How To Use SubpathwayMiner

Chunquan Li

September 2, 2009

Contents

1 Overview

This vignette demonstrate how to easily annotate genes to pathways or sub-pathways using the SubpathwayMiner package. To do this, let us generate an example of gene sets:

```
> geneList <- getAexample(k = 100)
> geneList[1:10]
```

```
 [1] "10"        "100"       "1000"      "10000"     "10005"     "10007"
 [7] "1001"      "100128525" "100130247" "100130561"
```

2 A simple example of annotating genes to pathways

Annotate a set of genes to pathways.

```
> geneList <- getAexample(k = 100)
> ann <- getAnn(geneList)
> result <- printAnn(ann)
```

Display 10 rows and 4 columns of results.

```
> result[1:10, 2:5]
```

```
           annGeneRatio annBgRatio                pvalue                qvalue
path:04110       12/100  119/24143  8.5043083686287e-14 1.35882615841157e-11
path:02010        8/100   44/24143 1.02069463991938e-11 8.15437609006045e-10
path:05222        7/100   86/24143 6.95087003510508e-08 3.50471159584496e-06
path:05200       11/100  328/24143 1.18920437941483e-07 4.75030404708571e-06
path:04810        9/100  217/24143 3.03943440305154e-07 9.71287209875354e-06
path:05220        6/100   75/24143 6.93495459502813e-07 1.84678896520311e-05
path:05214        5/100   65/24143 7.47568160008871e-06 0.000151711013930438
path:04115        5/100   69/24143 1.00392210432565e-05 0.000186924364824829
path:05218        5/100   71/24143 1.15546533421274e-05 0.000205134784917809
path:04120        6/100  137/24143 2.30227032608221e-05 0.000367858855442667
```

Annotate a set of genes to sub-pathways of metabolic pathways based on enzyme commission (EC) numbers.

```
> geneList <- getAexample(k = 100)
> ann <- getKcsmpAnn(geneList, k = 4)
> printAnn(ann)[1:10, 2:5]
```

```
             annGeneRatio annBgRatio               pvalue            qvalue
path:00272_1        3/100   44/24143 0.000807238137687127 0.312031195904082
path:00510_3        3/100   65/24143  0.00250023677917233 0.312031195904082
path:00510_1        3/100   69/24143  0.00296343071282412 0.312031195904082
path:00363_1        2/100   32/24143  0.00776968586125126 0.312031195904082
path:00906_1        3/100   99/24143   0.0081138382762106 0.312031195904082
path:00622_1        1/100    2/24143  0.00826698944909798 0.312031195904082
path:00622_2        1/100    2/24143  0.00826698944909798 0.312031195904082
path:00622_3        1/100    2/24143  0.00826698944909798 0.312031195904082
path:00622_4        1/100    2/24143  0.00826698944909798 0.312031195904082
path:00643_2        1/100    2/24143  0.00826698944909798 0.312031195904082
```

Annotate a set of genes to sub-pathways based on KEGG Orthology (KO) identifiers.

```
> geneList <- getAexample(k = 100)
> subGraphList <- getKcSubGraph(k = 4, graphList = getDefaultKOUndirectedGraph())
> ann <- getKOAnn(geneList, graphList = subGraphList)
> printAnn(ann)[1:10, 2:5]
```

```
             annGeneRatio annBgRatio               pvalue              qvalue
path:04810_1         7/100   40/24143 2.81666912016476e-10 1.47809046530489e-07
path:04810_4         7/100   42/24143 4.04846378643242e-10 1.47809046530489e-07
path:04810_3         7/100   43/24143 4.81937156848744e-10 1.47809046530489e-07
path:04810_9         7/100   44/24143 5.71183433883959e-10 1.47809046530489e-07
path:04810_10        7/100   45/24143 6.74122424548784e-10 1.47809046530489e-07
path:05212_7         3/100    4/24143 2.74973428804337e-07 5.02425559175355e-05
path:05220_4         4/100   20/24143 1.27568419372448e-06 0.000199791671450944
path:04110_2         4/100   23/24143 2.30936807410487e-06 0.000315195695547961
path:05200_3         4/100   25/24143 3.27817861878188e-06 0.000399321183572774
path:04110_3         4/100   30/24143 6.98985360936266e-06 0.000766302647534033
```

