("normal", "log"),
xlab = expression(log[2]("fold change")),
ylab = expression(-log[10]("FDR")), font.family = "sans",
signature = TRUE)
```

Arguments

data	a data frame
column.lfc	a character specifying 'lfc' column (log2-transformed fold change)
column.fdr	a character specifying 'fdr' column
cutoff.lfc	a numeric defining 'lfc' cutoff. By default, it is 1 (at least 2-fold changes)
cutoff.fdr	a numeric defining 'fdr' cutoff. By default, it is 0.05
colors	a 4-element vector for color-coded points. By default, it is c("#EEEEEE","darkgrey","pink","red")
column.label	a character specifying 'label' column
top	an integer specifying the number of the top points for labellings
top.direction	the direction (up- and down-regulated) of the top points. It can be one of 'both' (up- and down-regulated), 'up' (up-regulated only) and 'down' (down-regulated only)
label.size	the label size
label.color	the label color
label.alpha	the 0-1 value specifying transparency of labelling
label.padding	the padding around the labeled
label.arrow	the arrow pointing to the labeled
label.force	the repelling force between overlapping labels
xlim	the limits in the x-axis
ylim	the limits in the y-axis
y.scale	how to transform the y scale. It can be "normal" for no transformation, and "log" for log-based transformation
xlab	the x labelling. By default, it is expression(log[2]("fold change"))
ylab	the y labelling. By default, it is expression(-log[10]("FDR"))
font.family	the font family for texts
signature	logical to indicate whether the signature is assigned to the plot caption. By default, it sets TRUE

Value

a ggplot object

Note

none

See Also

xVolcano

Examples

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

## End(Not run)
```

Index