Package 'KEGGlincs'

October 14, 2021

Type Package

Title Visualize all edges within a KEGG pathway and overlay LINCS data

Version 1.18.0

Date 2016-06-02

Author Shana White

Maintainer
Shana White <vandersm@mail.uc.edu>, Mario Medvedovic <medvedm@ucmail.uc.edu>

Description See what is going on 'under the hood' of KEGG pathways by explicitly re-creating the pathway maps from information obtained from KGML files.

License GPL-3

LazyData true

RoxygenNote 6.1.1

Depends R (>= 3.3), KOdata, hgu133a.db, org.Hs.eg.db (>= 3.3.0)

SystemRequirements Cytoscape (>= 3.3.0), Java (>= 8)

Suggests BiocManager (>= 1.20.3), knitr, graph

biocViews NetworkInference, GeneExpression, DataRepresentation, ThirdPartyClient,CellBiology,GraphAndNetwork,Pathways,KEGG,Network

Imports AnnotationDbi,KEGGgraph,igraph,plyr,gtools,httr,RJSONIO,KEGGREST, methods,graphics,stats,utils, XML, grDevices

VignetteBuilder knitr

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

git_branch RELEASE_3_13

git_last_commit 899d372

git_last_commit_date 2021-05-19

Date/Publication 2021-10-14

R topics documented:

`add_edge_data` *Annotate KEGG edge mappings with user data*

Description

Add data column[s] to object created from function expand_KEGG_edges

Usage

```
add_edge_data(expanded_edges, KEGG_mappings, user_data,
  data_column_no = 3, map_type = "SYMBOL", only_mapped = TRUE)
```

Arguments

`expanded_edges`	The data frame object generated via the function expand_KEGG_edges
`KEGG_mappings`	KEGG_mappings The data.frame object generated by the function expand_KEGG_mappings
`user_data`	A data frame where in which the first two columns contain gene symbols representing an edge and any/all other column[s] contain corresponding edge data.
`data_column_no`	The column index for desired user data to be added
`map_type`	If the genes in your data set are left untranslated set to "NUMBER" (assuming numbers are gene accession numbers)
`only_mapped`	A logical indicator; if set to FALSE will return 'de-novo' edges that 'exist' in data but are not documented in KEGG

Value

A data frame object with detailed KEGG edge mappings annotated with user data

Examples

```
p53_KGML <- get_KGML('hsa04115')
p53_KEGG_mappings <- expand_KEGG_mappings(p53_KGML)
p53_edges <- expand_KEGG_edges(p53_KGML, p53_KEGG_mappings)
p53_HA1E_data <- overlap_info(p53_KGML, p53_KEGG_mappings, 'HA1E',
                              data_type = '100_bing', only_mapped = FALSE)

p53_edges_HA1E <- add_edge_data(p53_edges, p53_KEGG_mappings,
                                      p53_HA1E_data, c(3, 10,12))
```

---

cyto_vis *Send graph to Cytoscape via CyREST*

Description

View the KEGG pathway in Cytoscape. With either the 'expanded edges' or 'stacked nodes' layout, users can visualize and interact with the graphs [strictly] as they are documented in the most recent KGML available from KEGG. This function is a modified version of the function send2cy(), which is part of the cyREST utility functions.

Usage

```
cyto_vis(graph_object, title = "Cytoscape Graph Window",
  edge_width_attribute = "summary_score", port.number = 1234)
```

Arguments

graph_object	An igraph object such as the one generated by the function get_graph_object
title	An optional title for the graph when it is in Cytoscape
edge_width_attribute	The attribute that will be used for edge width; if data is not added or the attribute is not part of the graphing information, the edge width will default to 1.
port.number	The port address for Cytoscape

Value

A dynamic map in Cytoscape automatically formatted for convenient viewing.