3 Annotate genes to pathways

The function `getAnn` and `getKOAnn` in the SubpathwayMiner package not only facilitates the annotation and identification of pathways but also sub-pathway annotation and identification. It can annotate a set of genes to entire pathways or sub-pathways by setting the value of the argument `graphList`. The return value of the function is a list of the annotated information.The list has eight elements: 'pathwayName', 'annGeneList', 'annGeneNumber', 'annBgNumber', 'geneNumber', 'bgNumber', 'pvalue', 'qvalue'. They represent pathway name, genes annotated to the pathway, number of genes annotated to the pathway, number of background genes annoted to the pathway, number of genes in the study, number of background genes, p-value, and FDR-corrected q-value.

3.1 Annotate gene sets to entire pathways

If the value of argument `graphList` in the function `getAnn` is the return value of the function `getDefaultGraph`, these genes will be annotated to all pathways. Of course, this is the default setting of the function `getAnn`.

The code below can annotate a set of genes to pathways.

```
> ann <- getAnn(geneList)
> ann[1:2]

$`path:04110`
$`path:04110`$pathwayName
[1] "Cell cycle"

$`path:04110`$annGeneList
 [1] "100131844" "1017"      "1019"      "1021"      "1022"      "1026"
 [7] "1027"      "1028"      "1029"      "1030"      "1031"      "1032"

$`path:04110`$annGeneNumber
[1] 12

$`path:04110`$annBgNumber
[1] 119

$`path:04110`$geneNumber
[1] 100

$`path:04110`$bgNumber
[1] 24143

$`path:04110`$pvalue
[1] 8.504308e-14

$`path:04110`$qvalue
[1] 1.358826e-11


$`path:02010`
$`path:02010`$pathwayName
[1] "ABC transporters"

$`path:02010`$annGeneList
[1] "10057" "10058" "10060" "10257" "10347" "10349" "10350" "10351"
```

```
$`path:02010`$annGeneNumber
[1] 8

$`path:02010`$annBgNumber
[1] 44

$`path:02010`$geneNumber
[1] 100

$`path:02010`$bgNumber
[1] 24143

$`path:02010`$pvalue
[1] 1.020695e-11

$`path:02010`$qvalue
[1] 8.154376e-10
```

3.2 Annotate gene sets to sub-pathways of metabolic pathways based on enzyme commission (EC)

If the value of argument `graphList` is a list of subGraph, e.g., the return value of `getKcSubGraph`, these genes will be annotated to sub-pathways of metabolic pathways.

```
> subGraphList <- getKcSubGraph(k = 4)
> ann <- getAnn(geneList, graphList = subGraphList)
```

we also provide a simple function for the sub-pathway annotation of metabolic pathways.

```
> ann <- getKcsmpAnn(geneList, k = 4)
```

3.3 Annotate gene sets to sub-pathways based on KEGG Orthology (KO)

If the value of argument `graphList` is a list of subGraph, e.g., the return value of `getKcSubGraph` when setting arguments `graphList=getDefaultKOUndirectedGraph`, these genes will be annotated to sub-pathways based on KO.

```
> subGraphList <- getKcSubGraph(k = 4, graphList = getDefaultKOUndirectedGraph())
> ann <- getKOAnn(geneList, graphList = subGraphList)
```

