

How To Use SubpathwayMiner

Chunquan Li

August 31, 2010

Contents

1	Overview	2
2	A simple example of annotating genes to pathways	2
3	Annotate genes to pathways	3
3.1	Annotate gene sets to entire pathways	4
3.2	Annotate gene sets to sub-pathways of metabolic pathways based on enzyme commission (EC)	6
3.3	Annotate gene sets to sub-pathways based on KEGG Orthology (KO) . .	6
3.4	Identify pathways or sub-pathways	6
4	Display and save results	7
4.1	Use data frame to display results	7
4.2	Save annotation results to a tab-delimited file	7
5	Visualization of pathways	8
5.1	Visualize sub-pathways of metabolic pathways based on enzyme commission (EC) using the function plotAnn	8
5.2	Visualize sub-pathways based on KEGG Orthology (KO) using the function plotKOAnn	9
5.3	Visualize pathways or sub-pathways through linking to KEGG web site .	10
6	How to set organism and gene identifier	11
6.1	Set or update the organism and the type of gene identifier	11
6.2	Load and save the environment variable of the system	11
7	Use our flexible model to annotate genes to user-defined sub-pathways	12
7.1	Simplification version of metabolic pathways	12
7.2	Create a subGraph with the algorithms based on the concepts of graph .	14
7.3	annotate genes to sub-pathways defined by yourself	15

8	The extensive application of SubpathwayMiner	16
8.1	Construct human disease-metabolic subpathway networks	16

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"	"10010"	"100130247"	"100132108"		

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:04666	9/100	95/25668	1.27563071217196e-10	2.04179186731454e-08
path:04110	9/100	128/25668	1.86228721332782e-09	1.49040112098838e-07
path:05100	7/100	73/25668	1.45341443325009e-08	7.75451636415696e-07
path:04810	9/100	216/25668	1.75639548238848e-07	5.78286662386364e-06
path:05131	6/100	64/25668	1.87631013059963e-07	6.00649502816575e-06
path:05220	6/100	73/25668	4.14154993300109e-07	1.10483685842495e-05
path:04514	7/100	136/25668	1.06612736061606e-06	2.43779707633078e-05
path:05130	5/100	59/25668	3.43785987488854e-06	6.3587702461993e-05
path:05214	5/100	65/25668	5.56817785746233e-06	8.79026241353678e-05
path:01100	16/100	1120/25668	6.62712704091462e-06	9.7518290409218e-05

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:00624_2	3/100	46/25668	0.00077121620300813	0.158391647217433
path:00903_1	3/100	46/25668	0.00077121620300813	0.158391647217433
path:00903_2	3/100	50/25668	0.000984611164790317	0.158391647217433
path:00565_4	3/100	67/25668	0.00229413721906402	0.158391647217433
path:00565_3	3/100	70/25668	0.00259946638763209	0.158391647217433
path:00310_1	2/100	20/25668	0.00272738541968709	0.158391647217433
path:00310_13	2/100	20/25668	0.00272738541968709	0.158391647217433
path:00310_14	2/100	20/25668	0.00272738541968709	0.158391647217433
path:00310_15	2/100	20/25668	0.00272738541968709	0.158391647217433
path:00310_3	2/100	22/25668	0.00329915729170316	0.158391647217433

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

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

	annGeneRatio	annBgRatio	pvalue	qvalue
path:04810_1	7/100	39/25668	1.52833967703714e-10	8.7029682754571e-08
path:04810_4	7/100	41/25668	2.21979767900393e-10	8.7029682754571e-08
path:04810_3	7/100	42/25668	2.65531485688086e-10	8.7029682754571e-08
path:04810_9	7/100	43/25668	3.16157433566389e-10	8.7029682754571e-08
path:04810_10	7/100	45/25668	4.42411218770644e-10	9.74271775484595e-08
path:05212_7	3/100	4/25668	2.28856436579328e-07	4.19987026656624e-05
path:05220_4	4/100	20/25668	1.0014964644256e-06	0.000157534389651716
path:04110_2	4/100	24/25668	2.17033935334676e-06	0.000281653690908699
path:05200_3	4/100	25/25668	2.57601991093814e-06	0.000315159715897018
path:04110_3	4/100	30/25668	5.49787207537733e-06	0.000605366835310966

