Package 'MetamapsDB'

December 6, 2017

Title Network Analysis of Metabolic Pathways and Gene Based Assemblies in Metagenomics

Maintainer Wesley GOI <wesley@bic.nus.edu.sg>

Author Wesley GOI

Version 0.0.2

License GPL (>= 2)

Depends R (>= 3.1.1)

Description Functions for path based queries against 'Omics' <https://github.com/etheleon/omics-neo4j-container>, a 'NEO4J' <https://neo4j.com/> database populated with data from metagenomic surveys and their accompanying metabolic and taxonomic annotations.

Imports IRanges, ShortRead, future, GenomicRanges, forcats, ggplot2, purrr, gridExtra, lubridate, zoo, magrittr, stats, utils, grDevices, parallel, RJSONIO, RCurl, igraph, rgexf, dplyr (>= 0.3.0.1), Matrix (>= 1.2-12), stringr, tidyr, shiny, base64enc, cluster, DBI, RSQLite, data.table, httr,

LazyData TRUE

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-12-06 09:51:00 UTC

R topics documented:

`addContigProperty` *addContigProperty*

Description

Adds property to contig node

Usage

```
addContigProperty(contig, ko, property, all = FALSE, batchSize = 10000,
  add = FALSE)
```

Arguments

contig	the contigID
ko	the KO the contig belongs to
property	list containing the property which you would want to insert, name of property in list is the name of the property in the
all	add all
batchSize	the size of number of entries
add	if to add property

---

addKOProperty *addKOProperty*

Description

Adds property to ko node

Usage

```
addKOProperty(ko, property)
```

Arguments

ko	the koID
property	the property which you want to add

---

adjacentPairs *Given a igraph object, adjacentPairs finds adjacent pairs of KOs*

Description

Given a igraph object, adjacentPairs finds adjacent pairs of KOs

Usage

```
adjacentPairs(g, ko2ko = FALSE, ...)
```

Arguments

g	graph object
ko2ko	if true returns data.frame of ko pairs, if false returns the compound which its connected with
...	ellipise pass in other arguments which are required

Examples

```
# gives ko 2 ko pairs
adjacentPairs(nitrogenMetab, ko2ko=TRUE)
#        ko1       ko2 direction
#1 ko:K18246 ko:K01725       out
#2 ko:K18245 ko:K01725       out
#3 ko:K17877 ko:K17877       out
#4 ko:K17877 ko:K15876       out
#5 ko:K17877 ko:K15371       out
#6 ko:K17877 ko:K10946       out
```

`allTrios` *Find all trios surrounding the KO of interest*

Description

Searches graphDB for all cases of trios surrounding the KO of interest

Usage

```
allTrios(KOI = "ko:K00001", koi = "ko:K00001", toUnique = TRUE,
  withDetails = FALSE, local = TRUE, contracted = TRUE, ...)
```

Arguments

`KOI`	the ko id (for remote)
`koi`	the ko id (for local)
`toUnique`	if to return only unique (only applicable for remote)
`withDetails`	with details (only applicable for remote)
`local`	for doing the search locally, faster
`contracted`	to contract anot (only applicable for local)
`...`	the other args for dbquery

`annotateContigs.taxonomy`
Adds taxonomic annotations of the contigs

Description

After loading into the graphDB contigs information we annotate contigs with their taxonomy sequences

Usage

```
annotateContigs.taxonomy(csv)
```

Arguments

csv	FULL path pointing to outputfile from mapBlat package's blast2lcaIn fxn

---

blatting *blatting*

Description

performs blat

Usage

```
blatting(ko, type, newblerInput)
```

Arguments

ko	the ko of interest
type	the
newblerInput	path to newbler dir eg. /root in /root/K0000X/input/K0000X.1.fq

---

buildTree *Taxonomic tree of the taxons*

Description

'buildTree' Returns taxonomic tree as a igraphObject of the taxids up to phylum

Usage

```
buildTree(taxids = c(287, 280), standardisedRanks = c("superkingdom",
  "phylum", "class", "order", "family", "genus", "species"))
```