3.4 Identify pathways or sub-pathways

Get the statistically signicantly enriched pathways according to pvalue.

```
> ann <- getAnn(geneList)
> cutedAnn <- cutoffAnn(ann, "pvalue", "<", 1e-04)
> printAnn(cutedAnn)[2:5]
```

```
           annGeneRatio annBgRatio               pvalue               qvalue
path:04110       12/100  119/24143  8.5043083686287e-14 1.35882615841157e-11
path:02010        8/100   44/24143 1.02069463991938e-11 8.15437609006045e-10
path:05222        7/100   86/24143 6.95087003510508e-08 3.50471159584496e-06
path:05200       11/100  328/24143 1.18920437941483e-07 4.75030404708571e-06
path:04810        9/100  217/24143 3.03943440305154e-07 9.71287209875354e-06
path:05220        6/100   75/24143 6.93495459502813e-07 1.84678896520311e-05
path:05214        5/100   65/24143 7.47568160008871e-06 0.000151711013930438
path:04115        5/100   69/24143 1.00392210432565e-05 0.000186924364824829
path:05218        5/100   71/24143 1.15546533421274e-05 0.000205134784917809
path:04120        6/100  137/24143 2.30227032608221e-05 0.000367858855442667
path:05223        4/100   54/24143 7.47448190139277e-05  0.00104626649102234
path:05010        6/100  177/24143 9.61447896431489e-05  0.00128017519178884
```

4 Display and save results

4.1 Use data frame to display results

To visualize the results, the list of results returned from the function `getAnn` or `getKOAnn` can be converted to the `data.frame` by using the function `printAnn`. But, note that Compared with `data.frame`, the `list` provides more information, e.g., the annotated genes are saved in list, yet not in the data.frame. The row names data.frame are pathway identifiers, e.g, path:00010. It's columns include pathwayName, annGeneRatio, annBgRatio, pvalue, qvalue. The annGeneRatio is the ratio of the annotated genes, e.g., 30/1000 means that 30 genes in 1000 genes are annotated. The qvalue is the FDR-corrected q-value.

```
> ann <- getAnn(geneList)
> result <- printAnn(ann)
> result[1:10, 2:5]
```

4.2 Save annotation results to a tab-delimited file

One can easily save the annotation results to a tab-delimited file. Note that the argument `col.names`=NA is essential.

```
> geneList <- getAexample(k = 1000)
> ann <- getAnn(geneList)
> result <- printAnn(ann)
> write.table(result, file = "result", col.names = NA, sep = "\t")
```

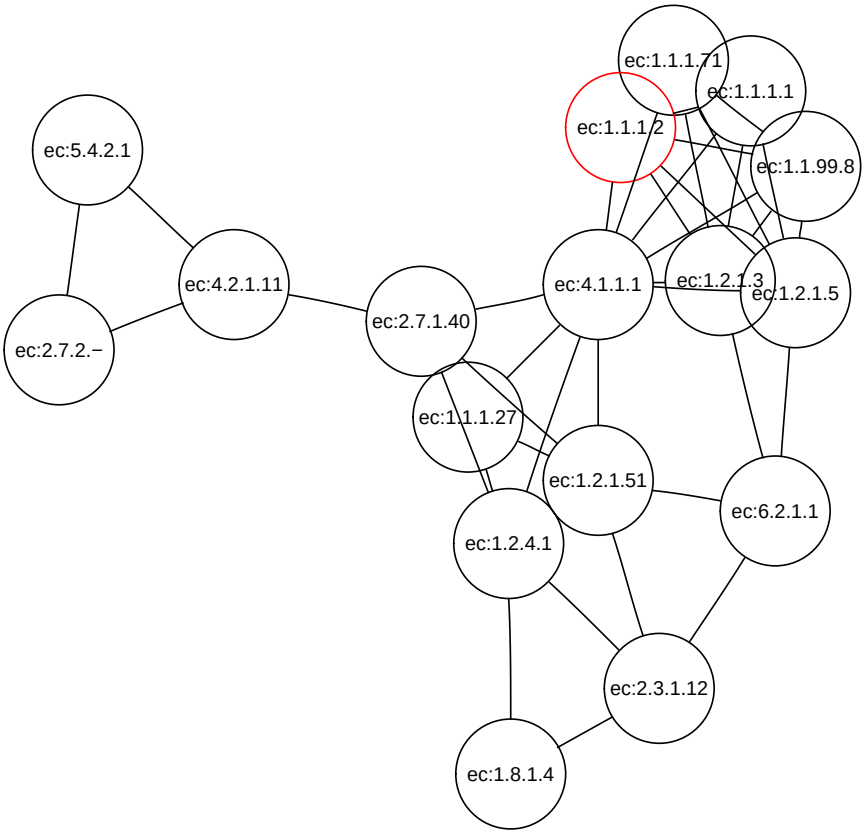
5 Visualization of pathways

5.1 Visualize sub-pathways of metabolic pathways based on enzyme commission (EC) using the function plotAnn

Users can use the function `plotAnn` to visualize the pathways or sub-pathways of metabolic pathways based on ec. The red nodes in the result graph represent the enzymes which include the submitted genes.

