oGraph

Xin He

2016-02-19

Overview

oGraph is a package for ontology representation and visualization.

Quick start guide

This section describes a simple working session using **oGraph**.

A typical session can be divided into two steps:

1. *Data preparation* build the oGraph object which has the ontology store in an underlying igraph object.
2. *Running analysis* treeview, subtree, find parent/child terms…

Here are example vignettes:

Load the package and see what ontology packages are available in you current .libPaths()

```r
library('ograph')
```

```
## ograph loaded.
## These ontology package(s) are currently available in your libPath:
## GOBP GOCC GOMF HDO150 HDOCORENIGO HDO HDOOLD HDOSHAREGENENIGO HPO NiGOBP NiGO RECTOMEPATHWAY SBO
```

```r
ograph::initWHAT()
```

```
## These ontology package(s) are currently available in your libPath:
## GOBP GOCC GOMF HDO150 HDOCORENIGO HDO HDOOLD HDOSHAREGENENIGO HPO NiGOBP NiGO RECTOMEPATHWAY SBO
```

init the **ontGraph** object.

```r
ontGraph<-new("ontGraph",ontology='HDO')
print(ontGraph)
```

```
##
## ------------------------- ontGraph object -------------------------
##
##  Ontology:
##    -  HDO
##
##  Graph:
## IGRAPH DN-- 6819 7030 --
## attr: name (v/c), def (v/c), description (v/c), level (v/n),
##   is_leaf (v/l), color (v/c), type (e/c)
##
##  levels:
```

```
##              Length Class  Mode
## nodes2level 6819   -none- list
## level2nodes   14   -none- list
## noOfLevels     1   -none- numeric
## noOfNodes      1   -none- numeric
```

The ontGraph object contains the 'DAG', 'level' and 'termid2def' mapping. You can access these properties by using '@'.

```
levels<-ontGraph@levels
head(levels$nodes2level)
```

```
## $`DOID:2722`
## [1] 7
##
## $`DOID:399`
## [1] 6
##
## $`DOID:2723`
## [1] 6
##
## $`DOID:9432`
## [1] 7
##
## $`DOID:2725`
## [1] 9
##
## $`DOID:11971`
## [1] 9
```

The most important object is the 'DAG' object.

```
g=ontGraph@graph
g
```

```
## IGRAPH DN-- 6819 7030 --
## + attr: name (v/c), def (v/c), description (v/c), level (v/n),
##   is_leaf (v/l), color (v/c), type (e/c)
```

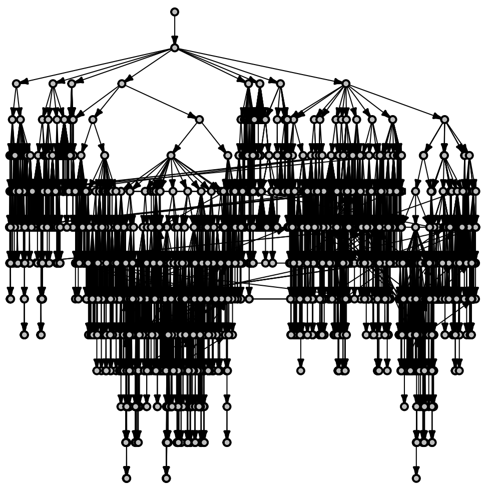
A complete list of functions can be found by running:

```
ls('package:ograph')
```

```
##  [1] "addFCurve"                          "analysis"
##  [3] "buildLevels"                        "buildOntGraph"
##  [5] "calculatePRF"                       "calculate.upper.adj.matrix"
##  [7] "calculate.upper.adj.matrix2"        "calculate.upper.adj.matrix3"
##  [9] "calculate.upper.adj.matrix.print" "colorMapNode"
## [11] "disease.2.disease"                  "entrez2symbol"
## [13] "findAllChildrenNodes"               "findAllParentNodes"
## [15] "findBiLink"                         "findChildrenNodes"
## [17] "findInducedSubGraphNodes"           "findLeafNode"
## [19] "findlink"                      "find.node.level"
## [21] "findParentNodes"               "findResult"
## [23] "findRoot"                      "get.node.attribute"
## [25] "init"                          "initialize"
## [27] "initOGraph"                    "initWHAT"
## [29] "is.leaf"                       "is.nodes.in.graph"
## [31] "loadGraph"                     "mapGene2Graph"
## [33] "mapGene2Graph2"                "node.addColorAttributeByLevel"
## [35] "node.addDefAttribute"          "node.addLeafAttribute"
## [37] "node.addLevelAttribute"        "nomalplot"
## [39] "peekNode"                      "plot2file"
## [41] "plot.graphNEL"                 "plotGraphStructure"
## [43] "plotSig"                       "plotWordcloud"
## [45] "print"                         "readMappings"
## [47] "reverseArch"                   "rollUpAnnotation"
## [49] "rollUpAnnotation2"             "rollUpToLevel"
## [51] "saveAnnotationFromGraph"       "saveGraph"
## [53] "searchDescription4Keyword"     "searchName4Keyword"
## [55] "set.node.attribute"            "set.node.color"
## [57] "shortest_path"                 "shortest_path_to_root"
## [59] "subGraphByLevel"               "subGraphByNodes"
## [61] "tktreeplot"                    "to.latex"
## [63] "to.latex.content"              "treeplot"
## [65] "which.node.in.graph"
```

