Package: pathview (via r-universe)

November 1, 2026

Type Package

Title a tool set for pathway based data integration and visualization

Version 1.52.0

Date 2025-03-28

Author Weijun Luo

Maintainer Weijun Luo <luo_weijun@yahoo.com>

Description Pathview is a tool set for pathway based data integration and visualization. It maps and renders a wide variety of biological data on relevant pathway graphs. All users need is to supply their data and specify the target pathway. Pathview automatically downloads the pathway graph data, parses the data file, maps user data to the pathway, and render pathway graph with the mapped data. In addition, Pathview also seamlessly integrates with pathway and gene set (enrichment) analysis tools for large-scale and fully automated analysis.

biocViews Pathways, GraphAndNetwork, Visualization, GeneSetEnrichment, DifferentialExpression, GeneExpression, Microarray, RNASeq, Genetics, Metabolomics, Proteomics, SystemsBiology, Sequencing

Depends R (>= 2.10)

Imports KEGGgraph, XML, Rgraphviz, graph, png, AnnotationDbi, org.Hs.eg.db, KEGGREST, methods, utils

Suggests gage, org.Mm.eg.db, RUnit, BiocGenerics

License GPL (>=3.0)

URL https://github.com/datapplab/pathview, https://pathview.uncc.edu/

LazyLoad yes

LazyData yes

Config/pak/sysreqs make libpng-dev libxml2-dev libssl-dev zlib1g-dev

Repository Bioconductor 3.23

Date/Publication 2026-04-28 12:37:26 UTC

RemoteUrl https://github.com/bioc/pathview

RemoteRef RELEASE_3_23

RemoteSha 18f70b18cbfe57bec7ef629d6e21f63f155db044

Contents

pathview-package . . . 2
combineKEGGnodes . . . 3
cpd.accs . . . 4
cpdidmap . . . 5
demo.data . . . 6
download.kegg . . . 7
eg2id . . . 8
kegg.species.code . . . 10
korg . . . 11
mol.sum . . . 12
node.color . . . 14
node.info . . . 16
node.map . . . 18
pathview . . . 20
sim.mol.data . . . 26
wordwrap . . . 28

Index **30**

`pathview-package` *Pathway based data integration and visualization*

Description

Pathway based data integration and visualization

Details

Package:	pathview
Type:	Package
Version:	1.0
Date:	2012-12-26
License:	What license is it under?
LazyLoad:	yes

~~ An overview of how to use the package, including the most important ~~ ~~ functions ~~

Author(s)

Weijun Luo <luo_weijun@yahoo.com>

Maintainer: Weijun Luo <luo_weijun@yahoo.com>

References

Luo, W. and Brouwer, C., Pathview: an R/Bioconductor package for pathway based data integration and visualization. Bioinformatics, 2013, 29(14): 1830-1831, doi: 10.1093/bioinformatics/btt285

`combineKEGGnodes` *Special treatment of nodes or edges for KEGG pathway rendering*

Description

`combineKEGGnodes` combines nodes into a group in a KEGG pathway graph. `reaction2edge` converts reactions into edges in KEGG pathway graph.

Usage

```
combineKEGGnodes(nodes, graph, combo.node)
reaction2edge(path, gR)
```

Arguments

`nodes`	character, names of the names to be combined.
`graph`, `gR`	a object of "graphNEL" class, the graph parsed and converted from KEGG pathway.
`path`	a object of "KEGGPathway" class, the parsed KEGG pathway.
`combo.node`	character, the name of result combined node.

Details

`combineKEGGnodes` not only combines nodes in the graph object, but also corresponding node data in the KEGG pathway object. This function is needed for KEGG-defined group nodes and parsed enzyme groups involved in the same reaction. `reaction2edge` converts a reaction into 2 consecutive edges between substrate and enzyme and enzyme and product. This function is needed as to faithfully show the compound-enzyme nodes and their interactions in Graphviz-style view of KEGG pathway.

Value

The results returned by `combineKEGGnodes` is a combined graph of "graphNEL" class. The results returned by `reaction2edge` is a list of 3 elements: `gR`, the converted graph ("graphNEL"); edata.new, the new edge data ("KEGGEdge"); ndata.new, the new node data ("KEGGNode").

Author(s)

Weijun Luo <luo_weijun@yahoo.com>

References

Luo, W. and Brouwer, C., Pathview: an R/Bioconductor package for pathway based data integration and visualization. Bioinformatics, 2013, 29(14): 1830-1831, doi: 10.1093/bioinformatics/btt285

See Also

node.info the main parser function

---

cpd.accs *Mapping data between compound or gene IDs and KEGG accessions*

Description

Mapping data between compound or gene IDs and KEGG accessions

Usage

```
data(cpd.accs)
data(cpd.names)
data(kegg.met)
data(ko.ids)
data(rn.list)
data(gene.idtype.list)
data(gene.idtype.bods)
data(cpd.simtypes)
```

Format

cpd.accs is a data frame with 30054 observations on the following 4 variables. cpd.names is a data frame with 12314 observations on the following 5 variables. kegg.met is a character matrix of 694 rows and 3 columns. ko.ids is a character vector 8511 KEGG ortholog gene IDs, as used in KEGG ortholog pathways. rn.list is a namedlist of 21 vectors. Each vector records the row numbers for one of 21 dfferent compound ID types in cpd.accs data.frame. gene.idtype.list is a character vector of 13 common gene, transcript or protein ID types. Note some ID types are species specific, for example TAIR or ORF. gene.idtype.bods is a list of character vectors ofcommon gene, transcript or protein ID types for the 19 major research species in bods. Each element corresponds to a species. cpd.simtypes is a character vector of 7 common compound related ID types, each of them has over 1000 unique entries. Hence these ID types are good for generating simulation compound data.

Source

ftp://ftp.ebi.ac.uk/pub/databases/chebi/Flat_file_tab_delimited/

http://www.genome.jp/kegg-bin/get_htext?br08001.keg

Examples

```
data(cpd.accs)
data(rn.list)
names(rn.list)
cpd.accs[rn.list[[1]][1:4],]
lapply(rn.list[1:4], function(rn) cpd.accs[rn[1:4],])

data(kegg.met)
head(kegg.met)
```

cpdidmap *Mapping between compound IDs and KEGG accessions*

Description

These auxillary compound ID mappers connect KEGG compound/glycan/drug accessions to compound names/synonyms and other commonly used compound-related IDs.

Usage

```
cpdidmap(in.ids, in.type, out.type)
cpd2kegg(in.ids, in.type)
cpdkegg2name(in.ids, in.type = c("KEGG", "KEGG COMPOUND accession")[1])
cpdname2kegg(in.ids)
```

Arguments

in.ids	character, input IDs to be mapped.
in.type	character, the input ID type, needs to be either "KEGG" (including compound, glycan and durg) or one of the compound-related ID types used in CHEMBL database. For a full list of the CHEMBL IDs, do `data(rn.list); names(rn.list)`. For `cpdkegg2name`), default in.type = "KEGG".
out.type	character, the output ID type, needs to be either "KEGG" (including compound/glycan/durg) or one of the compound-related ID types used in CHEMBL database. For a full list of the CHEMBL IDs, do `data(rn.list); names(rn.list)`.

Details

character, the output ID type, needs to be either "KEGG" or one of the compound-related ID types used in CHEMBL database. For a full list of the CHEMBL IDs, do `data(rn.list); names(rn.list)`.