Visualize sub-pathways of metabolic pathways based on EC.

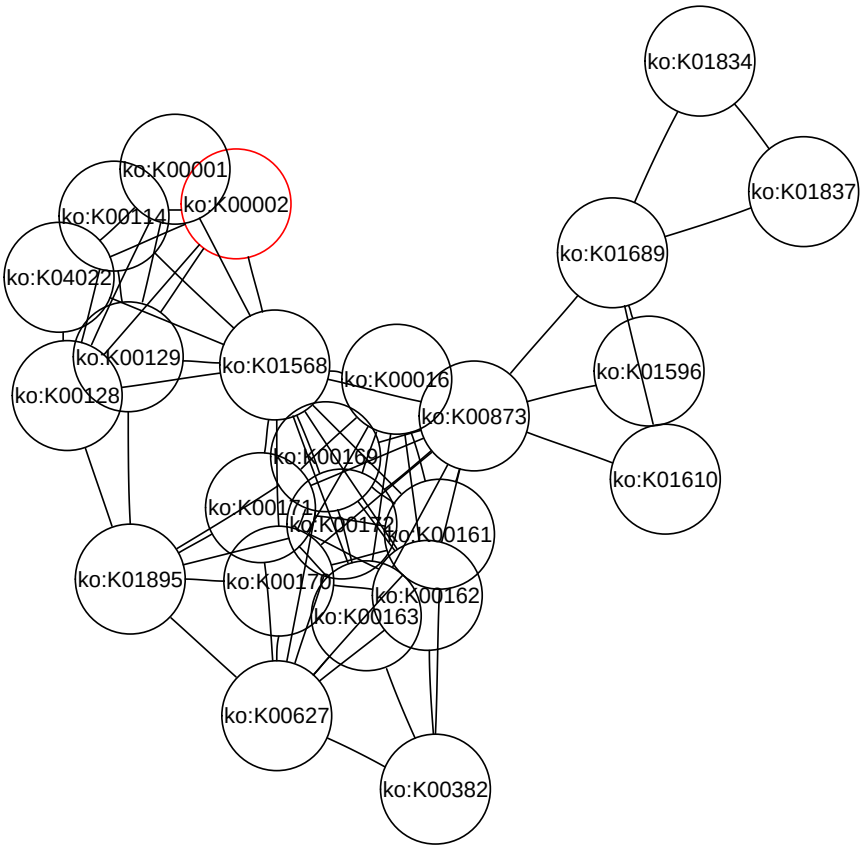
```
> subGraphList <- getKcSubGraph(k = 4)
> ann <- getAnn(geneList, graphList = subGraphList)
> plotAnn("path:00010_1", subGraphList, ann)
```



5.2 Visualize sub-pathways based on KEGG Orthology (KO) using the function plotKOAnn

Visualize sub-pathways based on KO.

```
> subGraphList <- getKcSubGraph(k = 4, graphList = getDefaultKOUndirectedGraph())
> ann <- getKOAnn(geneList, graphList = subGraphList)
> plotKOAnn("path:00010_1", subGraphList, ann)
```



5.3 Visualize pathways or sub-pathways through linking to KEGG web site

```
> subGraphList <- getKcSubGraph(k = 4)
> ann <- getAnn(geneList, graphList = subGraphList)
> gotoKEGG("path:00010_1", ann)

> subGraphList <- getKcSubGraph(k = 4, graphList = getDefaultKOUndirectedGraph())
> ann <- getKOAnn(geneList, graphList = subGraphList)
> gotoKEGG("path:00010_1", ann)
```

Visualize pathways.

```
> ann <- getAnn(geneList)
> gotoKEGG("path:00010", ann)
```

6 How to set organism and gene identifier

Users that want to annotate genes to pathways or sub-pathways should ensure that the type of organism and gene identifiers accord with the return value of the function `getOrgAndIdType` that can check the type of organism and identifier in the current study. You can do:

```
> getOrgAndIdType()
```

```
[1] "hsa"          "ncbi-geneid"
```

The return values mean that the type of organism and identifier in the current study are Homo sapiens and Entrez gene identifiers. If they are different from the type of your genes, you need to change them with some functions, e.g., `updateOrgAndIdType`, `data`, `loadKe2g`.