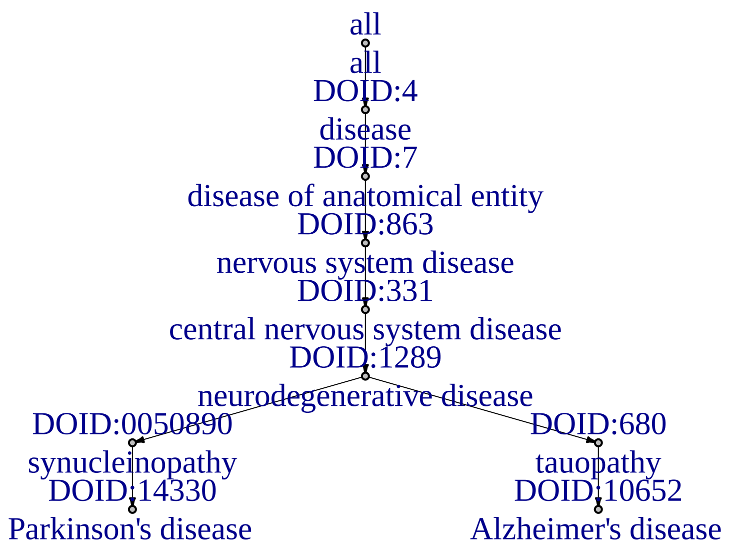
A set of functions are designed to manipulate this **igrph** object reporesenting the ontology 'DAG': First let's plot the graph

```
treeplot(g,label = F)
```



This graph is too big, we will use a subgraph that seeded from c('DOID:10652','DOID:14330')

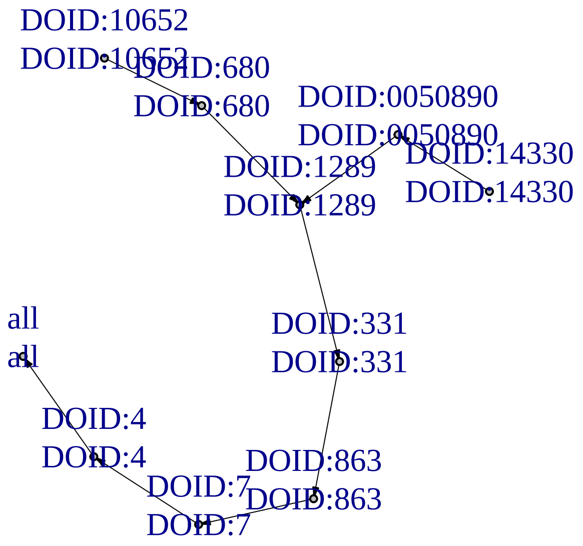
```
g<-subGraphByNodes(g,c('DOID:10652','DOID:14330'))
##plot again
treeplot(g,label = T)
```