KEGG has its own compound ID system, including compound (glycan/durg) accessions. Therefore, all compound data need to be mapped to KEGG accessions when working with KEGG pathways. Function `cpd2kegg` does this mapping by calling `cpdname2kegg` or `cpdidmap`. On the other hand, we frequently want to check or show compound full names or other commonly used IDs instead of the less informative KEGG accessions when working with KEGG compound nodes, Functions `cpdkegg2name` and `cpdidmap` do this reverse mapping. These functions are written as part of the Pathview mapper module, they are equally useful for other compound ID or data mapping tasks. The use of these functions depends on a few data objects: "cpd.accs", "cpd.names", "keg.met" and "rn.list", which are included in this package. To access them, use `data()` function.

Value

a 2-column character matrix recording the mapping between input IDs to the target ID type.

Author(s)

Weijun Luo <luo_weijun@yahoo.com>

References

Luo, W. and Brouwer, C., Pathview: an R/Bioconductor package for pathway based data integration and visualization. Bioinformatics, 2013, 29(14): 1830-1831, doi: 10.1093/bioinformatics/btt285

See Also

eg2id and id2eg the auxillary gene ID mappers, mol.sum the auxillary molecular data mapper, node.map the node data mapper function.

Examples

```
data(cpd.simtypes)
#generate simulated compound data named with non-KEGG ("CAS Registry Number")IDs
cpd.cas <- sim.mol.data(mol.type = "cpd", id.type = cpd.simtypes[2],
    nmol = 10000)
#construct map between non-KEGG ID and KEGG ID ("KEGG COMPOUND accession")
id.map.cas <- cpdidmap(in.ids = names(cpd.cas), in.type = cpd.simtypes[2],
    out.type = "KEGG COMPOUND accession")
#Map molecular data onto standard KEGG IDs
cpd.kc <- mol.sum(mol.data = cpd.cas, id.map = id.map.cas)
#check the results
head(cpd.cas)
head(id.map.cas)
head(cpd.kc)

#map KEGG ID to compound name
cpd.names=cpdkegg2name(in.ids=id.map.cas[,2])
head(cpd.names)
```

---

demo.data *Data for demo purpose*

Description

demo.paths includes pathway ids and optimal plotting parameters when calling pathview.

GSE16873 is a breast cancer study (Emery et al, 2009) downloaded from Gene Expression Omnibus (GEO). Dataset gse16873 is pre-processed using FARMS method and includes 6 patient cases, each with HN (histologically normal) and DCIS (ductal carcinoma in situ) RMA samples. The same dataset is also used in gage package. Dataset gse16873.d includes the gene expression changes of two pairs of DCIS vs HN samples.

paths.hsa includes the full list of human pathway ID/names from KEGG.

Usage

```
data(demo.paths)
data(gse16873.d)
data(paths.hsa)
```

Format

demo.paths is a named list with ids and plotting parameters for 3 pathways. For details do:

`data(demo.paths); demo.paths`

gse16873.d is a numeric matrix with over 10000 rows (genes) and 2 columns (samples). For details do: `data(gse16873.d); str(gse16873.d)`.

paths.hsa is a named vector mapping KEGG pathway ID to human pathway names.

Source

http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE16873

---

download.kegg *Download KEGG pathway graphs and associated KGML data*

Description

This is the downloader function for KEGG pathways, automatically download graph images and associated KGML data.

Usage

```
download.kegg(pathway.id = "00010", species = "hsa", kegg.dir = ".",
file.type=c("xml", "png"))
```

Arguments

pathway.id	character, 5-digit KEGG pathway IDs. Default pathway.id="00010".
species	character, either the KEGG code, scientific name or the common name of the target species. When KEGG ortholog pathway is considered, species="ko". Default species="hsa", it is equivalent to use either "Homo sapiens" (scientific name) or "human" (common name).
kegg.dir	character, the directory of KEGG pathway data file (.xml) and image file (.png). Default kegg.dir="." (current working directory).
file.type	character, the file type(s) to be downloaded, either KEGG pathway data file (xml) or image file (png). Default include both types.

Details

Species can be specified as either kegg code, scientific name or the common name. Scientific name and the common name are always mapped to kegg code first. Length of species should be either 1 or the same as pathway.id, if not, the same set of pathway.id will be applied to all species.

Value

a named character vector, either "succeed" or "failed", indicating the download status of corresponding pathways.

Author(s)

Weijun Luo <luo_weijun@yahoo.com>

References

Luo, W. and Brouwer, C., Pathview: an R/Bioconductor package for pathway based data integration and visualization. Bioinformatics, 2013, 29(14): 1830-1831, doi: 10.1093/bioinformatics/btt285

See Also

pathview the main function, node.info the parser,

Examples

```
data(demo.paths)
sel.2paths=demo.paths$sel.paths[1:2]
download.kegg(pathway.id = sel.2paths, species = "hsa")
#pathway files should be downloaded into current working directory
```

---

eg2id *Mapping between different gene ID and annotation types*

Description

These auxillary gene ID mappers connect different gene ID or annotation types, especially they are used to map Entrez Gene ID to external gene, transcript or protein IDs or vise versa.

Usage

```
eg2id(eg, category = gene.idtype.list[1:2], org = "Hs", pkg.name = NULL,
...)
id2eg(ids, category = gene.idtype.list[1], org = "Hs", pkg.name = NULL, ...)
geneannot.map(in.ids, in.type, out.type, org="Hs", pkg.name=NULL,
unique.map=TRUE, na.rm=TRUE, keep.order=TRUE)
```

Arguments

eg	character, input Entrez Gene IDs.
ids	character, input gene/transcript/protein IDs to be converted to Entrez Gene IDs.
in.ids	character, input gene/transcript/protein IDs to be converted or mapped to other Gene IDs or annotation types.

category	character, for eg2id the output ID types to map from Entrez Gene, d to be c("SYMBOL", "GENENAME"); for id2eg, the input ID type to be mapped to Entrez Gene, default to be "SYMBOL".
in.type	character, the input gene/transcript/protein ID type to be mapped or converted to other ID/annotation types.
out.type	character, the output gene/transcript/protein ID type to be mapped or converted to other ID/annotation types.
org	character, the two-letter abbreviation of organism name, or KEGG species code, or the common species name, used to determine the gene annotation package. For all potential values check: data(bods); bods. Default org="Hs", and can also be "hsa" or "human" (case insensitive). Only effective when pkg.name is not NULL.
pkg.name	character, name of the gene annotation package. This package should be one of the standard annotation packages from Bioconductor, such as "org.Hs.eg.db". Check data(bods); bods for a full list of standard annotation packages. You may also use your custom annotation package built with AnnotationDbi, the Bioconductor Annotation Database Interface. Default pkg.name=NULL, hence argument org should be specified.
unique.map	logical, whether to combine multiple entries mapped to the same input ID as a single entry (separted by "; "). Default unique.map=TRUE.
na.rm	logical, whether to remove the lines where input ID is not mapped (NA for mapped entries). Default na.rm=TRUE.
keep.order	logical, whether to keep the original input order even with all unmapped input IDs. Default keep.order=TRUE.
...	other arguments to be passed to geneannot.map function.

Details

KEGG uses Entrez Gene ID as its standard gene ID. Therefore, all gene data need to be mapped to Entrez Genes when working with KEGG pathways. Function id2eg does this mapping. On the other hand, we frequently want to check or show gene symbols or full names instead of the less informative Entrez Gene ID when working with KEGG gene nodes, Function eg2id does this reverse mapping. Both id2eg and eg2id are wrapper functions of geneannot.map function. The latter can be used to map between a range of major gene/transcript/protein IDs or annotation types, not just Entrez Gene ID. These functions are written as part of the Pathview mapper module, they are equally useful for other gene ID or data mapping tasks. The use of these functions depends on gene annotation packages like "org.Hs.eg.db", which are Bioconductor standard. IFf no such packages not available for your interesting organisms, you may build one with Bioconductor AnnotationDbi package.