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 annotated 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:04666`
$path:04666`$pathwayName
[1] "Fc gamma R-mediated phagocytosis"

$path:04666`$annGeneList
[1] "10000"      "100137049" "10059"      "10092"      "10093"      "10094"
[7] "10095"      "10109"      "10163"

$path:04666`$annBgGeneList
[1] "10000"      "100137049" "10059"      "10092"      "10093"      "10094"
[7] "10095"      "10109"      "10163"      "10451"      "10552"      "1072"
[13] "1073"      "10810"      "123745"     "1398"      "1399"      "1759"
[19] "1785"      "1794"      "207"        "208"        "2209"      "2212"
[25] "2213"      "2214"      "23396"      "23533"      "255189"     "26052"
[31] "27040"      "273"        "283748"     "2934"      "3055"      "3635"
[37] "382"       "3984"      "3985"      "4067"      "4082"      "4651"
[43] "5058"      "50807"     "5290"      "5291"      "5293"      "5294"
[49] "5295"      "5296"      "5321"      "5335"      "5336"      "5337"
[55] "5338"      "55616"     "5578"      "5579"      "5580"      "5581"
[61] "5582"      "5594"      "5595"      "5604"      "56848"     "5788"
[67] "5879"      "5880"      "5894"      "6198"      "6199"      "65108"
[73] "653361"    "653888"    "6850"      "7408"      "7409"      "7410"
[79] "7454"      "81873"     "8394"      "8395"      "8398"      "8503"
[85] "85477"     "8611"      "8612"      "8613"      "8853"      "8877"
[91] "8936"      "8976"      "9103"      "9846"      "998"

$path:04666`$annGeneNumber
[1] 9

$path:04666`$annBgNumber
[1] 95

$path:04666`$geneNumber
```

[1] 100

\$`path:04666`\$bgNumber

[1] 25668

\$`path:04666`\$pvalue

[1] 1.275631e-10

\$`path:04666`\$qvalue

[1] 2.041792e-08

\$`path:04110`

\$`path:04110`\$pathwayName

[1] "Cell cycle"

\$`path:04110`\$annGeneList

[1] "1017" "1019" "1021" "1022" "1026" "1027" "10274" "1028" "1029"

\$`path:04110`\$annBgGeneList

[1]	"1017"	"1019"	"1021"	"1022"	"1026"	"1027"	"10274"	"1028"
[9]	"1029"	"1030"	"1031"	"1032"	"10393"	"10459"	"10735"	"10744"
[17]	"10912"	"10926"	"10971"	"1111"	"11200"	"1387"	"1647"	"1869"
[25]	"1870"	"1871"	"1874"	"1875"	"2033"	"23594"	"23595"	"246184"
[33]	"25"	"25847"	"27127"	"2810"	"2932"	"29882"	"29945"	"3065"
[41]	"3066"	"4085"	"4087"	"4088"	"4089"	"4171"	"4172"	"4173"
[49]	"4174"	"4175"	"4176"	"4193"	"4609"	"4616"	"472"	"494551"
[57]	"4998"	"4999"	"5000"	"5001"	"5111"	"51343"	"51433"	"51434"
[65]	"51529"	"5347"	"545"	"5591"	"5885"	"5925"	"5933"	"5934"
[73]	"595"	"64682"	"6500"	"6502"	"650621"	"651610"	"699"	"701"
[81]	"7027"	"7029"	"7040"	"7042"	"7043"	"7157"	"7272"	"728622"
[89]	"731751"	"7465"	"7529"	"7531"	"7532"	"7533"	"7534"	"7709"
[97]	"8243"	"8317"	"8318"	"8379"	"8454"	"85417"	"8555"	"8556"
[105]	"8697"	"8881"	"890"	"8900"	"891"	"894"	"896"	"898"
[113]	"902"	"9088"	"9126"	"9133"	"9134"	"9184"	"9232"	"9700"
[121]	"983"	"990"	"991"	"993"	"994"	"995"	"996"	"9978"

\$`path:04110`\$annGeneNumber

[1] 9

\$`path:04110`\$annBgNumber

[1] 128

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

```
$`path:04110`$bgNumber  
[1] 25668
```

```
$`path:04110`$pvalue  
[1] 1.862287e-09
```

```
$`path:04110`$qvalue  
[1] 1.490401e-07
```

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.