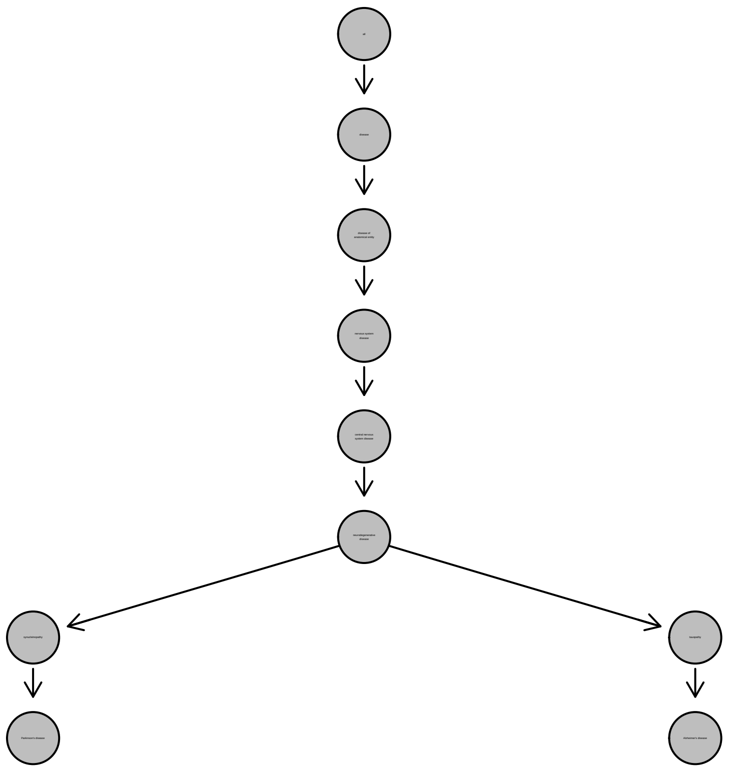


```
nomalplot(g,label = T)
plot.graphNEL(g,term2def = ontGraph@term2def)
```

```
## Loading required package: Rgraphviz
```

```
## Loading required package: grid
```





We can change the node attributes

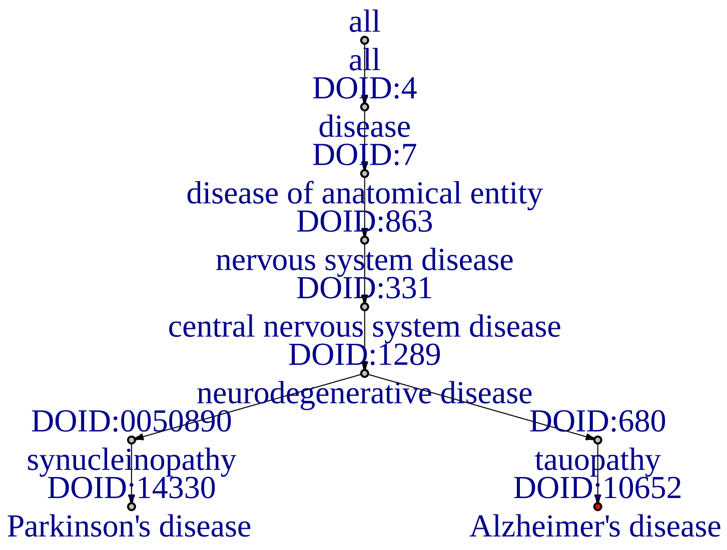
```r
list.vertex.attributes(g)
```

```
## [1] "name"        "def"         "description" "level"       "is_leaf"
## [6] "color"
```

```r
g<-set.node.attribute(graph = g,attr_name = 'color',attr_value = 'red',nodes = c('DOID:10652'))
get.node.attribute(g,'color')
```

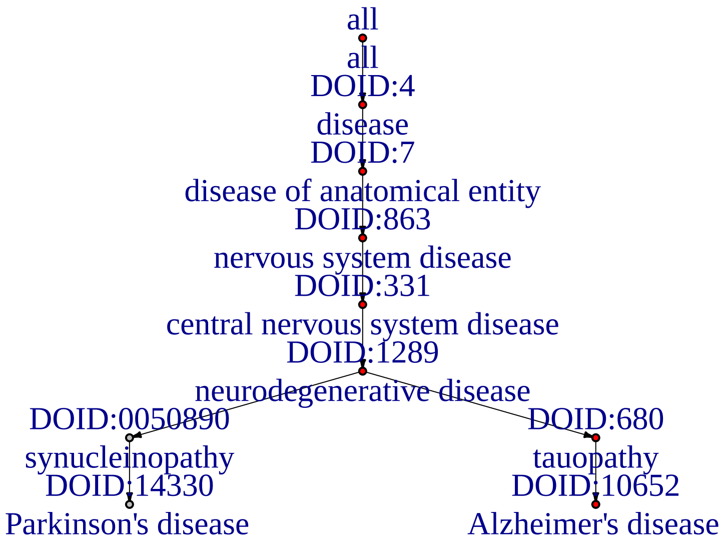
```
##  [1] "grey" "red"  "grey" "grey" "grey" "grey" "grey" "grey" "grey" "grey"
```

```r
##replot
treeplot(g,T)
```