Value

a 2- or multi-column character matrix recording the mapping between input IDs to the target ID type(s).

Author(s)

Weijun Luo <luo_weijun@yahoo.com>

References

Luo, W. and Brouwer, C., Pathview: an R/Bioconductor package for pathway based data integration and visualization. Bioinformatics, 2013, 29(14): 1830-1831, doi: 10.1093/bioinformatics/btt285

See Also

cpd2kegg etc the auxillary compound ID mappers, mol.sum the auxillary molecular data mapper, node.map the node data mapper function.

Examples

```
data(gene.idtype.list)
#generate simulated gene data named with non-KEGG/Entrez gene IDs
gene.ensprot <- sim.mol.data(mol.type = "gene", id.type = gene.idtype.list[4],
    nmol = 50000)
#construct map between non-KEGG ID and KEGG ID (Entrez gene)
id.map.ensprot <- id2eg(ids = names(gene.ensprot),
    category = gene.idtype.list[4], org = "Hs")
#Map molecular data onto Entrez Gene IDs
gene.entrez <- mol.sum(mol.data = gene.ensprot, id.map = id.map.ensprot)
#check the results
head(gene.ensprot)
head(id.map.ensprot)
head(gene.entrez)

#map Entrez Gene to Gene Symbol and Name
eg.symbname=eg2id(eg=id.map.ensprot[,2])
#entries with more than 1 Entrez Genes are not mapped
head(eg.symbname)

#not run: map between other ID types for other species
#ath.tair=sim.mol.data(id.type="tair", species="ath", nmol=1000)
#data(gene.idtype.bods)
#gid.map <-geneannot.map(in.ids=names(ath.tair)[rep(1:100,each=2)],
#in.type="tair", out.type=gene.idtype.bods$ath[-1], org="At")
#gid.map1 <-geneannot.map(in.ids=names(ath.tair)[rep(1:100,each=2)],
#in.type="tair", out.type=gene.idtype.bods$ath[-1], org="At",
#unique.map=F, keep.order=F)
#str(gid.map)
#str(gid.map1)
```

---

kegg.species.code *Mapping species name to KEGG code*

Description

This function maps species name to KEGG code.

Usage

```
kegg.species.code(species = "hsa", na.rm = FALSE, code.only = TRUE)
```

Arguments

species	character, either the KEGG code, scientific name or the common name of the target species. Default species="hsa", it is equivalent to use either "Homo sapiens" (scientific name) or "human" (common name).
na.rm	logical, should unmapped entris be removed. Default na.rm = FALSE.
code.only	logical, whether to extract KEGG species code only or with gene ID usage info too. Default , code.only = TRUE.

Value

a character vector of mapped KEGG code of species.

Author(s)

Weijun Luo <luo_weijun@yahoo.com>

References

Luo, W. and Brouwer, C., Pathview: an R/Bioconductor package for pathway based data integration and visualization. Bioinformatics, 2013, 29(14): 1830-1831, doi: 10.1093/bioinformatics/btt285

See Also

korg the species and KEGG code mapping data, cpd2kegg etc the auxillary compound ID mappers, download.kegg the downloader function.

Examples

```
species=c("ptr", "Mus musculus", "dog", "happ")
kcode=kegg.species.code(species = species, na.rm = FALSE)
print(kcode)
```

---

korg *Mapping data on KEGG species code and corresponding Bioconductor gene annotation package*

Description

Data on KEGG species, including taxonomy IDs, KEGG code, scientific name, common name, corresponding gene ID types, and gene annotation package names in Bioconductor

Usage

```
data(korg)
data(bods)
```

Format

korg is a character matrix of ~4800 rows and 10 columns. First 5 columns are KEGG and NCBI taxonomy IDs, KEGG species code, scientific name and common name, followed columns on gene ID types used for each species: entrez.gnodes ("1" or "0", whether EntrezGene is the default gene ID) and representative KEGG gene ID, NCBI or Entrez Gene ID, NCBI protein and Uniprot ID. Note korg includes 4800 KEGG species (as of 06/2017), in the meantime, an updated version of korg is now checked out from Pathview Web server each time pathview package is loaded.

bods is a character matrix of 19 rows and 3 columns on the mapping between gene annotation package names in Bioconductor, common name and KEGG code of most common research species.

Source

http://www.genome.jp/kegg-bin/get_htext?br08601.keg

http://bioconductor.org/packages/release/BiocViews.html#___OrgDb

Examples

```
data(korg)
data(bods)
head(korg)
head(bods)
```

---

mol.sum *Mapping and summation of molecular data onto standard IDs*

Description

Molecular data like gene or metabolite data are frequently annotated by various types of IDs. This function maps and summarize molecular data onto standard gene or compound IDs. It would be straightforward to integrate, analyze or visualize the "standardized" data with pathways or functional categories.

Usage

```
mol.sum(mol.data, id.map, gene.annotpkg = "org.Hs.eg.db", sum.method =
c("sum", "mean", "median", "max", "max.abs", "random")[1])
```

Arguments

mol.data	Either vector (single sample) or a matrix-like data (multiple sample). Vector should be numeric with molecule IDs as names or it may also be character of molecule IDs. Character vector is treated as discrete or count data. Matrix-like data structure has molecules as rows and samples as columns. Row names should be molecule IDs. Default mol.data=NULL. This argument is equivalent to gene.data or cpd.data in the pathview function. Check pahtview function for more information.
id.map	a two-column character matrix, giving the mapping between molecular IDs used in mol.data and taget/standard molecular IDs. Then mol.data are gene data, id.map may also be a character specifying the type of IDs used in mol.data. The two-column mapping matrix will be generated automatically.
gene.annotpkg	character, name of the gene annotation package. This package should be one of the standard annotation packages from Bioconductor, such as "org.Hs.eg.db" (default). Check data(bods); bods for a full list of standard annotation packages. You may also use your custom annotation package built with AnnotationDbi, the Bioconductor Annotation Database Interface. Only effective when mol.data are gene.data and id.map gives the ID type being used.
sum.method	character, the method name to calculate node summary given that multiple genes or compounds are mapped to it. Poential options include "sum","mean", "median", "max", "max.abs" and "random". Default sum.method="sum".

Details

This function is called in pathview main function when gene.idtype or cpd.idtype is not the standard type, so that the molecular data can be mapped and summarized onto standard IDs. This is needed for further mapping to KEGG pathways. The same standard ID mapping is needed when carry out pathway or functional analysis on molecular data, which are labeled by non-standard (or alien) IDs or probe names, like in most of the microarray or metabolomics datasets. In other words, function mol.sum can be useful in all these situations.

Value

a numeric vector or matrix. Its dimensionality is the same as the input mol.data except row names are standard molecular IDs.