Arguments

taxids	vector of leave nodes
standardisedRanks	the ranks to report. superkingdom phylum class order family genus species

Examples

```
## Not run:
standardisedRanks  = c("superkingdom", "phylum", "class", "order","family", "genus", "species")
buildTree(taxids = c(287, 280), standardisedRanks )

## End(Not run)
```

---

connect *Connecting to NEO4J Graph Database.*

Description

'connect' sets params such as the REST API URL, port, db username and password, In the newer versions of neo4j default requires username and password. If this is the first time connecting to a new neo4jDB, please index the nodes else the queries will be very slow.

Usage

```
connect(url = "127.0.0.1", port = 7474, username = "neo4j",
  password = "neo4j", test = TRUE)
```

Arguments

url	url to running instance of neo4j
port	port
username	username
password	password
test	run test to show connection success or failure

Examples

```
## Not run:
connect(url="127.0.0.1", port=7474, username="neo4j", password="neo4j")

If this is your first time connecting to omics DB please index the nodes, using `index`
else query speed will be very slow.

Sending test query: Searching for K00001
#A successful connection will give the following
Connection Successful

Found these indices:
  property_keys   label
1        contig contigs
2           bin contigs
3           cpd     cpd
4            ko      ko
```

```
5         taxid   Taxon

## End(Not run)
```

---

contigInfo *contigInfo*

Description

contigInfo function greps the contig related info such as whether its in the MDR

Usage

```
contigInfo(ko, contig, withAnnotation = FALSE)
```

Arguments

ko	the ko bin which the contig was assembled from
contig	contigID (optional)
withAnnotation	0j/

Examples

```
## Not run:
To query a specific contig
contigInfo(ko = "K00001", contig="contig00001")
# Sending query
# Query Completed

#        bin             contig MDR spanning readnum length
#1 ko:K00001 K00001:contig00001   1        1    1497   1236
With annotation
   assignment taxid       bin             contig readnum length MDR spanning
1 Bacteroides   816 ko:K00001 K00001:contig00001    1497   1236   1         1
2 Bacteroides   816 ko:K00001 K00001:contig00001    1497   1236   1         1
     Rank
1   genus
2 simTaxa

To query all contigs associated with a KO
contigInfo(ko = "K00001") %>% head
#Sending query
#
#Query Completed
#
#   MDR spanning       bin             contig length readnum
#1 <NA>     <NA> ko:K00001 K00001:contig00168    297      31
#2 <NA>     <NA> ko:K00001 K00001:contig00167    298     118
#3    1        0 ko:K00001 K00001:contig00166    298      45
```

```
#4    1        0 ko:K00001 K00001:contig00164    299     236
#5    1        1 ko:K00001 K00001:contig00163    300     106
#6 <NA>     <NA> ko:K00001 K00001:contig00162    302     310

## End(Not run)
```

`contractMetab` *contractMetab shrinks a metabolic network's KOs into non-redundant units*

Description

Constracts KOs which share the same cpd product and substrates.

Usage

```
contractMetab(g)
```

Arguments

g	metabolic graph, igraph object. vertices must have the Symbol property

Value

new non-redundant (gene) metabolic graph

Examples

```
contactedMetab <- contractMetab(nitrogenMetab)
```

`cpdname` *Finds the details of the CPD when given its ID*

Description

Finds the details of the CPD when given its ID

Usage

```
cpdname(cpd = "C00022", ...)
```

Arguments

cpd	the compound ID with or without the prefix is accepted either as a single or vector accepts with or without prefix
...	allows for additional arguments use for dbquery

dbquery *Function for querying metamaps DB*

Description

Takes variable queries with table output ie. no nodes no rels

Usage

```
dbquery(query, params = FALSE, cypherurl = NULL, user = NULL,
  password = NULL, justPost = FALSE, verbose = FALSE, test = FALSE,
  jsonresponse = "", ...)
```