6.1 Set or update the organism and the type of gene identifier

The existing tools mainly use DBMS (data base management system) to store all data relative to analysis of pathways and the update process of the data is transparent to users, which means that the annotation results users get from these tools may become outdated. We don't use DBMS to store data. We present a new method that enables users to update data by themselves. Users are firstly required to set organism and type of gene identifier before annotateing genes to the pathways. According to the setting, the system can download all data relative to analysis of pathways in the organism, and then treat and store them in an environment variable in R. Through the method the system can synchronize the data with the KEGG GENE database and support most organisms and cross reference identifiers in the KEGG GENE database.

The code below means that the type of organism and identifier in the current study are setted as Saccharomyces cerevisiae and sgd identifier in Saccharomyces Genome Database. When we run it, the system will download all data relative to analysis of pathways in the organism, and then treat and store them in an environment variable in R. Finally, Users can use our system to annotate and identify pathways or sub-pathways.

```
> updateOrgAndIdType("sce", "sgd-sce")
```

6.2 Load and save the environment variable of the system

We have considered that our method to store and update data may be time consuming for large organisms that have many genes in common. Thus, the system provide two functions to easily save and load the environment variable of the system, which make users update all data relative to analysis of pathways in the organism one time only and repeatedly use them in the future.

The code below is used to save the environment variable of Saccharomyces cerevisiae. Note that the data is saved to the working directory.

```
> saveKe2g("sce_sgd-sce.rda")
```

When one needs to use the environment variables of Saccharomyces cerevisiae next time, one can use the function loadKe2g to load the last environment variable.

The code below is used to load the environment variables of Saccharomyces cerevisiae. Note that you need to set your working directory to the directory of the data file.

```
> loadKe2g("sce_sgd-sce.rda")
```

6.3 Select the organism provided by the system

The environment variables of organisms with well annotated genomes are provided by the system and users can use the function `data` to load them.

The code below is used to load the environment variables of Saccharomyces cerevisiae provided by our system. the type of gene identifier is ncbi-geneid.

```
> data("sce_ncbi-geneid")
```

7 Use our flexible model to annotate genes to user-defined sub-pathways

Our system provides a flexible model for supporting the user-defined sub-pathways. To date, many algorithms in concepts of graph are vailable in the R packages (Huber et al., 2007). Through our model users can use easily these algorithms to annotate genes to the sub-pathways themselves.

7.1 Simplification version of metabolic pathways

Generally, A metabolic pathway can be considered as a graph with chemical compounds as nodes and enzymes as edges. We simplify metabolic pathways. Each metabolic pathway is converted to an undirected graph with enzymes as nodes. Two enzymes are connected by an edge if their corresponding reactions have a common compound. Chemical compounds are then omitted from graphs. If we consider the direction of reaction. The pathway will be a directed graph. We use the XML package to take out the relationship of enzymes from the XML version of the metabolic pathway maps, and then save simplification version of metabolic pathways to a list of *graph*.