Author(s)

Weijun Luo <luo_weijun@yahoo.com>

References

Luo, W. and Brouwer, C., Pathview: an R/Bioconductor package for pathway based data integration and visualization. Bioinformatics, 2013, 29(14): 1830-1831, doi: 10.1093/bioinformatics/btt285

See Also

node.map the node data mapper function. id2eg, cpd2kegg etc the auxillary molecular ID mappers, pathview the main function,

Examples

```
data(gene.idtype.list)
#generate simulated gene data named with non-KEGG/Entrez gene IDs
gene.ensprot <- sim.mol.data(mol.type = "gene", id.type = gene.idtype.list[4],
    nmol = 50000)
#construct map between non-KEGG ID and KEGG ID (Entrez gene)
id.map.ensprot <- id2eg(ids = names(gene.ensprot),
    category = gene.idtype.list[4], org = "Hs")
#Map molecular data onto Entrez Gene IDs
gene.entrez <- mol.sum(mol.data = gene.ensprot, id.map = id.map.ensprot)
#check the results
head(gene.ensprot)
head(id.map.ensprot)
head(gene.entrez)
```

---

node.color *Code molecular data as pseudo colors on the pathway graph*

Description

node.color converts the mapped molecular (gene, protein or metabolite etc) data as pseudo colors on pathway nodes. col.key draws color key(s) for mapped molecular data on the pathway graph.

Usage

```
node.color(plot.data = NULL, discrete=FALSE, limit, bins, both.dirs =
TRUE, low = "green", mid = "gray", high = "red", na.col = "transparent",
trans.fun = NULL)
col.key(discrete=FALSE, limit = 1, bins = 10, cols = NULL, both.dirs =
TRUE, low = "green", mid = "gray", high = "red", graph.size, node.size,
size.by.graph = TRUE, key.pos = "topright", off.sets = c(x = 0, y = 0),
align = "n", cex = 1, lwd = 1)
```

Arguments

plot.data	the result returned by node.map function. It is a data.frame composed of parsed KGML data and summary molecular data for each mapped node. Rows are mapped nodes, and columns are parsed or mapped node data. Check node.map for details.
discrete	logical, whether to treat the molecular data or node summary data as discrete. d discrete=FALSE, otherwise, mol.data will be a charactor vector of molecular IDs.
limit	a list of two numeric elements with "gene" and "cpd" as the names. This argument specifies the limit values for gene.data and cpd.data when converting them to pseudo colors. Each element of the list could be of length 1 or 2. Length 1 suggests discrete data or 1 directional (positive-valued) data, or the absolute limit for 2 directional data. Length 2 suggests 2 directional data. Default limit=list(gene=0.5, cpd=1).

bins	a list of two integer elements with "gene" and "cpd" as the names. This argument specifies the number of levels or bins for gene.data and cpd.data when converting them to pseudo colors. Default limit=list(gene=10, cpd=10).
both.dirs	a list of two logical elements with "gene" and "cpd" as the names. This argument specifies whether gene.data and cpd.data are 1 directional or 2 directional data when converting them to pseudo colors. Default limit=list(gene=TRUE, cpd=TRUE).
trans.fun	a list of two function (not character) elements with "gene" and "cpd" as the names. This argument specifies whether and how gene.data and cpd.data are transformed. Examples are log, abs or users' own functions. Default limit=list(gene=NULL, cpd=NULL).
low, mid, high	each is a list of two colors with "gene" and "cpd" as the names. This argument specifies the color spectra to code gene.data and cpd.data. When data are 1 directional (TRUE value in both.dirs), only mid and high are used to specify the color spectra. Default spectra (low-mid-high) "green"-"gray"-"red" and "blue"-"gray"-"yellow" are used for gene.data and cpd.data respectively. The values for 'low, mid, high' can be given as color names ('red'), plot color index (2=red), and HTML-style RGB, ("\#FF0000"=red).
na.col	color used for NA's or missing values in gene.data and cpd.data. d na.col="transparent".
cols	character, specifying a discrete spectrum of colors to be plotted as color key. Note this argument is usually NULL (default), otherwise, the number of discrete colors has to match bins.
graph.size	numeric vector of length 2, i.e. the sizes (width, height) of the pathway graph panel. This is needed to determine the sizes and exact location of the color key.
node.size	numeric vector of length 2, i.e. the sizes (width, height) of the standard gene nodes (rectangles). This is needed to determine the sizes and exact location of the color key when size.by.graph=FALSE.
size.by.graph	logical, whether to determine the sizes and exact location of the color key with respect to the size of the whole graph panel or that of a single node. Default size.by.graph=TRUE.
key.pos	character, controlling the position of color key(s). Potentail values are "bottomleft", "bottomright", "topleft" and "topright". d key.pos="topright".
off.sets	numeric vector of length 2, with "x" and "y" as the names. This argument specifies the offset values in x and y axes when plotting a new color key, as to avoid overlap with existing color keys or boundaries. Note that the off.sets value is reset and returned each time col.key function is called, as for the reference of plotting the next color key. Default off.sets=c(0,0).
align	character, controlling how the color keys are aligned when needed. Potential values are "x", aligned by x coordinates, and "y", aligned by y coordinates. Default align="x".
cex	A numerical value giving the amount by which legend text and symbols should be scaled relative to the default 1.
lwd	numeric, the line width, a _positive_ number, defaulting to '1'.

Details

node.color converts the mapped molecular data (gene.data or cpd.data) by node.map function into pseudo colors, which then can be plotted on the pathway graph. col.key is used in combination with node.color in pathview, although this function can be used independently for similar tasks.

Value

node.color returns a vector or matrix of colors. Its dimensionality is the same as the corresponding gene.data or cpd.data. col.key plots a color key on existing pathway graph, then returns a updated version of off.sets for the reference of next color key.

Author(s)

Weijun Luo <luo_weijun@yahoo.com>

References

Luo, W. and Brouwer, C., Pathview: an R/Bioconductor package for pathway based data integration and visualization. Bioinformatics, 2013, 29(14): 1830-1831, doi: 10.1093/bioinformatics/btt285

See Also

keggview.native and keggview.graph the viwer functions, node.map the node data mapper function.

Examples

```
xml.file=system.file("extdata", "hsa04110.xml", package = "pathview")
node.data=node.info(xml.file)
names(node.data)
data(gse16873.d)
plot.data.gene=node.map(mol.data=gse16873.d[,1], node.data,
  node.types="gene")
head(plot.data.gene)
cols.ts.gene=node.color(plot.data.gene, limit=1, bins=10)
head(cols.ts.gene)
```

---

node.info *Extract node information from KEGG pathway*

Description

The parser function, parser KGML file and/or extract node information from KEGG pathway.

Usage

```
node.info(object, short.name = TRUE)
```

Arguments

object	either a character specifying the full KGML file name (with directory), or a object of "KEGGPathway" class, or a object of "graphNEL" class. The latter two are parsed results of KGML file.
short.name	logical, if TRUE, the short labels, i.e. the first iterm separated by "," in the long labels are parsed out as node labels. Default short.name=TRUE.

Details

Parser function node.info extract node data from parsed KEGG pathways. KGML files are parsed using `parseKGML2` and `KEGGpathway2Graph2`. These functions from KEGGgraph package have been heavily modified for reaction parsing and conversion to edges.

Value

a named list of 10 elements: "kegg.names", "type", "component", "size", "labels", "shape", "x", "y", "width" and "height". Each elements record the corresponding attribute for all nodes in the parsed KEGG pathway.

Author(s)

Weijun Luo <luo_weijun@yahoo.com>

References

Luo, W. and Brouwer, C., Pathview: an R/Bioconductor package for pathway based data integration and visualization. Bioinformatics, 2013, 29(14): 1830-1831, doi: 10.1093/bioinformatics/btt285

See Also

`pathview` the main function, `combineKEGGnodes` and `reaction2edge` for special treatment of nodes or edges.