Arguments

query	char string with cypher Query, rmbr to escape double quotes if any; read LUCENE
params	a list object. ie params required by a cypher query, false if query requires no parameters; default
cypherurl	address of database, inclusive of ports
user	database username
password	database password
justPost	its just a POST request
verbose	to print details out
test	for testing, function takes a given json response in the form of a character string provided in argument jsonresponse
jsonresponse	string equivalent to the json response from querying the database
...	allows for additional arguments to be passed into dbquery

findHomology *searches for KO's homology assignments*

Description

'findHomology' seeks contigs which have been assigned via to the given KO of interest used after homology search data has been uploaded into the db

Usage

```
findHomology(koi)
```

Arguments

koi	the ko of interest

`findNextKO` *FindNextKO*

Description

Given a specied KO, one is suppose to find the next adjacent KO.

Usage

```
findNextKO(cpd, graph, originalKO, direction)
```

Arguments

`cpd`	the intermediary cpd
`graph`	metabolic graph
`originalKO`	KO of interest
`direction`	the direction in which to find

`findSeeds` *To find all seed compounds in the metabolic graph*

Description

Function finds strongly connected components within a bipartite metabolic network. And selects for components with at least 1 outgoing edge and no inbound edges. This is an implementation of Large-scale reconstruction and phylogenetic analysis of metabolic environments by Borenstein et al

Usage

```
findSeeds(mbgraph)
```

Arguments

`mbgraph`	igraph object representing the metabolic graph of interest

findTrios — *findTrios searches valid three-KO reactions resticting by the center KO and reports reaction clusters*

Description

findTrios uses K-means clustering to identify reaction groups/clusters and the GAP statistic by Ryan Tibshirani to identity the best k implemented in the library cluster Clustering algorithm uses the input matrix cmoposed of the the KS statistics of the KOs flanking the KO of interest.

Usage

```
findTrios(KOI, ks, toPrint = TRUE, outputFile, plotDir)
```

Arguments

KOI	KOs-of-interest ie KOs above a selected amount of expression ie. highly expressed KOs and with D-statistics above that of the desired threshold
ks	Precalculated data.frame output from ksCal fxn containing all KO's gene distribution KS statistic when compared with whole sample's empirical gene distribution
toPrint	conditional to print the results of the classification plots
outputFile	the txt file to write the results of findTrios to
plotDir	the location to save the diagnostics plots to

Details

The KS statistics is calculated based on the comparison of each KO's gene/contig expression distribution against the 'null' ie. empirical distribution against all genes in all contigs

In addition it also identifies reactions where flanking KOs have high gene diversity, given by low KS (<= 0.5)

Value

data.frame of all reactions, corresponding clusters and the KS statistics for each KO and the selected Cluster

findtype *findType finds KOs/compounds ID in metabolic graph*

Description

findType finds KOs/compounds ID in metabolic graph

Usage

```
findtype(graph, type = c("c", "k"))
```

Arguments

graph	graph object
type	either find Cpds or KOs

Value

vector which nodes are of the certain type

grepgraph *Returns the metabolic graph given vector of KOs*

Description

Takes in a vector list of KO ids and generates a metabolic graph as a igraph object

Usage

```
grepgraph(kos, fullGraph = FALSE, ...)
```

Arguments

kos	vector of kos
fullGraph	to output the full metabolic graph (yet to be implemented)
...	additional dbquery parameters

grepgraph.cpd *Returns the metabolic graph given vector of KOs*

Description

Takes in a vector list of cpd ids and generates a subset of metabolic graph as a igraph object

Usage

```
grepgraph.cpd(cpds, fullGraph = FALSE, ...)
```

Arguments

cpds	vector of cpds
fullGraph	to output the full metabolic graph (yet to be implemented)
...	additional dbquery parameters

ig2ggplot *Convert igraph to ggplot2 object*

Description

Converts a metabolic igraph obj into a ggplot plot

Usage

```
ig2ggplot(g, dfOnly = TRUE, labels = FALSE, metab = TRUE, ...)
```

Arguments

g	igraph object
dfOnly	outputs the data.frame
labels	to show labels or not
metab	it is a metabolic graph
...	additional dbquery parameters