Examples

```
p53_KGML <- get_KGML("hsa04115")
p53_KEGG_mappings <- expand_KEGG_mappings(p53_KGML, FALSE)
nodes <- node_mapping_info(p53_KEGG_mappings)

p53_edges <- expand_KEGG_edges(p53_KGML, p53_KEGG_mappings)
edges <- edge_mapping_info(p53_edges)

p53_graph_object <- get_graph_object(nodes, edges)

## Not run:
cyto_vis(p53_graph_object, "Default p53 Graph [no data added]")

#Workflow to visualize graph with data-dependent attributes:

p53_KGML <- get_KGML("hsa04115")
p53_KEGG_mappings <- expand_KEGG_mappings(p53_KGML)
nodes <- node_mapping_info(p53_KEGG_mappings)

p53_edges <- expand_KEGG_edges(p53_KGML, p53_KEGG_mappings)

p53_HA1E_data <- overlap_info(p53_KGML, p53_KEGG_mappings, "HA1E",
                              data_type = "100_bing")
p53_edges_plus_data <- add_edge_data(p53_edges, p53_KEGG_mappings,
                                     p53_HA1E_data, c(3, 10,12),
                                     only_mapped = TRUE)

edges <- edge_mapping_info(p53_edges_plus_data, data_added = TRUE)

p53_plus_data_graph_object <- get_graph_object(nodes, edges)

cyto_vis(p53_plus_data_graph_object, "p53 Graph: Mapped Edges + HA1E Data",
         edge_width_attribute = "UP")

## End(Not run)
```

---

edge_mapping_info *Prepare edges for mapping*

Description

Modify the mapping information for desired look when graphed in Cytoscape

Usage

```
edge_mapping_info(expanded_edges, data_added = FALSE,
  significance_markup = FALSE, tidy_edge = TRUE)
```

Arguments

expanded_edges	The data frame object generated via the function expand_KEGG_edges() OR has been modified by the function add_edge_data()
data_added	A logical indicator; must be set to TRUE if user data has been added (i.e. edges modified by function add_edge_data())
significance_markup	A logical indicator; if set to TRUE will color edges based on direction and significance of correlation (as determined by user-data-analysis)
tidy_edge	A logical indicator; must be set to FALSE for expanded edges

Value

A data.frame object for edges that will be passed on to the function get_graph_object

Examples

```
p53_KGML <- get_KGML("hsa04115")
p53_KEGG_mappings <- expand_KEGG_mappings(p53_KGML)

#Default; no data added to edges:

p53_edges <- expand_KEGG_edges(p53_KGML, p53_KEGG_mappings)
p53_edge_mapping_info <- edge_mapping_info(p53_edges)

#If data is added to edges as additional attribute[s]:

p53_HA1E_data <- overlap_info(p53_KGML, p53_KEGG_mappings,
                              "HA1E", data_type = "100_bing")

p53_edges_HA1E_data_MAPPED <- add_edge_data(p53_edges, p53_KEGG_mappings,
                                            p53_HA1E_data,
                                            data_column_no = c(3, 10,12),
                                            only_mapped = TRUE)

p53_edge_mapping_HA1E <- edge_mapping_info(p53_edges_HA1E_data_MAPPED,
                                                      data_added = TRUE)
```

---

expand_KEGG_edges *Get detailed KEGG mapping information for each relation [edge] documented in KEGG*

Description

Extract relationship information from KGML object and re-map based on normalized node information

Usage

```
expand_KEGG_edges(KGML_file, KEGG_mappings)
```

Arguments

KGML_file	An object of formal class KEGGPathway
KEGG_mappings	The data.frame object generated by the function expand_KEGG_mappings

Value

A dataframe object with unique entry information for all edges documented in the KEGG pathway. Note that each row has a unique combination of values for (entry1, entry2, entry1symbol, entry2symbol).

Examples

```
p53_KGML <- get_KGML("hsa04115")
p53_KEGG_mappings <- expand_KEGG_mappings(p53_KGML, FALSE)
p53_edges <- expand_KEGG_edges(p53_KGML, p53_KEGG_mappings)
```

---

expand_KEGG_mappings *Get detailed KEGG mapping information for each map entity*

Description

Extract mapping information from KGML object and normalize mappings based on multi-valued name attribute

Usage

```
expand_KEGG_mappings(KGML_file, convert_KEGG_IDs = TRUE)
```

Arguments

KGML_file	An object of formal class KEGGPathway
convert_KEGG_IDs	A logical indicator; if set to FALSE will run faster however genes and compounds will remain labeled via KEGG codes (compounds) or accession numbers (genes). This option must be taken into account if data is being added. For example, the genes in 'KO_data' are identified by symbols, thus it is neccessary to retain the default option to convert IDs to symbols when planning to add edge data of this type.