We want to know the path between two node:

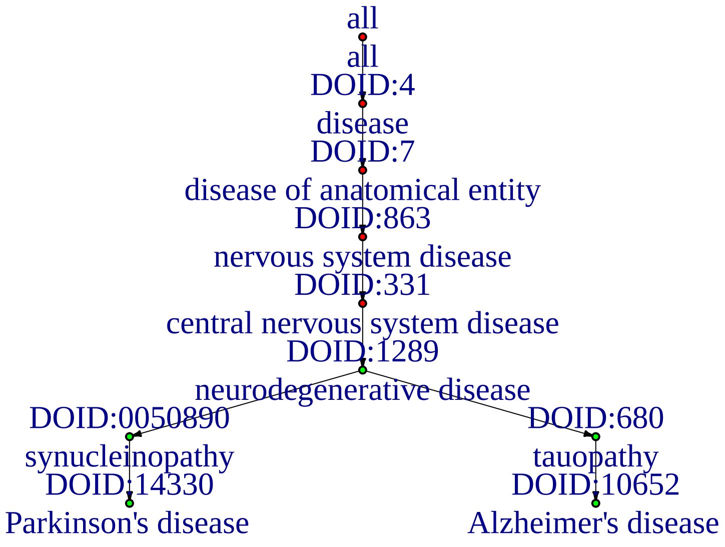
```r
##to root
nodes2root<-shortest_path_to_root(graph=g,node='DOID:10652')
g<-set.node.attribute(graph = g,attr_name = 'color',attr_value = 'red',nodes = nodes2root)
treeplot(g,T)
```



```
##to another node
node2node<-shortest_path(g,'DOID:10652','DOID:14330',levels = ontGraph@levels,self.includ = T)
##node2node contains two object, up and down. This is two route going from node A to node B. In this ca
##connecting these two node by going up the tree
node2node
```

```
## $up
## [1] "DOID:1289"    "DOID:680"     "DOID:10652"   "DOID:0050890"
## [5] "DOID:14330"
##
## $down
## list()
```

```
g<-set.node.attribute(graph = g,attr_name = 'color',attr_value = 'green',nodes = node2node$up)
treeplot(g,T)
```



We are interested in getting all the children/parents nodes:

```r
##direct parents/children
findChildrenNodes(g,'DOID:1289')
```

```
## [1] "DOID:0050890" "DOID:680"
```

```r
findParentNodes(g,'DOID:1289')
```

```
## [1] "DOID:331"
```

```r
##all parents/children
findAllChildrenNodes(g,'DOID:1289')
```

```
## [1] "DOID:0050890" "DOID:680"     "DOID:14330"   "DOID:10652"
```

```r
findAllParentNodes(g,'DOID:1289')
```

```
## [1] "DOID:331" "DOID:863" "DOID:7"   "DOID:4"   "all"
```

We want to create a sub 'DAG' with a set of interesting nodes. We are using a seed node c('DOID:150') and we need to work first on the full graph.