Examples

```
p = nitrogenMetab %>% prettifyGraph %>% ig2ggplot(., dfOnly=FALSE)
p
```

igraph2gexf *Converts igraph obj two gexf Function for converting igraph 2 gexf*

Description

Converts igraph obj two gexf Function for converting igraph 2 gexf

Usage

```
igraph2gexf(mbgraph)
```

Arguments

mbgraph	metabolic graph

index *Indexes the database for faster retrieval*

Description

'index' indexes nodes based on the given property. Lets you do fast search for a node using a property field.

Usage

```
index(label, property = "contig", cacheEnv)
```

Arguments

label	label of the nodes you want to index
property	the property key on which you would like to index on
cacheEnv	stores details

Examples

```
## Not run:
f = future({
    dbquery(query="CREATE INDEX ON :contigs(contig)", justPost = TRUE)
    dbquery(query="CREATE INDEX ON :contigs(bin)", justPost = TRUE)
    dbquery(query="CREATE INDEX ON :Taxon(taxid)", justPost = TRUE)
    dbquery(query="CREATE INDEX ON :cpd(cpd)", justPost = TRUE)
    message
}) %plan% multiprocess

## End(Not run)
```

ko2path *ko2path Finds all pathways related to the KO Finds all associated pathways, and returns a data.frame with ko and pathway details*

Description

ko2path Finds all pathways related to the KO Finds all associated pathways, and returns a data.frame with ko and pathway details

Usage

```
ko2path(ko = "K00001", ...)
```

Arguments

ko	the ko ID int the form eg. K00001
...	the extra params to pass to dbquery

koname *Gives KO details when supplied with KO id*

Description

Gives KO details when supplied with KO id

Usage

```
koname(ko = "K00001", minimal = TRUE, test = FALSE, ...)
```

Arguments

ko	the koid
minimal	return minimal details
test	if it should be run as a example
...	additional dbquery parameters

Examples

```
koname(test=TRUE)
# JSON response from DB
# {
#  "columns" : [ "ko.ko", "ko.name", "ko.definition"  ],
#    "data" : [ [ "ko:K00001", "E1.1.1.1, adh", "alcohol dehydrogenase [EC:1.1.1.1]"  ] ]
#
#}
#
#       ko.ko       ko.name                       ko.definition
# 1 ko:K00001 E1.1.1.1, adh alcohol dehydrogenase [EC:1.1.1.1]
```

lca *Finds the lowest common ancestor*

Description

Finds the lowest common ancestor

Usage

```
lca(taxon1 = "10090", taxon2 = "9096", recurse = TRUE, ...)
```

Arguments

taxon1	First taxon NCBI tax id
taxon2	Second taxon NCBI tax id
recurse	to keep moving up to find the LCA
...	additional dbquery parameters

listquery *Function for querying metamaps DB*

Description

Takes variable queries with table output ie. no nodes no rels (simpler version of DBquery gives a list)

Usage

```
listquery(query, params = FALSE, cypherurl, user, password, ...)
```

Arguments

query	cypher query
params	string containing cypher query
cypherurl	address of database, inclusive of ports
user	database username
password	database password
...	accepts additional parameters from external function calls

lookupTable — *generates 'lookupTable' for filtering raw data post gene centric assembly*

Description

returns three tables, 1. gene centric assembly readOrigin identities, 2. homologySearch assignment, 3 superimposed

Usage

```
lookupTable(genesOfInterest = scg,
  annotations = "~/simulation_fr_the_beginning/out/template.csv", rs)
```

Arguments

genesOfInterest	the kos of interest defaults to scg must be in the ko:KXXXXX format
annotations	table containing the number of KOs for each taxID
rs	processed readStatuses from newbler output