Value

A dataframe object with unique entry information for all [node] objects documented in the KEGG pathway. Note that if mutiple objects (i.e. genes or compounds) have the same entryID, this indicates that they share the same node [location] in the pathway.

Examples

```
p53_KGML <- get_KGML("hsa04115")
p53_KEGG_mappings <-  expand_KEGG_mappings(p53_KGML, FALSE)
```

---

generate_mappings *The 'boilerplate' for this package's desired graph style*

Description

Generates an object that can be converted to a JSON file and subsequently applied to the graph for the markup specified by this package and the layout mirroring KEGG. Intended for use within cyto_vis

Usage

```
generate_mappings(style_name, map_edge_width, edge_width_attribute,
  min_score, max_score)
```

Arguments

style_name	An argument to name style; when used inside of cyto_vis no name is needed
map_edge_width	A logical indicator; if FALSE no continuous mapping of edge width will be applied
edge_width_attribute	The attribute that will be used for edge width; if data is not added or the attribute is not part of the graphing information, the edge width will default to 1.
min_score	The minimum attribute value for the column used to map edge width
max_score	The maximum attribute value for the column used to map edge width

Value

A list that can be converted to a JSON file to apply desired style/layout in Cytoscape

Examples

```
style.name = "myKEGGstyle"
mappings <- generate_mappings(style.name, FALSE)
```

get_fisher_info *Perform Fisher's Exact test for edges in pathway*

Description

Obtain a measure for strength and significance for the relationship (i.e. an edge) based on the concordance/discordance of UP-and-DOWN regulated genes shared by two different experimental gene-knockouts Intended for use within overlap_info

Usage

```
get_fisher_info(edges, method)
```

Arguments

edges	The set of eges to be analyzed; Although the intended use is for LINCS data overlaps, the function should work with any typical data object as long as it has columns labeled ("UP", "DOWN", "UK1_DK2", "DK1_UK2") that contain integer values.
method	The method to correct/adjust p-values for multiple testing. For available methods, type 'p.adjust.methods' into command promt and press enter.

Value

The input edge data.frame object with additional columns containing the results of the applied statistical test

Examples

```
ex.data <- data.frame("UP" = c(70,6), "DOWN" = c(8,20),
                      "UK1_DK2" = c(4,47), "DK1_UK2" = c(3, 28))

overlaps <- get_fisher_info(ex.data, method = "BH")
```

get_graph_object *Generate graph object from nodes and edges*

Description

Obtain a graph object in the form of an igraph with KEGG-specific graphical information

Usage

```
get_graph_object(node_mapping_info, expanded_edges,
  layered_nodes = FALSE)
```

Arguments

node_mapping_info	The data.frame object generated by the function node_mapping_info()
expanded_edges	The data.frame object generated by the function edge_mapping_info()
layered_nodes	A logical indicator; if set to TRUE will create a graph with 'stacked' nodes that the user can manipulate when multiple nodes are mapped to one location

Value

A list object with the node and edge information from the graph required for mapping.

Examples

```
p53_KGML <- get_KGML("hsa04115")
p53_KEGG_mappings <-  expand_KEGG_mappings(p53_KGML)

p53_node_mapping_info <- node_mapping_info(p53_KEGG_mappings)
p53_edge_mapping_info <- expand_KEGG_edges(p53_KGML, p53_KEGG_mappings)

#Default graph object will have 'expanded edges':
expanded_edges_graph_object <- get_graph_object(p53_node_mapping_info,
                                                p53_edge_mapping_info)

#Graph with layered nodes:
layered_nodes_graph_object <- get_graph_object(p53_node_mapping_info,
                                               p53_edge_mapping_info,
                                               layered_nodes = TRUE)
```

---

get_KGML *Download and parse KGML file*

Description

Download and parse KGML file

Usage

```
get_KGML(pathwayid, get_if_no_edges = FALSE)
```

Arguments

pathwayid	A KEGG pathway ID of the form "hsa12345" (only human pathways currently)
get_if_no_edges	A logical indicator; if pathway has no edges returns null value if set to TRUE