```
> subGraphListKO<-getKcSubGraph(k=4,graphList=getDefaultKOUndirectedGraph())  
> annKO<-getKOAnn(geneList,graphList=subGraphListKO)
```

3.4 Identify pathways or sub-pathways

Get the statistically significantly enriched pathways according to pvalue.

```
> ann<-getAnn(geneList)  
> cutedAnn<-cutoffAnn(ann,"pvalue","<",0.0001)  
> printAnn(cutedAnn)[2:5]
```

	annGeneRatio	annBgRatio	pvalue	qvalue
path:04666	9/100	95/25668	1.27563071217196e-10	2.04179186731454e-08
path:04110	9/100	128/25668	1.86228721332782e-09	1.49040112098838e-07
path:05100	7/100	73/25668	1.45341443325009e-08	7.75451636415696e-07
path:04810	9/100	216/25668	1.75639548238848e-07	5.78286662386364e-06
path:05131	6/100	64/25668	1.87631013059963e-07	6.00649502816575e-06
path:05220	6/100	73/25668	4.14154993300109e-07	1.10483685842495e-05
path:04514	7/100	136/25668	1.06612736061606e-06	2.43779707633078e-05
path:05130	5/100	59/25668	3.43785987488854e-06	6.3587702461993e-05
path:05214	5/100	65/25668	5.56817785746233e-06	8.79026241353678e-05
path:01100	16/100	1120/25668	6.62712704091462e-06	9.7518290409218e-05
path:04115	5/100	69/25668	7.48340840739381e-06	0.000104388758036484
path:05218	5/100	71/25668	8.61638012072952e-06	0.000112451214617683
path:04270	6/100	126/25668	1.0148257356235e-05	0.000121833139805542
path:03040	6/100	128/25668	1.11055459796061e-05	0.000126969199554623
path:05222	5/100	84/25668	1.96295668871693e-05	0.000207585965399748
path:02010	4/100	44/25668	2.6118099874628e-05	0.000261281161766076
path:05200	8/100	328/25668	4.38150549241234e-05	0.000410734183175453
path:05223	4/100	54/25668	5.90567589201507e-05	0.000525150286323025

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

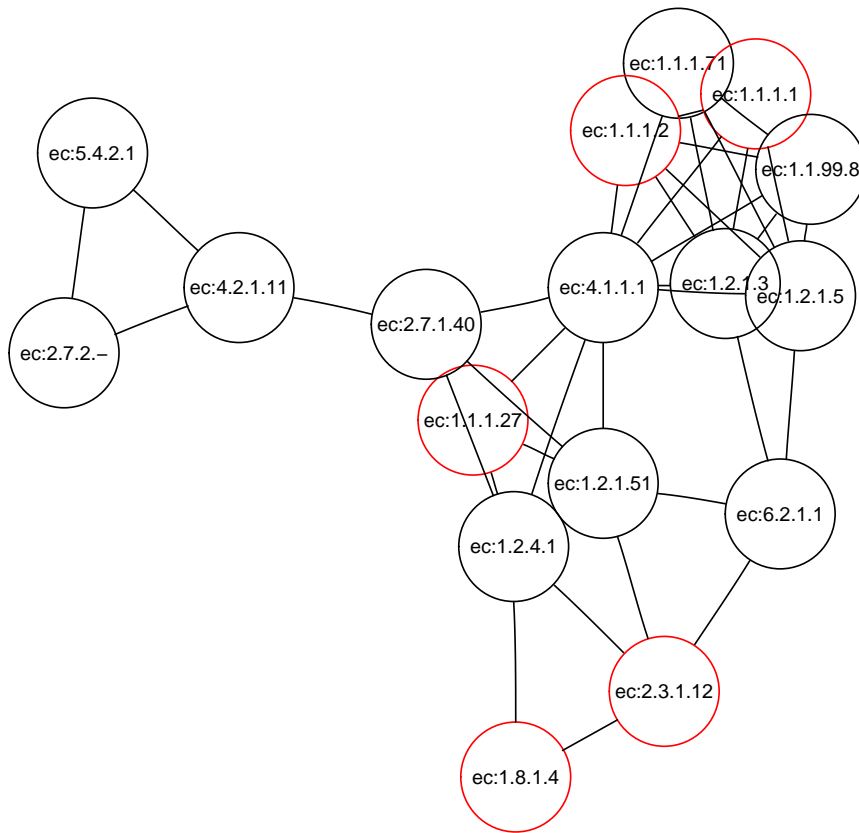
Users can use the function `plotAnn` to visualize the pathways or sub-pathways of metabolic pathways based on ec. The red nodes in the result graph represent the enzymes which include the submitted genes.

Visualize sub-pathways of metabolic pathways based on EC.