Examples

```
## Not run:
anno = "~/simulation_fr_the_beginning/out/template.csv"
rs = lapply(dynList, function(x) x$rs %>% mutate(ko = x$ko)) %>%
   do.call(rbind,.)
output = lookupTable(genesOfInterest = scg,
   annotations = anno, rs)
e

## End(Not run)
```

map — *map takes cDNA reads (fastQ format) and maps them onto contigs*

Description

uses blat to map the mRNA to the contigs fasta and returns the tabular blast format blat was chosen for its ability to show multiple hits https://www.biostars.org/p/17613/

Usage

```
map(reads, contigs)
```

Arguments

reads	Shortread object or path to short reads file in fastq format
contigs	path to contig file in fa format

Examples

```
## Not run:
# newbler2/K00927
#    454AllContigs.fna
#    input
#        K00927.1.fq
#        K00927.2.fq

lapply(kois, function(ko){
    read1 = sprintf("./newbler2/%s/input/%s.2.fq", ko, ko)
    if file.exists(read1)
        read1.fq = read1 %>% readFastq
    read2 = sprintf("./newbler2/%s/input/%s.2.fq", ko, ko)
    if file.exists(read2)
        read2.fq = read2 %>% readFastq
    combined = c(grep.cDNA(read1.fq),grep.cDNA(read2.fq))
    contigs = sprintf("./newbler2/%s/454AllContigs.fna", ko)
    output = sprintf("./blat/%s.m8", ko)
    blast8 = map(combined, contigs, output) %<>% tbl_df
})

## End(Not run)
```

---

mapContig *finds the location of the MDR on the contig*

Description

finds the location of the MDR on the contig

Usage

```
mapContig(contigid, seq, s, e)
```

Arguments

contigid	the id of the contig
seq	character string of the contig from the MSA
s	starting loc
e	ending loc

mappingInfo *mappingInfo sequence analysis of contigs (for SIMULATION only)*

Description

Used for mapping short reads against contigs using blast

Usage

```
mappingInfo(koi = "K00927", doBlast = FALSE, ...)
```

Arguments

koi	the KO of interest
doBlast	read from file or carry out blast
...	root, outputDir, query, rangesFile, genomesFile, blastFile

Examples

```
## Not run:
blastfile = "~/simulation_fr_the_beginning/reAssemble/everybodyelse/out/simBlast/K00927/blast.tsv"
mappingInfo(blastfile, doBlast=FALSE)

## End(Not run)
```

mapReads2MDR *overlaps*

Description

calculates overlaps

Usage

```
mapReads2MDR(ko, passDir, newblerInput)
```

Arguments

ko	the ko of interest
passDir	the directory of the pass project folder
newblerInput	directory for newbler output files

Examples

```
## Not run:
mapReads2MDR('K00927', passDir="./out", newblerInput="./newbler2")

## End(Not run)
```

mdrRanges *Returns contig ranges for those captured within the MDR*

Description

Header for contigs contig00026 ref|YP_002288046.1| (ntRev) ## spanning:123 msaStart:653 msaEND:944 max10BPwindow:205.4 contigStart: 221 contigEnd: 421 Give GRanges output with seqname ko|contig00001

Usage

```
mdrRanges(ko, passDir)
```

Arguments

ko	the KO of interest
passDir	path to pAss outputs

Examples

```
## Not run:
locsMDR = mdrRanges('K00927')

## End(Not run)
```

nitrogenMetab *nitrogenMetab*

Description

Igraph object containing KOs and compounds involved in nitrogen metabolism

Usage

```
nitrogenMetab
```

Format

83 nodes and 251 edges. 50 KOs and 33 compounds

path2kingdom *List all intermediaries between taxa and the superkingdom it belongs to*