Examples

```
xml.file=system.file("extdata", "hsa04110.xml", package = "pathview")
node.data=node.info(xml.file)
names(node.data)
#or parse into a graph object, then extract node info
gR1=pathview:::parseKGML2Graph2(xml.file, genesOnly=FALSE, expand=FALSE, split.group=FALSE)
node.data=node.info(gR1)
```

node.map *Map molecular data onto KEGG pathway nodes*

Description

The mapper function, mapping molecular data(gene expression, metabolite abundance etc)to nodes in KEGG pathway.

Usage

```
node.map(mol.data = NULL, node.data, node.types = c("gene", "ortholog",
"compound")[1], node.sum = c("sum", "mean", "median", "max", "max.abs",
"random")[1], entrez.gnodes=TRUE)
```

Arguments

mol.data	Either vector (single sample) or a matrix-like data (multiple sample). Vector should be numeric with molecule IDs as names or it may also be character of molecule IDs. Character vector is treated as discrete or count data. Matrix-like data structure has molecules as rows and samples as columns. Row names should be molecule IDs. Default mol.data=NULL. This argument is equivalent to gene.data or cpd.data in the pathview function. Check pahtview function for more information.
node.data	a named list of 10 elements, the results returned by node.info, check the function for details.
node.types	character, sepcify the node type to map the mol.data to, either "gene", "compound", or "compound". Default node.types="gene".
node.sum	character, the method name to calculate node summary given that multiple genes or compounds are mapped to it. Poential options include "sum","mean", "median", "max", "max.abs" and "random". Default node.sum="sum".
entrez.gnodes	logical, whether EntrezGene (NCBI GeneID) is used as the default gene ID in the KEGG data files. This is needed because KEGG uses different types default gene ID for different species. Some most common model species use EntrezGene, but majority of others use Locus tag. Default entrez.gnodes=TRUE.

Details

Mapper function node.map maps user supplied molecular data to KEGG pathways. This function takes standard KEGG molecular IDs (Entrez Gene ID or KEGG Compound Accession) and map them to pathway nodes. None KEGG molecular gene IDs or Compound IDs are pre-mapped to standard KEGG IDs by calling another function mol.sum. When multiple molecules map to one node, the corresponding molecular data are summarized into a single node summary by calling function specified by node.sum. This mapped node summary data together with the parsed KGML data are then returned for further processing. Proper input data include: gene expression, protein expression, genetic association, metabolite abundance, genomic data, literature, and other data types mappable to pathways. The input mol.data may be NULL, then no molecular data are actually

mapped, but all nodes of the specified node.type are considered "mappable" and their parsed KGML data returned.

Value

A data.frame composed of parsed KGML data and summary molecular data for each mapped node. Each row is a mapped node, and columns are:

`kegg.names`	standard KEGG IDs/Names for mapped nodes. It's Entrez Gene ID or KEGG Compound Accessions.
`labels`	Node labels to be used when needed
`type`	node type, currently 4 types are supported: "gene","enzyme", "compound" and "ortholog".
`x`	x coordinate in the original KEGG pathway graph.
`y`	y coordinate in the original KEGG pathway graph.
`width`	node width in the original KEGG pathway graph.
`height`	node height in the original KEGG pathway graph.
`other columns`	columns of the mapped gene/compound data

Author(s)

Weijun Luo <luo_weijun@yahoo.com>

References

Luo, W. and Brouwer, C., Pathview: an R/Bioconductor package for pathway based data integration and visualization. Bioinformatics, 2013, 29(14): 1830-1831, doi: 10.1093/bioinformatics/btt285

See Also

`mol.sum` the auxillary molecular data mapper, `id2eg`, `cpd2kegg` etc the auxillary molecular ID mappers, `node.color` the node color coder, `pathview` the main function, `node.info` the parser.

Examples

```
xml.file=system.file("extdata", "hsa04110.xml", package = "pathview")
node.data=node.info(xml.file)
names(node.data)
data(gse16873.d)
plot.data.gene=node.map(mol.data=gse16873.d[,1], node.data,
  node.types="gene")
head(plot.data.gene)
```

pathview *Pathway based data integration and visualization*

Description

Pathview is a tool set for pathway based data integration and visualization. It maps and renders user data on relevant pathway graphs. All users need is to supply their gene or compound data and specify the target pathway. Pathview automatically downloads the pathway graph data, parses the data file, maps user data to the pathway, and render pathway graph with the mapped data. Pathview generates both native KEGG view and Graphviz views for pathways. keggview.native and keggview.graph are the two viewer functions, and pathview is the main function providing a unified interface to downloader, parser, mapper and viewer functions.

Usage

```
pathview(gene.data = NULL, cpd.data = NULL, pathway.id,
species = "hsa", kegg.dir = ".", cpd.idtype = "kegg", gene.idtype =
"entrez", gene.annotpkg = NULL, min.nnodes = 3, kegg.native = TRUE,
map.null = TRUE, expand.node = FALSE, split.group = FALSE, map.symbol =
TRUE, map.cpdname = TRUE, node.sum = "sum", discrete=list(gene=FALSE,
cpd=FALSE), limit = list(gene = 1, cpd = 1), bins = list(gene = 10, cpd
= 10), both.dirs = list(gene = T, cpd = T), trans.fun = list(gene =
NULL, cpd = NULL), low = list(gene = "green", cpd = "blue"), mid =
list(gene = "gray", cpd = "gray"), high = list(gene = "red", cpd =
"yellow"), na.col = "transparent", ...)

keggview.native(plot.data.gene = NULL, plot.data.cpd = NULL,
cols.ts.gene = NULL, cols.ts.cpd = NULL, node.data, pathway.name,
out.suffix = "pathview", kegg.dir = ".", multi.state=TRUE, match.data =
TRUE, same.layer = TRUE, res = 300, cex = 0.25, discrete =
list(gene=FALSE, cpd=FALSE), limit= list(gene = 1, cpd = 1), bins =
list(gene = 10, cpd = 10), both.dirs =list(gene = T, cpd = T), low =
list(gene = "green", cpd = "blue"), mid = list(gene = "gray", cpd =
"gray"), high = list(gene = "red", cpd = "yellow"), na.col =
"transparent", new.signature = TRUE, plot.col.key = TRUE, key.align =
"x", key.pos = "topright", ...)

keggview.graph(plot.data.gene = NULL, plot.data.cpd = NULL, cols.ts.gene
= NULL, cols.ts.cpd = NULL, node.data, path.graph, pathway.name,
out.suffix = "pathview", pdf.size = c(7, 7), multi.state=TRUE,
same.layer = TRUE, match.data = TRUE, rankdir = c("LR", "TB")[1],
is.signal = TRUE, split.group = F, afactor = 1, text.width = 15, cex =
0.5, map.cpdname = FALSE, cpd.lab.offset = 1.0,
discrete=list(gene=FALSE, cpd=FALSE), limit = list(gene = 1, cpd = 1),
bins = list(gene = 10, cpd = 10), both.dirs = list(gene = T, cpd = T),
low = list(gene = "green", cpd = "blue"), mid = list(gene = "gray", cpd
= "gray"), high = list(gene = "red", cpd = "yellow"), na.col =
```

```
"transparent", new.signature = TRUE, plot.col.key = TRUE, key.align =
"x", key.pos = "topright", sign.pos = "bottomright", ...)
```