Value

an object of Formal class KEGGPathway

Examples

```
mtor_KGML <- get_KGML("hsa04150")

# Some pathways contain only node information; since the purpose of this
# package is to explore pathways in an edge-focused manner, the default
# options return a warning message instead of a parsed KGML file if the
# input pathway has no edges.
ribosome_KGML <- get_KGML("hsa03020")
ribosome_KGML <- get_KGML("hsa03020", get_if_no_edges = TRUE)
```

---

keggerize_edges *Add in edges to map documented in other pathways*

Description

For a specific pathway entity(gene), search KEGG databases to see if it has any other documented relationships in KEGG. expand_KEGG_edges

Usage

```
keggerize_edges(entry_accession, KGML, KEGG_mappings, edges)
```

Arguments

entry_accession	The Accession # of the pathway entity to 'keggerize'
KGML	The KGML file of the current pathway
KEGG_mappings	KEGG mappings for the current pathway
edges	The expanded edges for the current pathway

Value

A modified expanded edges data frame with additional rows for new entries

Examples

```
## Not run:
KGML <- get_KGML("hsa04150")
KEGG_mappings <- expand_KEGG_mappings(KGML)
edges <- expand_KEGG_edges(KGML, KEGG_mappings)
entry_accession <- "2475"
mtor_plus_mtor <- keggerize_edges(entry_accession = entry_accession,
                                  KGML = KGML,KEGG_mappings = KEGG_mappings,
                                  edges = edges)
## End(Not run)
```

KEGGlincs *KEGGlincs: an R package designed to explore the edges in KEGG pathways*

Description

KEGGlincs: an R package designed to explore the edges in KEGG pathways

KEGG_lincs *Combines all other package functions for one-step pathway visualization*

Description

Combines all other package functions for one-step pathway visualization

Usage

```
KEGG_lincs(pathwayid, cell_line = NA, refine_by_cell_line = NA,
  add_L1000_edge_data = TRUE, significance_markup = TRUE,
  data_type = "100_full", pert_time = 96, only_mapped = TRUE,
  layered_nodes = FALSE, graph_title = "default", get_data = FALSE,
  convert_KEGG_IDs = TRUE, tidy_edge = FALSE)
```

Arguments

pathwayid	A KEGG pathway ID of the form "hsa12345" (only human pathways currently)
cell_line	If left as NA will generate a pathway map without data-dependent attributes (such as edge width). To use in combination with LINCS data, choose from the set of cell lines: (A375,A549,ASC,HA1E,HCC515,HEK293T,HEKTE,HEPG2,HT29,MCF7,NCIH716,N SHSY5Y,SKL,SW480,VCAP)
refine_by_cell_line	A logical indicator
add_L1000_edge_data	A logical indicator
significance_markup	A logical indicator; if set to TRUE will color edges based on direction and significance of correlation (as determined by user-data-analysis)
data_type	Choose from data types: (100_full, 100_bing, 50_lm)
pert_time	Choose from (6,24,48,96,120,144,168)
only_mapped	A logical indicator; if set to FALSE will return 'de-novo' edges that 'exist' in data but are not documented in KEGG

layered_nodes	A logical indicator; if set to TRUE will create a graph with 'stacked' nodes that the user can manipulate when multiple nodes are mapped to one location
graph_title	An optional user-specified graph title
get_data	A logical indicator; if set to true, will return the 'expanded' edge information for the specified pathway
convert_KEGG_IDs	A logical indicator; if set to TRUE KEGG compounds will remain labeled via KEGG codes (do not need KEGGREST)
tidy_edge	A logical indicator; must be set to FALSE for expanded edges

Value

A dynamic map in Cytoscape automatically formatted for convenient viewing and, if idicated by user, a data.frame object with detailed information for 'expanded' KEGG edges

Examples

```
## Not run:

#Default KEGG pathway with colored edges representing type of relationship:
KEGG_lincs("hsa04115", convert_KEGG_IDs = FALSE)

#KEGG pathway with edge width and color based on observed experimental data:
KEGG_lincs("hsa04115", "HA1E")

#Have edge information data.frame to be output to the global environment:
p53_edge_info <- KEGG_lincs("hsa04115", graph_title = "p53"
                            convert_KEGG_IDs = FALSE, get_data = TRUE)

## End(Not run)
```