```

> geneList<-getAexample(k=1000)
> subGraphList<-getKcSubGraph(k=4)
> ann<-getAnn(geneList,graphList=subGraphList)
> plotAnn("path:00010_1",subGraphList,ann)

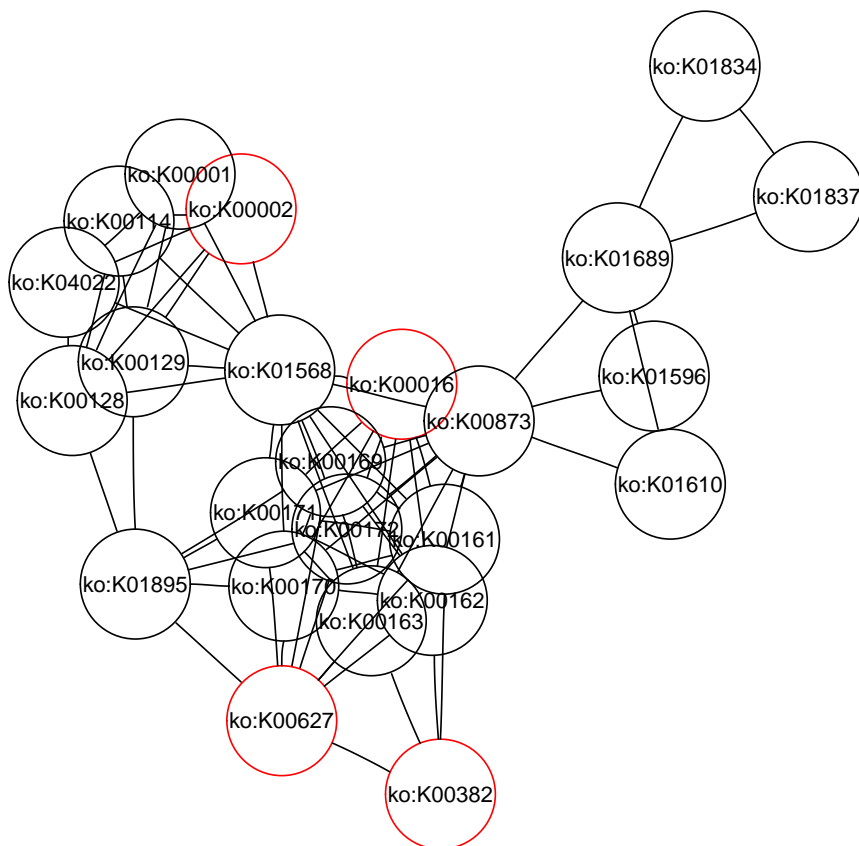
```

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

Visualize sub-pathways based on KO.