Arguments

gene.data	either vector (single sample) or a matrix-like data (multiple sample). Vector should be numeric with gene IDs as names or it may also be character of gene IDs. Character vector is treated as discrete or count data. Matrix-like data structure has genes as rows and samples as columns. Row names should be gene IDs. Here gene ID is a generic concepts, including multiple types of gene, transcript and protein uniquely mappable to KEGG gene IDs. KEGG ortholog IDs are also treated as gene IDs as to handle metagenomic data. Check details for mappable ID types. Default gene.data=NULL. numeric, character, continuous
cpd.data	the same as gene.data, excpet named with IDs mappable to KEGG compound IDs. Over 20 types of IDs included in CHEMBL database can be used here. Check details for mappable ID types. Default cpd.data=NULL. Note that gene.data and cpd.data can't be NULL simultaneously.
pathway.id	character vector, the KEGG pathway ID(s), usually 5 digit, may also include the 3 letter KEGG species code.
species	character, either the kegg code, scientific name or the common name of the target species. This applies to both pathway and gene.data or cpd.data. When KEGG ortholog pathway is considered, species="ko". Default species="hsa", it is equivalent to use either "Homo sapiens" (scientific name) or "human" (common name).
kegg.dir	character, the directory of KEGG pathway data file (.xml) and image file (.png). Users may supply their own data files in the same format and naming convention of KEGG's (species code + pathway id, e.g. hsa04110.xml, hsa04110.png etc) in this directory. Default kegg.dir="." (current working directory).
cpd.idtype	character, ID type used for the cpd.data. Default cpd.idtype="kegg" (include compound, glycan and drug accessions).
gene.idtype	character, ID type used for the gene.data, case insensitive. Default gene.idtype="entrez", i.e. Entrez Gene, which are the primary KEGG gene ID for many common model organisms. For other species, gene.idtype should be set to "KEGG" as KEGG use other types of gene IDs. For the common model organisms (to check the list, do: `data(bods); bods`), you may also specify other types of valid IDs. To check the ID list, do: `data(gene.idtype.list); gene.idtype.list`.
gene.annotpkg	character, the name of the annotation package to use for mapping between other gene ID types including symbols and Entrez gene ID. Default gene.annotpkg=NULL.
min.nnodes	integer, minimal number of nodes of type "gene","enzyme", "compound" or "ortholog" for a pathway to be considered. Default min.nnodes=3.
kegg.native	logical, whether to render pathway graph as native KEGG graph (.png) or using graphviz layout engine (.pdf). Default kegg.native=TRUE.
map.null	logical, whether to map the NULL gene.data or cpd.data to pathway. When NULL data are mapped, the gene or compound nodes in the pathway will be rendered as actually mapped nodes, except with NA-valued color. When NULL

data are not mapped, the nodes are rendered as unmapped nodes. This argument mainly affects native KEGG graph view, i.e. when kegg.native=TRUE. Default map.null=TRUE.

`expand.node` logical, whether the multiple-gene nodes are expanded into single-gene nodes. Each expanded single-gene nodes inherits all edges from the original multiple-gene node. This option only affects graphviz graph view, i.e. when kegg.native=FALSE. This option is not effective for most metabolic pathways where it conflits with converting reactions to edges. Default expand.node=FLASE.

`split.group` logical, whether split node groups are split to individual nodes. Each split member nodes inherits all edges from the node group. This option only affects graphviz graph view, i.e. when kegg.native=FALSE. This option also effects most metabolic pathways even without group nodes defined orginally. For these pathways, genes involved in the same reaction are grouped automatically when converting reactions to edges unless split.group=TRUE. d split.group=FLASE.

`map.symbol` logical, whether map gene IDs to symbols for gene node labels or use the graphic name from the KGML file. This option is only effective for kegg.native=FALSE or same.layer=FALSE when kegg.native=TRUE. For same.layer=TRUE when kegg.native=TRUE, the native KEGG labels will be kept. Default map.symbol=TRUE.

`map.cpdname` logical, whether map compound IDs to formal names for compound node labels or use the graphic name from the KGML file (KEGG compound accessions). This option is only effective for kegg.native=FALSE. When kegg.native=TRUE, the native KEGG labels will be kept. Default map.cpdname=TRUE.

`node.sum` character, the method name to calculate node summary given that multiple genes or compounds are mapped to it. Poential options include "sum","mean", "median", "max", "max.abs" and "random". Default node.sum="sum".

`discrete` a list of two logical elements with "gene" and "cpd" as the names. This argument tells whether gene.data or cpd.data should be treated as discrete. Default dsicrete=list(gene=FALSE, cpd=FALSE), i.e. both data should be treated as continuous.

`limit` a list of two numeric elements with "gene" and "cpd" as the names. This argument specifies the limit values for gene.data and cpd.data when converting them to pseudo colors. Each element of the list could be of length 1 or 2. Length 1 suggests discrete data or 1 directional (positive-valued) data, or the absolute limit for 2 directional data. Length 2 suggests 2 directional data. Default limit=list(gene=1, cpd=1).

`bins` a list of two integer elements with "gene" and "cpd" as the names. This argument specifies the number of levels or bins for gene.data and cpd.data when converting them to pseudo colors. Default limit=list(gene=10, cpd=10).

`both.dirs` a list of two logical elements with "gene" and "cpd" as the names. This argument specifies whether gene.data and cpd.data are 1 directional or 2 directional data when converting them to pseudo colors. Default limit=list(gene=TRUE, cpd=TRUE).

`trans.fun` a list of two function (not character) elements with "gene" and "cpd" as the names. This argument specifies whether and how gene.data and cpd.data are transformed. Examples are `log`, `abs` or users' own functions. Default limit=list(gene=NULL, cpd=NULL).

`low, mid, high` each is a list of two colors with "gene" and "cpd" as the names. This argument specifies the color spectra to code gene.data and cpd.data. When data are 1 directional (TRUE value in both.dirs), only mid and high are used to specify the color spectra. Default spectra (low-mid-high) "green"-"gray"-"red" and "blue"-"gray"-"yellow" are used for gene.data and cpd.data respectively. The values for 'low, mid, high' can be given as color names ('red'), plot color index (2=red), and HTML-style RGB, ("\#FF0000"=red).

`na.col` color used for NA's or missing values in gene.data and cpd.data. d na.col="transparent".

`...` extra arguments passed to keggview.native or keggview.graph function.

special arguments for keggview.native or keggview.graph function.

`plot.data.gene` data.frame returned by node.map function for rendering mapped gene nodes, including node name, type, positions (x, y), sizes (width, height), and mapped gene.data. This data is also used as input for pseduo-color coding through node.color function. Default plot.data.gene=NULL.

`plot.data.cpd` same as plot.data.gene function, except for mapped compound node data. d plot.data.cpd=NULL. Default plot.data.cpd=NULL. Note that plot.data.gene and plot.data.cpd can't be NULL simultaneously.

`cols.ts.gene` vector or matrix of colors returned by node.color function for rendering gene.data. Dimensionality is the same as the latter. Default cols.ts.gene=NULL.

`cols.ts.cpd` same as cols.ts.gene, except corresponding to cpd.data. d cols.ts.cpd=NULL. Note that cols.ts.gene and cols.ts.cpd plot.data.gene can't be NULL simultaneously.

`node.data` list returned by node.info function, which parse KGML file directly or indirectly, and extract the node data.

`pathway.name` character, the full KEGG pathway name in the format of 3-letter species code with 5-digit pathway id, eg "hsa04612".

`out.suffix` character, the suffix to be added after the pathway name as part of the output graph file. Sample names or column names of the gene.data or cpd.data are also added when there are multiple samples. Default out.suffix="pathview".

`multi.state` logical, whether multiple states (samples or columns) gene.data or cpd.data should be integrated and plotted in the same graph. Default match.data=TRUE. In other words, gene or compound nodes will be sliced into multiple pieces corresponding to the number of states in the data.