KL_compare *Combines all other package functions for one-step cell line comparison*

Description

Combines all other package functions for one-step cell line comparison

Usage

```
KL_compare(pathwayid, cell_line1 = NA, cell_line2 = NA,
  refine_by_cell_line = TRUE, data_type = "100_full", pert_time = 96,
  only_mapped = TRUE, get_data = FALSE, convert_KEGG_IDs = TRUE,
  graph_title = "default", tidy_edge = TRUE, layered_nodes = FALSE)
```

Arguments

pathwayid	A KEGG pathway ID of the form "hsa12345" (only human pathways currently)
cell_line1	Choose from the set of cell lines: (A375,A549,ASC,HA1E,HCC515,HEK293T,HEKTE,HEPG2,HT29,M SHSY5Y,SKL,SW480,VCAP)
cell_line2	A cell line such that cell_line1 != cell_line2
refine_by_cell_line	A logical indicator
data_type	Choose from data types: (100_full, 100_bing, 50_lm)
pert_time	Choose from (6,24,48,96,120,144,168)
only_mapped	A logical indicator; if set to FALSE will return 'de-novo' edges that 'exist' in data but are not documented in KEGG
get_data	A logical indicator; if set to true, will return the 'expanded' edge information for the specified pathway
convert_KEGG_IDs	A logical indicator; if set to TRUE KEGG compounds will remain labeled via KEGG codes (do not need KEGGREST)
graph_title	An optional user-specified graph title
tidy_edge	A logical indicator; must be set to FALSE for expanded edges
layered_nodes	A logical indicator; if set to TRUE will create a graph with 'stacked' nodes that the user can manipulate when multiple nodes are mapped to one location

Value

A dynamic map in Cytoscape automatically formatted for convenient viewing and, if idicated by user, a data.frame object with detailed information for 'expanded' KEGG edges

Examples

```
## Not run:

# Compare p53 pathway between cell lines A375 and A549:
KL_compare("hsa04115", "A375", "A549")

## End(Not run)
```

---

node_mapping_info *Prepare nodes for mapping*

Description

Modify the mapping information for desired look when graphed in Cytoscape

Usage

```
node_mapping_info(KEGG_mappings)
```

Arguments

KEGG_mappings	The data.frame object generated by the function expand_KEGG_mappings()

Value

A data.frame object for nodes that will be passed on to the function get_graph_object

Examples

```
p53_KGML <- get_KGML("hsa04115")
p53_KEGG_mappings <- expand_KEGG_mappings(p53_KGML, FALSE)

p53_node_mapping_info <- node_mapping_info(p53_KEGG_mappings)
```

---

overlap_info *Get overlap information for pairs of gene knock-outs from LINCS data*

Description

Get overlap information for pairs of gene knock-outs from LINCS data

Usage

```
overlap_info(KGML_file, KEGG_mappings, cell_type, data_type = "100_full",
  pert_time = 96, only_mapped = TRUE, affy_based = FALSE,
  keep_counts_only = TRUE, add_fisher_information = TRUE,
  p.adjust.method = "BH")
```

Arguments

KGML_file	An object of formal class KEGGPathway
KEGG_mappings	The data.frame object generated by the function expand_KEGG_mappings
cell_type	Choose from the set of cell lines: (A375,A549,ASC,HA1E,HCC515,HEK293T,HEKTE,HEPG2,HT29,M SHSY5Y,SKL,SW480,VCAP)
data_type	Choose from data types: (100_full, 100_bing, 50_lm)
pert_time	Choose from (6,24,48,96,120,144,168)
only_mapped	A logical indicator; if set to FALSE will return 'de-novo' edges that 'exist' in data but are not documented in KEGG
affy_based	A logical indicator; if set to TRUE will return lists/counts based on probeID instead of gene symbol.
keep_counts_only	A logical indicator; if set to FALSE will return data frame with lists [of gene symbols or probe ids] as well as counts
add_fisher_information	A logical indicator; by default the relationships are analyzed for strength of correlation via Fisher's Exact Test

`p.adjust.method`
: For available methods, type 'p.adjust.methods' into command promt and press enter.