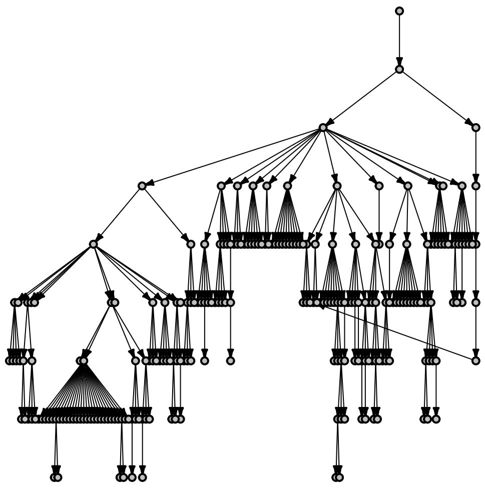
```r
g<-ontGraph@graph
nodes<-c('DOID:150')
cs<-findAllChildrenNodes(g,nodes)
cs
```

```
##   [1] "DOID:0060037" "DOID:0060043" "DOID:10935"   "DOID:10937"
##   [5] "DOID:1234"    "DOID:1510"    "DOID:1561"    "DOID:1766"
##   [9] "DOID:303"     "DOID:4737"    "DOID:507"     "DOID:535"
##  [13] "DOID:0060038" "DOID:0060040" "DOID:0050696" "DOID:0050856"
##  [17] "DOID:1059"    "DOID:1094"    "DOID:12995"   "DOID:2033"
##  [21] "DOID:2303"    "DOID:2769"    "DOID:8670"    "DOID:8927"
##  [25] "DOID:9923"    "DOID:0050665" "DOID:0050666" "DOID:0050667"
##  [29] "DOID:0050668" "DOID:0050888" "DOID:0050889" "DOID:0060179"
##  [33] "DOID:0060309" "DOID:0050776" "DOID:0060307" "DOID:0060308"
##  [37] "DOID:12685"   "DOID:4090"    "DOID:92"      "DOID:93"
##  [41] "DOID:0060130" "DOID:0060131" "DOID:0060132" "DOID:0060133"
##  [45] "DOID:0060134" "DOID:0060135" "DOID:0060136" "DOID:0060137"
##  [49] "DOID:0060138" "DOID:0060139" "DOID:0060140" "DOID:0060141"
##  [53] "DOID:0060142" "DOID:0060143" "DOID:0060144" "DOID:0060145"
##  [57] "DOID:0060146" "DOID:0060147" "DOID:0060148" "DOID:0060149"
##  [61] "DOID:0060150" "DOID:0060151" "DOID:0060152" "DOID:0060154"
##  [65] "DOID:0060155" "DOID:13417"   "DOID:4970"    "DOID:4260"
##  [69] "DOID:4627"    "DOID:0060153" "DOID:0060156" "DOID:4186"
##  [73] "DOID:4188"    "DOID:4189"    "DOID:0060243" "DOID:0060046"
##  [77] "DOID:0060244" "DOID:11385"   "DOID:4541"    "DOID:10600"
##  [81] "DOID:11119"   "DOID:2768"    "DOID:11507"   "DOID:12128"
##  [85] "DOID:12129"   "DOID:8689"    "DOID:0060047" "DOID:12568"
##  [89] "DOID:13365"   "DOID:0060223" "DOID:4540"    "DOID:4428"
##  [93] "DOID:0060041" "DOID:1206"    "DOID:13487"   "DOID:0050432"
##  [97] "DOID:0060042" "DOID:12849"   "DOID:0060044" "DOID:10132"
## [101] "DOID:10817"   "DOID:10849"   "DOID:1876"    "DOID:10236"
## [105] "DOID:10834"   "DOID:1235"    "DOID:13351"   "DOID:9336"
## [109] "DOID:1233"    "DOID:10131"   "DOID:11120"   "DOID:13709"
## [113] "DOID:13868"   "DOID:1875"    "DOID:4762"    "DOID:10934"
## [117] "DOID:11037"   "DOID:11038"   "DOID:0050587" "DOID:12399"
## [121] "DOID:12400"   "DOID:12401"   "DOID:12402"   "DOID:2510"
## [125] "DOID:10919"   "DOID:13352"   "DOID:10646"   "DOID:10930"
## [129] "DOID:10931"   "DOID:10932"   "DOID:10936"   "DOID:10938"
## [133] "DOID:10939"   "DOID:1509"    "DOID:2745"    "DOID:334"
## [137] "DOID:10914"   "DOID:1307"    "DOID:2030"    "DOID:2468"
## [141] "DOID:3324"    "DOID:13027"   "DOID:4543"    "DOID:5340"
## [145] "DOID:12217"   "DOID:8725"    "DOID:10685"   "DOID:10933"
## [149] "DOID:14320"   "DOID:2055"    "DOID:4964"    "DOID:591"
## [153] "DOID:594"     "DOID:6088"    "DOID:11257"   "DOID:593"
## [157] "DOID:599"     "DOID:0060048" "DOID:600"     "DOID:605"
## [161] "DOID:602"     "DOID:603"     "DOID:6950"    "DOID:11328"
## [165] "DOID:5418"    "DOID:5419"    "DOID:778"     "DOID:8646"
## [169] "DOID:1229"    "DOID:6680"    "DOID:1203"    "DOID:251"
## [173] "DOID:8645"    "DOID:1742"    "DOID:252"     "DOID:12139"
## [177] "DOID:12294"   "DOID:1596"    "DOID:3312"    "DOID:0060167"
## [181] "DOID:1595"    "DOID:2848"    "DOID:1470"    "DOID:9478"
## [185] "DOID:0060166" "DOID:14042"   "DOID:845"     "DOID:0060045"
## [189] "DOID:0060001" "DOID:302"     "DOID:9973"    "DOID:9828"
## [193] "DOID:11206"   "DOID:11718"   "DOID:12797"   "DOID:1574"
## [197] "DOID:5062"    "DOID:670"     "DOID:809"     "DOID:8519"
## [201] "DOID:9505"    "DOID:0050741" "DOID:0050742" "DOID:9974"
## [205] "DOID:2559"    "DOID:2575"    "DOID:9975"    "DOID:9977"
## [209] "DOID:2560"    "DOID:9976"    "DOID:1849"    "DOID:0060163"
## [213] "DOID:0060164" "DOID:11569"   "DOID:12883"   "DOID:13918"
## [217] "DOID:1768"    "DOID:0050628" "DOID:0050847" "DOID:2846"
## [221] "DOID:8619"    "DOID:8986"    "DOID:9091"    "DOID:9207"
## [225] "DOID:0050848" "DOID:9220"    "DOID:0060165"
```

```r
sub<-subGraphByNodes(g,cs)
sub
```

```
## IGRAPH DN-- 235 235 --
## + attr: name (v/c), def (v/c), description (v/c), level (v/n),
##   is_leaf (v/l), color (v/c), type (e/c)
```

```r
treeplot(sub,F)
```