`match.data` logical, whether the samples of gene.data and cpd.data are paired. Default match.data=TRUE. When let sample sizes of gene.data and cpd.data be m and n, when m>n, extra columns of NA's (mapped to no color) will be added to cpd.data as to make the sample size the same. This will result in the smae number of slice in gene nodes and compound when multi.state=TRUE.

`same.layer` logical, control plotting layers: 1) if node colors be plotted in the same layer as the pathway graph when kegg.native=TRUE, 2) if edge/node type legend be plotted in the same page when kegg.native=FALSE.

`res` The nominal resolution in ppi which will be recorded in the bitmap file, if a positive integer. Also used for 'units' other than the default, and to convert points to pixels. This argument is only effective when kegg.native=TRUE. Default res=300.

cex	A numerical value giving the amount by which plotting text and symbols should be scaled relative to the default 1. Default cex=0.25 when kegg.native=TRUE, cex=0.5 when kegg.native=FALSE.
new.signature	logical, whether pathview signature is added to the pathway graphs. Default new.signature=TRUE.
plot.col.key	logical, whether color key is added to the pathway graphs. Default plot.col.key= TRUE.
key.align	character, controlling how the color keys are aligned when both gene.data and cpd.data are not NULL. Potential values are "x", aligned by x coordinates, and "y", aligned by y coordinates. Default key.align="x".
key.pos	character, controlling the position of color key(s). Potentail values are "bottomleft", "bottomright", "topleft" and "topright". d key.pos="topright".
sign.pos	character, controlling the position of pathview signature. Only effective when kegg.native=FALSE, Signature position is fixed in place of the original KEGG signature when kegg.native=TRUE. Potentail values are "bottomleft", "bottomright", "topleft" and "topright". d sign.pos="bottomright".
path.graph	a graph object parsed from KGML file, only effective when kegg.native=FALSE.
pdf.size	a numeric vector of length 2, giving the width and height of the pathway graph pdf file. Note that pdf width increase by half when same.layer=TRUE to accommodate legends. Only effective when kegg.native=FALSE. Default pdf.size=c(7,7).
rankdir	character, either "LR" (left to right) or "TB" (top to bottom), specifying the pathway graph layout direction. Only effective when kegg.native=FALSE. Default rank.dir="LR".
is.signal	logical, if the pathway is treated as a signaling pathway, where all the unconnected nodes are dropped. This argument also affect the graph layout type, i.e. "dot" for signals or "neato" otherwise. Only effective when kegg.native=FALSE. Default is.signal=TRUE.
afactor	numeric, node amplifying factor. This argument is for node size fine-tuning, its effect is subtler than expected. Only effective when kegg.native=FALSE. Default afctor=1.
text.width	numeric, specifying the line width for text wrap. Only effective when kegg.native= FALSE. Default text.width=15 (characters).
cpd.lab.offset	numeric, specifying how much compound labels should be put above the default position or node center. This argument is useful when map.cpdname=TRUE, i.e. compounds are labelled by full name, which affects the look of compound nodes and color. Only effective when kegg.native=FALSE. Default cpd.lab.offset=1.0.

Details

Pathview maps and renders user data on relevant pathway graphs. Pathview is a stand alone program for pathway based data integration and visualization. It also seamlessly integrates with pathway and functional analysis tools for large-scale and fully automated analysis. Pathview provides strong support for data Integration. It works with: 1) essentially all types of biological data mappable to pathways, 2) over 10 types of gene or protein IDs, and 20 types of compound or metabolite IDs, 3) pathways for over 2000 species as well as KEGG orthology, 4) varoius data attributes and formats,

i.e. continuous/discrete data, matrices/vectors, single/multiple samples etc. To see mappable external gene/protein IDs do: `data(gene.idtype.list)`, to see mappable external compound related IDs do: `data(rn.list)`; names(rn.list). Pathview generates both native KEGG view and Graphviz views for pathways. Currently only KEGG pathways are implemented. Hopefully, pathways from Reactome, NCI and other databases will be supported in the future.

Value

From viersion 1.9.3, pathview can accept either a single pathway or multiple pathway ids. The result returned by pathview function is a named list corresponding to the input pathway ids. Each element (for each pathway itself is a named list, with 2 elements ("plot.data.gene", "plot.data.cpd"). Both elements are data.frame or NULL depends on the corresponding input data gene.data and cpd.data. These data.frames record the plot data for mapped gene or compound nodes: rows are mapped genes/compounds, columns are:

`kegg.names`	standard KEGG IDs/Names for mapped nodes. It's Entrez Gene ID or KEGG Compound Accessions.
`labels`	Node labels to be used when needed.
`all.mapped`	All molecule (gene or compound) IDs mapped to this node.
`type`	node type, currently 4 types are supported: "gene","enzyme", "compound" and "ortholog".
`x`	x coordinate in the original KEGG pathway graph.
`y`	y coordinate in the original KEGG pathway graph.
`width`	node width in the original KEGG pathway graph.
`height`	node height in the original KEGG pathway graph.
`other columns`	columns of the mapped gene/compound data and corresponding pseudo-color codes for individual samples

The results returned by `keggview.native` and codekeggview.graph are both a list of graph plotting parameters. These are not intended to be used externally.

Author(s)

Weijun Luo <luo_weijun@yahoo.com>

References

Luo, W. and Brouwer, C., Pathview: an R/Bioconductor package for pathway based data integration and visualization. Bioinformatics, 2013, 29(14): 1830-1831, doi: 10.1093/bioinformatics/btt285

See Also

`download.kegg` the downloader, `node.info` the parser, `node.map` and `node.color` the mapper.

Examples

```
#load data
data(gse16873.d)
data(demo.paths)

#KEGG view: gene data only
i <- 1
pv.out <- pathview(gene.data = gse16873.d[, 1], pathway.id =
demo.paths$sel.paths[i], species = "hsa", out.suffix = "gse16873",
kegg.native = TRUE)
str(pv.out)
head(pv.out$plot.data.gene)
#result PNG file in current directory

#Graphviz view: gene data only
pv.out <- pathview(gene.data = gse16873.d[, 1], pathway.id =
demo.paths$sel.paths[i], species = "hsa", out.suffix = "gse16873",
kegg.native = FALSE, sign.pos = demo.paths$spos[i])
#result PDF file in current directory

#KEGG view: both gene and compound data
sim.cpd.data=sim.mol.data(mol.type="cpd", nmol=3000)
i <- 3
print(demo.paths$sel.paths[i])
pv.out <- pathview(gene.data = gse16873.d[, 1], cpd.data = sim.cpd.data,
pathway.id = demo.paths$sel.paths[i], species = "hsa", out.suffix =
"gse16873.cpd", keys.align = "y", kegg.native = TRUE, key.pos = demo.paths$kpos1[i])
str(pv.out)
head(pv.out$plot.data.cpd)

#multiple states in one graph
set.seed(10)
sim.cpd.data2 = matrix(sample(sim.cpd.data, 18000,
    replace = TRUE), ncol = 6)
pv.out <- pathview(gene.data = gse16873.d[, 1:3],
    cpd.data = sim.cpd.data2[, 1:2], pathway.id = demo.paths$sel.paths[i],
    species = "hsa", out.suffix = "gse16873.cpd.3-2s", keys.align = "y",
    kegg.native = TRUE, match.data = FALSE, multi.state = TRUE, same.layer = TRUE)
str(pv.out)
head(pv.out$plot.data.cpd)

#result PNG file in current directory

##more examples of pathview usages are shown in the vignette.
```

`sim.mol.data` *Simulate molecular data for pathview experiment*

Description

The molecular data simulator generates either gene.data or cpd.data of different ID types, molecule numbers, sample sizes, either continuous or discrete.