The code below can get the data from the environment variable of the system.

```
> uGraph <- getDefaultUndirectedGraph()
> uGraph[1:2]
```

```
$`path:00010`
A graphNEL graph with undirected edges
Number of Nodes = 40
Number of Edges = 127

$`path:00020`
A graphNEL graph with undirected edges
Number of Nodes = 23
Number of Edges = 73
```

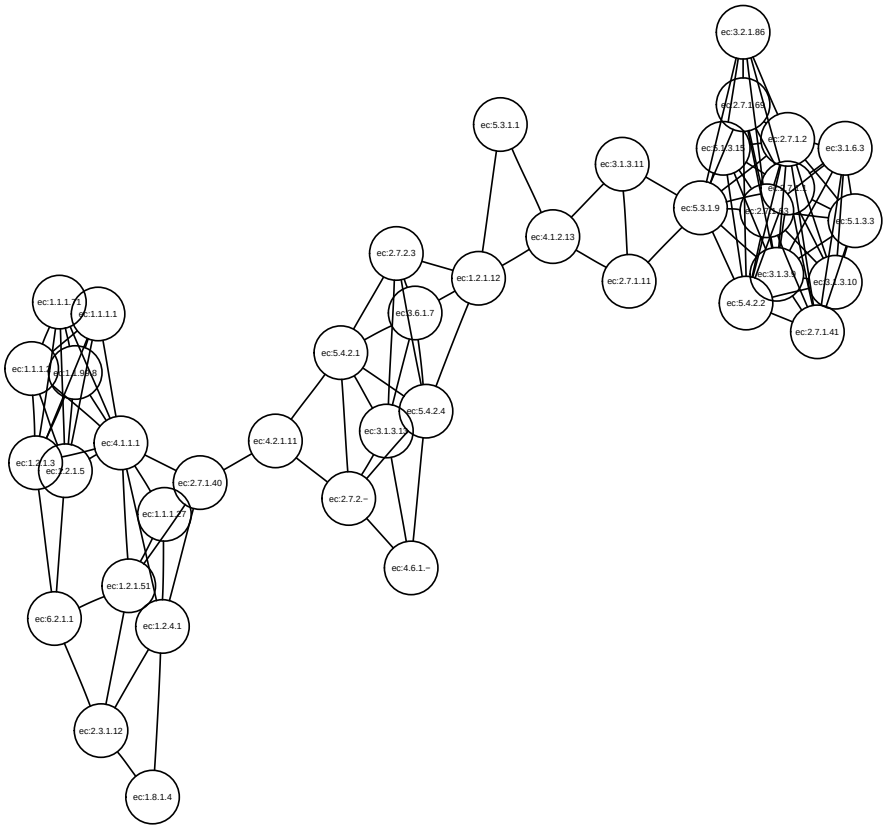
The return value of the function `getDefaultUndirectedGraph` is a list of *graph*. The first graph in the list is the graph representation of the pathway "path:00010". The pathway's name is Glycolysis / Gluconeogenesis. One can use the function `getPathwayNameFromId` to get it.

```
> getPathwayNameFromId("path:00010")

                          00010
"Glycolysis / Gluconeogenesis"
```

One can also use the function `plot` to display the graph.

```
> plot(uGraph$"path:00010", "neato")
```



You can now see that each pathway of metabolic pathways is converted to a graph with enzymes as nodes. All graphs are saved in a list. each element in the list is a *graph* and its name is pathway identifier.

7.2 Create a subGraph with the algorithms based on the concepts of graph

Users can mine sub-pathways of metabolic pathways by using certain sub-graph mining methods. The code below gives a simple example of mining sub-pathways by using the function maxClique in RBGL package that can look for all the cliques in a graph.

```
> graphList <- getDefaultUndirectedGraph()
> graphList <- graphList[sapply(graphList, function(x) length(x) >
+     0)]
> index <- 0
> mySubGraph <- list()
> mySubNames <- character()
> for (i in 1:length(graphList)) {
```

```
+     mc <- maxClique(graphList[[i]])
+     if (length(mc) > 0) {
+         for (j in 1:length(mc[[1]])) {
+             index <- index + 1
+             mySubGraph[index] <- subGraph(mc[[1]][[j]], graphList[[i]])
+             mySubNames[index] <- paste(names(graphList)[i], j,
+                 sep = "_")
+         }
+     }
+ }
> names(mySubGraph) <- mySubNames
```