We want to take all the node in this sub 'DAG' for clipping?

```
get.node.attribute(sub,attr = 'name')
```

```
##   [1] "DOID:0050432" "DOID:0050587" "DOID:0050628" "DOID:0050665"
##   [5] "DOID:0050666" "DOID:0050667" "DOID:0050668" "DOID:0050696"
##   [9] "DOID:0050741" "DOID:0050742" "DOID:0050776" "DOID:0050847"
##  [13] "DOID:0050848" "DOID:0050856" "DOID:0050888" "DOID:0050889"
##  [17] "DOID:0050890" "DOID:0060001" "DOID:0060037" "DOID:0060038"
##  [21] "DOID:0060040" "DOID:0060041" "DOID:0060042" "DOID:0060043"
##  [25] "DOID:0060044" "DOID:0060045" "DOID:0060046" "DOID:0060047"
##  [29] "DOID:0060048" "DOID:0060130" "DOID:0060131" "DOID:0060132"
##  [33] "DOID:0060133" "DOID:0060134" "DOID:0060135" "DOID:0060136"
##  [37] "DOID:0060137" "DOID:0060138" "DOID:0060139" "DOID:0060140"
##  [41] "DOID:0060141" "DOID:0060142" "DOID:0060143" "DOID:0060144"
##  [45] "DOID:0060145" "DOID:0060146" "DOID:0060147" "DOID:0060148"
##  [49] "DOID:0060149" "DOID:0060150" "DOID:0060151" "DOID:0060152"
##  [53] "DOID:0060153" "DOID:0060154" "DOID:0060155" "DOID:0060156"
##  [57] "DOID:0060163" "DOID:0060164" "DOID:0060165" "DOID:0060166"
##  [61] "DOID:0060167" "DOID:0060179" "DOID:0060223" "DOID:0060243"
##  [65] "DOID:0060244" "DOID:0060307" "DOID:0060308" "DOID:0060309"
##  [69] "DOID:10131"   "DOID:10132"   "DOID:10236"   "DOID:1059"
##  [73] "DOID:10600"   "DOID:10646"   "DOID:10685"   "DOID:10817"
##  [77] "DOID:10834"   "DOID:10849"   "DOID:10914"   "DOID:10919"
##  [81] "DOID:10930"   "DOID:10931"   "DOID:10932"   "DOID:10933"
##  [85] "DOID:10934"   "DOID:10935"   "DOID:10936"   "DOID:10937"
##  [89] "DOID:10938"   "DOID:10939"   "DOID:1094"    "DOID:11037"
##  [93] "DOID:11038"   "DOID:11119"   "DOID:11120"   "DOID:11206"
##  [97] "DOID:11257"   "DOID:11328"   "DOID:11385"   "DOID:11507"
## [101] "DOID:11569"   "DOID:11718"   "DOID:1203"    "DOID:1206"
## [105] "DOID:12128"   "DOID:12129"   "DOID:12139"   "DOID:12217"
## [109] "DOID:1229"    "DOID:12294"   "DOID:1233"    "DOID:1234"
## [113] "DOID:1235"    "DOID:12399"   "DOID:12400"   "DOID:12401"
## [117] "DOID:12402"   "DOID:12568"   "DOID:12685"   "DOID:12797"
## [121] "DOID:12849"   "DOID:12883"   "DOID:1289"    "DOID:12995"
## [125] "DOID:13027"   "DOID:1307"    "DOID:13351"   "DOID:13352"
## [129] "DOID:13365"   "DOID:13417"   "DOID:13487"   "DOID:13709"
## [133] "DOID:13868"   "DOID:13918"   "DOID:14042"   "DOID:14320"
## [137] "DOID:1470"    "DOID:150"     "DOID:1509"    "DOID:1510"
## [141] "DOID:1561"    "DOID:1574"    "DOID:1595"    "DOID:1596"
## [145] "DOID:1742"    "DOID:1766"    "DOID:1768"    "DOID:1849"
## [149] "DOID:1875"    "DOID:1876"    "DOID:2030"    "DOID:2033"
## [153] "DOID:2055"    "DOID:2303"    "DOID:2468"    "DOID:251"
## [157] "DOID:2510"    "DOID:252"     "DOID:2559"    "DOID:2560"
## [161] "DOID:2575"    "DOID:2745"    "DOID:2768"    "DOID:2769"
## [165] "DOID:2846"    "DOID:2848"    "DOID:302"     "DOID:303"
## [169] "DOID:331"     "DOID:3312"    "DOID:3324"    "DOID:334"
## [173] "DOID:4"       "DOID:4090"    "DOID:4186"    "DOID:4188"
## [177] "DOID:4189"    "DOID:4260"    "DOID:4428"    "DOID:4540"
## [181] "DOID:4541"    "DOID:4543"    "DOID:4627"    "DOID:4737"
## [185] "DOID:4762"    "DOID:4964"    "DOID:4970"    "DOID:5062"
## [189] "DOID:507"     "DOID:5340"    "DOID:535"     "DOID:5418"
## [193] "DOID:5419"    "DOID:591"     "DOID:593"     "DOID:594"
## [197] "DOID:599"     "DOID:600"     "DOID:602"     "DOID:603"
## [201] "DOID:605"     "DOID:6088"    "DOID:6680"    "DOID:670"
## [205] "DOID:6950"    "DOID:7"       "DOID:778"     "DOID:809"
## [209] "DOID:845"     "DOID:8519"    "DOID:8619"    "DOID:863"
## [213] "DOID:8645"    "DOID:8646"    "DOID:8670"    "DOID:8689"
## [217] "DOID:8725"    "DOID:8927"    "DOID:8986"    "DOID:9091"
## [221] "DOID:92"      "DOID:9207"    "DOID:9220"    "DOID:93"
## [225] "DOID:9336"    "DOID:9478"    "DOID:9505"    "DOID:9828"
## [229] "DOID:9923"    "DOID:9973"    "DOID:9974"    "DOID:9975"
## [233] "DOID:9976"    "DOID:9977"    "all"
```