Usage

```
sim.mol.data(mol.type = c("gene", "gene.ko", "cpd")[1], id.type = NULL,
species="hsa", discrete = FALSE, nmol = 1000, nexp = 1, rand.seed=100)
```

Arguments

mol.type	character of length 1, specifing the molecular type, either "gene" (including transcripts, proteins), or "gene.ko" (KEGG ortholog genes, as defined in KEGG ortholog pathways), or "cpd" (including metabolites, glycans, drugs). Note that KEGG ortholog gene are considered "gene" in function pathview. Default mol.type="gene".
id.type	character of length 1, the molecular ID type. When mol.type="gene", proper ID types include "KEGG" and "ENTREZ" (Entrez Gene). Multiple other ID types are also valid When species is among 19 major species fully annotated in Bioconductor, e.g. "hsa" (human), "mmu" (mouse) etc, check: data(gene.idtype.bods); gene.idtype.bods for other valid ID types. When mol.type="cpd", check data(cpd.simtypes); cpd.simtypes for valid ID types. Default id.type=NULL, then "Entrez" and "KEGG COMPOUND accession" will be assumed for mol.type = "gene" or "cpd".
species	character, either the kegg code, scientific name or the common name of the target species. This is only effective when mol.type = "gene". Setting species="ko" is equilvalent to mol.type="gene.ko". Default species="hsa", equivalent to either "Homo sapiens" (scientific name) or "human" (common name). Gene data id.type has multiple other choices for 19 major research species, for details do: data(gene.idtype.bods); gene.idtype.bods. When other species are specified, gene id.type is limited to "KEGG" and "ENTREZ".
discrete	logical, whether to generate discrete or continuous data. d discrete=FALSE, otherwise, mol.data will be a charactor vector of molecular IDs.
nmol	integer, the target number of different molecules. Note that the specified id.type may not have as many different IDs as nmol. In this case, all IDs of id.type are used.
nexp	integer, the sample size or the number of columns in the result simulated data.
rand.seed	numeric of length 1, the seed number to start the random sampling process. This argumemnt makes the simulation reproducible as long as its value keeps the same. Default rand.seed=100.

Details

This function is written mainly for simulation or experiment with pathview package. With the simulated molecular data, you may check whether and how pathview works for molecular data of different types, IDs, format or sample sizes etc. You may also generate both gene.data and cpd.data and check data pathway based integration with pathview.

Value

either vector (single sample) or a matrix-like data (multiple sample), depends on the value of nexp. Vector should be numeric with molecular IDs as names or it may also be character of molecular IDs depending on the value of discrete. Matrix-like data structure has molecules as rows and samples as columns. Row names should be molecular IDs.

This returned data can be used directly as gene.data or cpd.data input of pathview main function.

Author(s)

Weijun Luo <luo_weijun@yahoo.com>

References

Luo, W. and Brouwer, C., Pathview: an R/Bioconductor package for pathway based data integration and visualization. Bioinformatics, 2013, 29(14): 1830-1831, doi: 10.1093/bioinformatics/btt285

See Also

node.map the node data mapper function. mol.sum the auxillary molecular data mapper, id2eg, cpd2kegg etc the auxillary molecular ID mappers, pathview the main function,

Examples

```
#continuous compound data
cpd.data.c=sim.mol.data(mol.type="cpd", nmol=3000)
#discrete compound data
cpd.data.d=sim.mol.data(mol.type="cpd", nmol=3000, discrete=TRUE)
head(cpd.data.c)
head(cpd.data.d)
#continuous compound data named with "CAS Registry Number"
cpd.cas <- sim.mol.data(mol.type = "cpd", id.type = "CAS Registry Number", nmol = 10000)

#gene data with two samples
gene.data.2=sim.mol.data(mol.type="gene", nmol=1000, nexp=2)
head(gene.data.2)

#KEGG ortholog gene data
ko.data=sim.mol.data(mol.type="gene.ko", nmol=5000)
```

---

wordwrap *Wrap or break strings into lines of specified width*

Description

strfit does hard wrapping, i.e. break within long words, wordwrap is a wrapper of strfit but also provides soft wrapping option, i.e. break only between words, and keep long words intact.

Usage

```
wordwrap(s, width = 20, break.word = FALSE)
strfit(s, width = 20)
```

Arguments

s	characcter, strings to be wrapped or broken down.
width	integer, target line width in terms of number of characters. d width=20.
break.word	logical, whether to break within words or only between words as to fit the line width. Default break.word=FALSE, i.e. keep words intact and only break between words. Therefore, some line may exceed the width limit.

Details

These functions are called as to wrap long node labels into shorter lines on pathway graphs in keggview.graph function (when keggview.native=FALSE). They are equally useful for wrapping long labels in other types of graphs or output formats.

Value

character of the same length of s except that each element has been wrapped softly or hardly.

Author(s)

Weijun Luo <luo_weijun@yahoo.com>

References

Luo, W. and Brouwer, C., Pathview: an R/Bioconductor package for pathway based data integration and visualization. Bioinformatics, 2013, 29(14): 1830-1831, doi: 10.1093/bioinformatics/btt285

See Also

strwrap in R base.

Examples

```
long.str="(S)-Methylmalonate semialdehyde"
wr1=wordwrap(long.str, width=15)
#long word intact
cat(wr1, sep="\n")
wr2=strfit(long.str, width=15)
#long word split
cat(wr2, sep="\n")
```

Index

∗ **datasets**
  cpd.accs, 4
  demo.data, 6
  korg, 11
∗ **package**
  pathview-package, 2

bods *(*korg*)*, 11

col.key *(*node.color*)*, 14
combineKEGGnodes, 3, *17*
cpd.accs, 4
cpd.names *(*cpd.accs*)*, 4
cpd.simtypes *(*cpd.accs*)*, 4
cpd2kegg, *10, 11*, *13*, *19*, *28*
cpd2kegg *(*cpdidmap*)*, 5
cpdidmap, 5
cpdkegg2name *(*cpdidmap*)*, 5
cpdname2kegg *(*cpdidmap*)*, 5

demo.data, 6
demo.paths *(*demo.data*)*, 6
download.kegg, 7, *11*, *25*

eg2id, *6*, 8

gene.idtype.bods *(*cpd.accs*)*, 4
gene.idtype.list *(*cpd.accs*)*, 4
geneannot.map *(*eg2id*)*, 8
gse16873.d *(*demo.data*)*, 6

id2eg, *6*, *13*, *19*, *28*
id2eg *(*eg2id*)*, 8

kegg.met *(*cpd.accs*)*, 4
kegg.species.code, 10
keggview.graph, *16*
keggview.graph *(*pathview*)*, 20
keggview.native, *16*
keggview.native *(*pathview*)*, 20
ko.ids *(*cpd.accs*)*, 4

korg, 11, *11*

mol.sum, *6*, *10*, 12, *19*, *28*

node.color, 14, *19*, *25*
node.info, *4*, *8*, 16, *19*, *25*
node.map, *6*, *10*, *13*, *16*, 18, *25*, *28*

paths.hsa *(*demo.data*)*, 6
pathview, *8*, *13*, *17*, *19*, 20, *28*
pathview-package, 2

reaction2edge, *17*
reaction2edge *(*combineKEGGnodes*)*, 3
rn.list *(*cpd.accs*)*, 4

sim.mol.data, 26
strfit *(*wordwrap*)*, 28

wordwrap, 28