Description

Finds the superkingdom given the taxid

Usage

```
path2kingdom(taxID = "79255", ...)
```

Arguments

taxID	NCBI taxonomic id
...	additional dbquery paramaters

Examples

```
## Not run:
path2kingdom(taxidID='79255')
Sending query

Query Completed

              name   taxid         rank
1         Gordonia   79255        genus
2         Theaceae   27065       family
3         Ericales   41945        order
4         asterids   71274     subclass
5     Pentapetalae 1437201      no rank
6       Gunneridae   91827      no rank
7   eudicotyledons   71240      no rank
8  Mesangiospermae 1437183      no rank
9    Magnoliophyta    3398      no rank
10   Spermatophyta   58024      no rank
11   Euphyllophyta   78536      no rank
12    Tracheophyta   58023      no rank
13     Embryophyta    3193      no rank
14  Streptophytina  131221      no rank
15    Streptophyta   35493       phylum
16   Viridiplantae   33090      kingdom
17       Eukaryota    2759 superkingdom


## End(Not run)
```

path2ko *Finds all KOs in a given pathway Finds all KOs belonging to a Pathways*

Description

Finds all KOs in a given pathway Finds all KOs belonging to a Pathways

Usage

```
path2ko(pathway = "path:ko00010", ...)
```

Arguments

pathway	The pathway ID
...	additional paramters for dbquery

pathways *List Pathways Lists all metabolic pathways*

Description

List Pathways Lists all metabolic pathways

Usage

```
pathways(...)
```

Arguments

...	additional parameters for dbquery

prettifyGraph *returns metabolic graph with all the ornaments set Adds details into the igraph object*

Description

returns metabolic graph with all the ornaments set Adds details into the igraph object

Usage

```
prettifyGraph(g, vsize = c(1, 2), vcolor = c("grey", "red"),
  withText = TRUE, vtext = c(0.5, 1), layout = FALSE)
```

Arguments

g	igraph Object
vsize	the vertex size
vcolor	length 2 vector for KO and compounds
withText	Conditional to include labels in
vtext	two vector text valuevsize
layout	Whether to calculate the layoutvcolor

Examples

```
p = nitrogenMetab %>% prettifyGraph %>% ig2ggplot(., dfOnly=FALSE)
p
```

scg *singleCopyGenes'*

Description

Vector list showing Single Copy Genes

Usage

```
scg
```

Format

31 KEGG ids ko:KXXXXX

simpleThres *simpleThres*

Description

internal cluster

Usage

```
simpleThres(keptDF)
```

Arguments

keptDF	internal function

surrNODES *surrNODES finds nodes which are surrounding the given node*

Description

surrNODES finds nodes which are surrounding the given node

Usage

```
surrNODES(noi, graph, all = TRUE)
```

Arguments

noi	node of interest; the compound/ko ID not the vertex ID of the node in the graph
graph	igraph object
all	conditional to return all nodes regardless of direction

Value

vector of surrounding nodes if all is TRUE, if all is FALSE returns list of in and out nodes

taxnam.sql *taxname.sql*

Description

Faster way of querying the taxid names using structured SQL then querying the graphDB. Downloads a copy of NCBI's taxdump.tar.gz from ftp.ncbi.nlm.nih.gov/pub/taxonomy/taxdump.tar.gz and builds a taxid, name, rank table in sqlite3

Usage

```
taxnam.sql(taxids, byID = TRUE, taxrank = "genus")
```

Arguments

taxids	vector of integers/name representing the taxa of interest
byID	search by ID
taxrank	any of the common taxonomic ranks eg. genus

Examples

```
taxnam.sql(287)
```

taxname *Lists the taxonomic id's details*

Description

Lists the taxonomic id's details

Usage

```
taxname(taxon = 5062, name = FALSE, ...)
```

Arguments

taxon	The default is Aspergillus oryzae, accepts both name as well as
name	search by name
...	additional parameters for dbquery

top500kos *Top500kos*

Description

Dataset showing the top 500 KEGG Orthologs (KOS)

Usage

```
top500kos
```

Format

A vector of 500 KOs based on KOs ranked by abundance, not sure if the vector is ranked.

Index