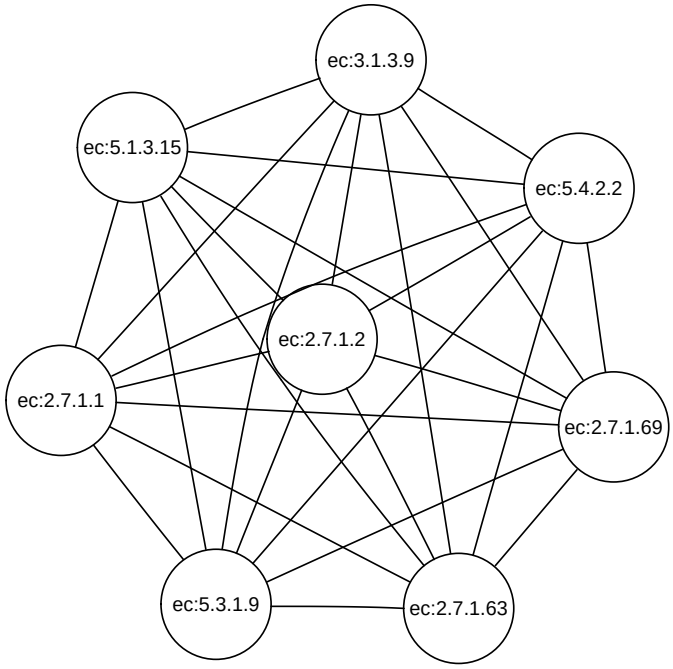
After running the code, You can get a variable `mySubGraph`. a list of subgraph is saved in the variable.

We display a sub-graph in the list.

```
> mySubGraph[1]

$`path:00010_1`
A graphNEL graph with undirected edges
Number of Nodes = 8
Number of Edges = 28

> plot(mySubGraph[[1]], "neato")
```



You can now see that it is a clique in a graph and its name is path:00010_____1. The name means that the graph is first subgraph of the pathway path:00010.

7.3 annotate genes to sub-pathways defined by yourself

After mining user-defined sub-pathways, you can easily annotate genes to these sub-pathways.

You can do:

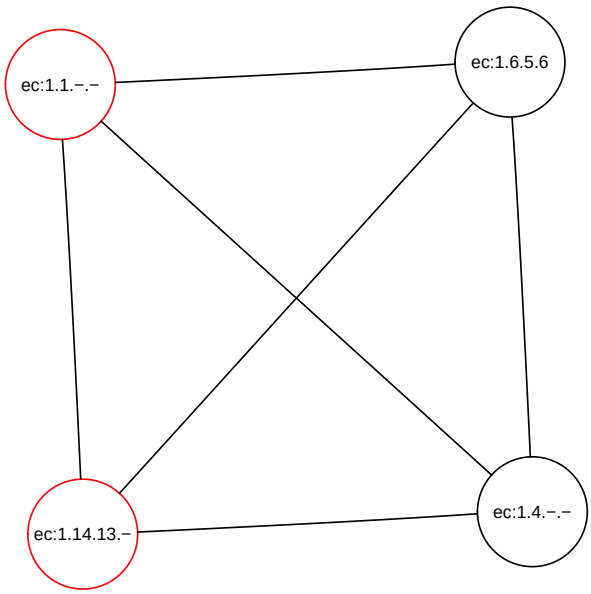
```
> geneList <- getAexample(k = 100)
> ann <- getAnn(geneList, graphList = mySubGraph)
> printAnn(ann)[1:10, 2:5]
```

```
              annGeneRatio annBgRatio                pvalue qvalue
path:00272_6         3/100   30/24143 0.000258079971682368      1
path:00272_4         3/100   32/24143 0.000313401211840314      1
path:00510_3         3/100   48/24143  0.00104165366927156      1
path:00361_2         2/100   14/24143  0.00149628019333981      1
```

```
path:00363_2          2/100    15/24143  0.00172182209288430      1
path:00361_1          2/100    18/24143  0.00248871367445813      1
path:00300_19         1/100     1/24143  0.00414198732551885      1
path:00565_15         1/100     1/24143  0.00414198732551885      1
path:00930_13         1/100     1/24143  0.00414198732551885      1
path:00565_1          2/100    28/24143  0.00598523175842292      1
```

You can also do:

```
> plotAnn("path:00361_2", mySubGraph, ann)
```



Of course, you can use other functions provided by the system.