Value

A data frame where each row corresponds to information for pairs of experimental gene knock-outs from LINCS data (found in selected pathway).

Examples

```
p53_KGML <- get_KGML("hsa04115")
p53_KEGG_mappings <-  expand_KEGG_mappings(p53_KGML)
p53_edges <- expand_KEGG_edges(p53_KGML, p53_KEGG_mappings)

summary <- path_genes_by_cell_type(p53_KEGG_mappings)
p53_HA1E_data <- overlap_info(p53_KGML, p53_KEGG_mappings,
                              "HA1E", data_type = "100_bing",
                              only_mapped = FALSE)
```

---

path_genes_by_cell_type

See how many pathway gene knock-outs are available from data

Description

Check quantity of data across cell lines available from LINCS corresponding to the pathway of interest

Usage

```
path_genes_by_cell_type(KEGG_mappings, pert_time = 96, get_KOs = FALSE,
  generate_plot = TRUE)
```

Arguments

`KEGG_mappings`	KEGG_mappings The data.frame object generated by the function expand_KEGG_mappings
`pert_time`	Choose from (6,24,48,96,120,144,168)
`get_KOs`	Logical indicator to have data frame returned
`generate_plot`	Logical indicator to generate histogram

Value

A plot depicting percentage of pathway genes knocked-out by cell line and a data frame object listing the genes [by cell line]

Examples

```
p53_KGML <- get_KGML("hsa04115")
p53_KEGG_mappings <- expand_KEGG_mappings(p53_KGML)

path_genes_by_cell_type(p53_KEGG_mappings)
```

`refine_mappings` *Refine pathway by cell type*

Description

Reduce the KEGG pathway by only including genes that are expressed within a given cell type

Usage

```
refine_mappings(KEGG_mappings, cell_line)
```

Arguments

`KEGG_mappings`	The data.frame object generated by the function expand_KEGG_mappings
`cell_line`	Choose from the set of cell lines with baseline data; cell-lines may or may not have corresponding KO data

Value

A dataframe object with reduced set of pathway mappings to be passed on to other functions

Examples

```
p53_KGML <- get_KGML("hsa04115")
p53_KEGG_mappings <-  expand_KEGG_mappings(p53_KGML)
MCF7_p53_mappings <- refine_mappings(p53_KEGG_mappings, "MCF7")
```

`tidy_edge` *Tidy up pathway by combining edges inside of edge_mapping_info*

Description

Combine edges that share nodes and have other commonalities

Usage

```
tidy_edge(edges, edge_id, data_added = FALSE, by_significance = FALSE,
  by_number = TRUE)
```

Arguments

edges	The edge dataframe
edge_id	The numeric value for the edge_id
data_added	A logical indicator; set to TRUE if data is added
by_significance	A logical indicator; option if data is added
by_number	A logical indicator; gives rough estimate of edge amount

Value

A data frame that has had the given edge condensed for viewing

Examples

```
## Not run:
if (tidy_edge == TRUE) {
   edge_IDs <- seq(min(expanded_edges$edgeID), max(expanded_edges$edgeID))
   for (i in edge_IDs){
     if(data_added == TRUE){
       expanded_edges <- tidy_edge(edges = expanded_edges,
                                   edge_id = edge_IDs[i],
                                   data_added = TRUE,
                                   by_significance = TRUE)
     }
     if(data_added == FALSE){
       expanded_edges <- tidy_edge(edges = expanded_edges,
                                   edge_id = edge_IDs[i],
                                   data_added = FALSE)
     }
   }

}

## End(Not run)
```

---

toCytoscape *cyREST utility functions*

Description

A subset of the R utility functions available from/defined by cyREST. The function mapAttributes is called from within toCytoscape which, in turn, is called from within cyto_vis.

Usage

```
toCytoscape(igraphobj)

mapAttributes(attr.names, all.attr, i)
```

Arguments

`igraphobj`	A graph object compatible for use with the package igraph
`attr.names`	Attribute names of an igraph object
`all.attr`	The attribute value if an igraph object
`i`	The index for a given igraph object

Value

A JSON object to be sent to Cytoscape

Index