Other functions:

```
findRoot(graph=g)
```

```
## [1] "all"
```

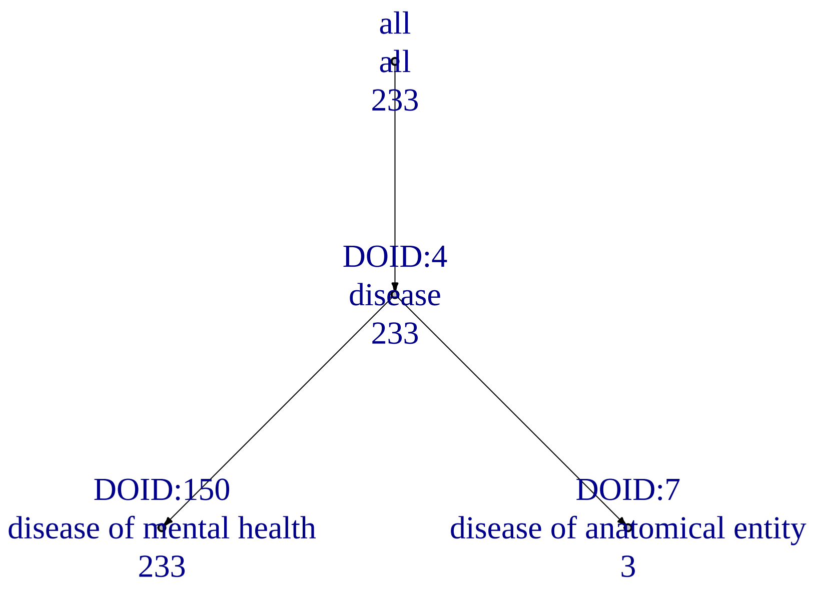
```
leaves<-findLeafNode(graph=g)
subg<-subGraphByLevel(graph=g,l=3)
subg
```

```
## IGRAPH DN-- 10 9 --
## + attr: name (v/c), def (v/c), description (v/c), level (v/n),
##   is_leaf (v/l), color (v/c), type (e/c)
```

```
sub=mapGene2Graph(graph=sub,file=system.file("extdata","human_gene2HDO_o", package ="ograph"),rollup =
```

```
## rolling up annotation of level 9 / 9
## rolling up annotation of level 8 / 9
## rolling up annotation of level 7 / 9
## rolling up annotation of level 6 / 9
## rolling up annotation of level 5 / 9
## rolling up annotation of level 4 / 9
## rolling up annotation of level 3 / 9
## rolling up annotation of level 2 / 9
## rolling up annotation of level 1 / 9
```

```
treeplot(subGraphByLevel(sub,3),label=1,show.genes=TRUE)
```



```
term='DOID:150'
get.node.attribute(sub,attr='def',nodes=term)
```

```
## [1] "disease of mental health"
```

```
peekNode(sub,node=term)
```

```
## name:DOID:150
##
## def:disease of mental health
##
## description:"A disease that involves a psychological or behavioral pattern generally associated with
##
## level:3
##
## is_leaf:FALSE
##
## color:grey
##
## genes:
```

We can search keyword in ontology terms.

```
clip_neuro<-unique(c(searchDescription4Keyword(graph=g,keys=c('neuro')),searchName4Keyword(g,keys=c('ne
head(get.node.attribute(g,attr='def',clip_neuro))
```

```
## [1] "intestinal botulism"                "Lambert-Eaton myasthenic syndrome"
## [3] "granulomatous amebic encephalitis" "baylisascariasis"
## [5] "foodborne botulism"                 "wound botulism"
```

```
clip_brain=unique(c(searchDescription4Keyword(graph=g,keys=c('brain')),searchName4Keyword(g,keys=c('bra
head(get.node.attribute(g,attr='def',clip_neuro))
```

```
## [1] "intestinal botulism"                 "Lambert-Eaton myasthenic syndrome"
## [3] "granulomatous amebic encephalitis" "baylisascariasis"
## [5] "foodborne botulism"                 "wound botulism"
```

```
g.clip=subGraphByNodes(g,nodes=unique(c(clip_brain,clip_neuro)))
g.clip=set.node.attribute(graph=g.clip,attr_name='color',attr_value='green',nodes=clip_neuro)
g.clip=set.node.attribute(graph=g.clip,attr_name='color',attr_value='blue',nodes=clip_brain)
g.clip=set.node.attribute(graph=g.clip,attr_name='color',attr_value='red',nodes=intersect(clip_brain,cl
treeplot(g.clip,label=0)
```