```
> geneList<-getAexample(k=1000)
> subGraphListKO<-getKcSubGraph(k=4,graphList=getDefaultKOUndirectedGraph())
> annKO<-getKOAnn(geneList,graphList=subGraphListKO)
> plotKOAnn("path:00010_1",subGraphListKO,annKO)
```



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)

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

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` and `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 annotating 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")
```

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 available 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
```

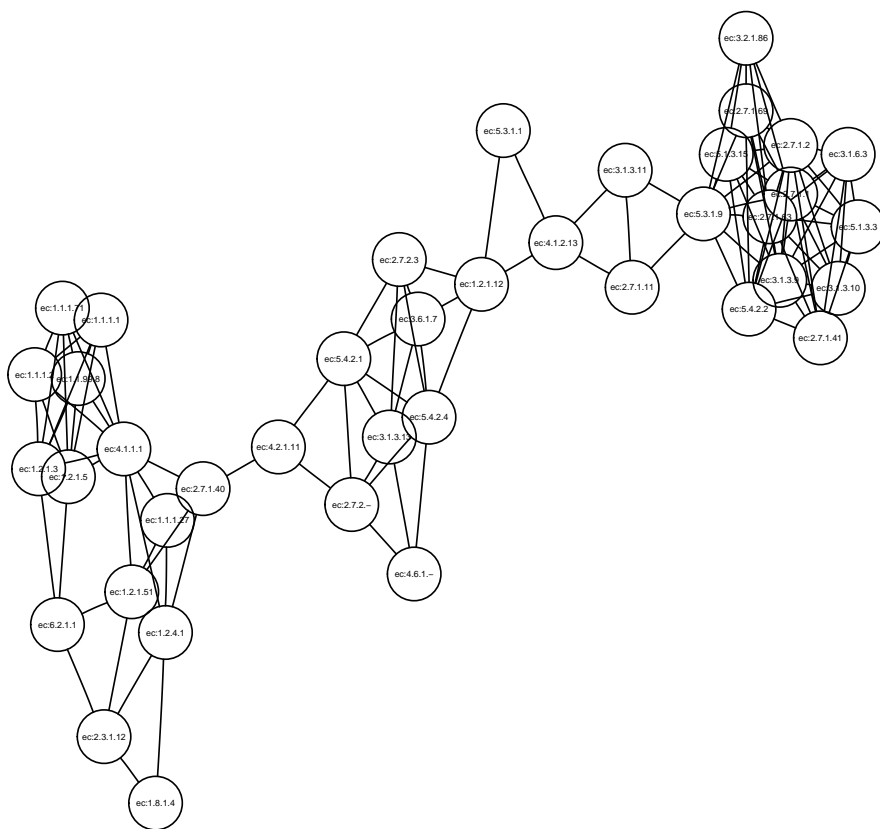
The return value of the function `getDefaultUndirectedGraph` is a list of *graph*. The first graph in the list is the graph representation of the pathway "path:00010". The pathway's name is Glycolysis / Gluconeogenesis. One can use the function `getPNameFromPid` to get it.

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

```
[1] "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
```

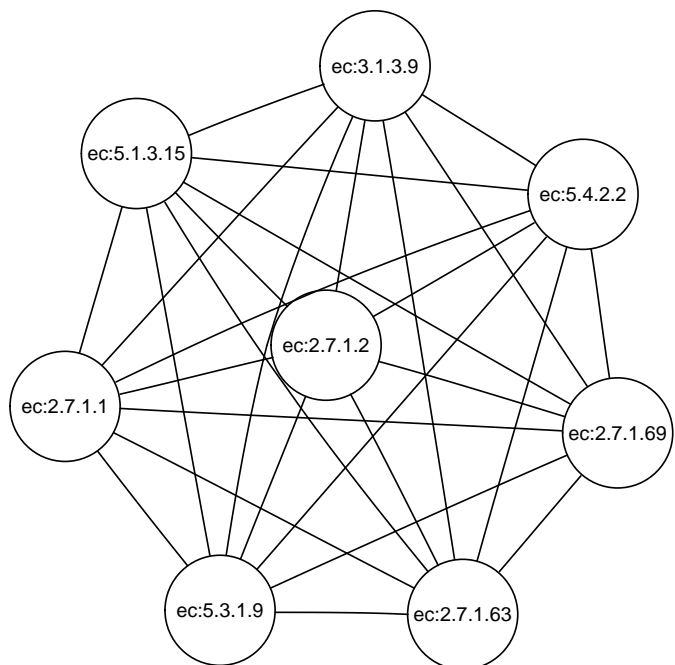
After running the code, You can get a variable `mySubGraph`. a list of subgraph is saved in the variable.

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:

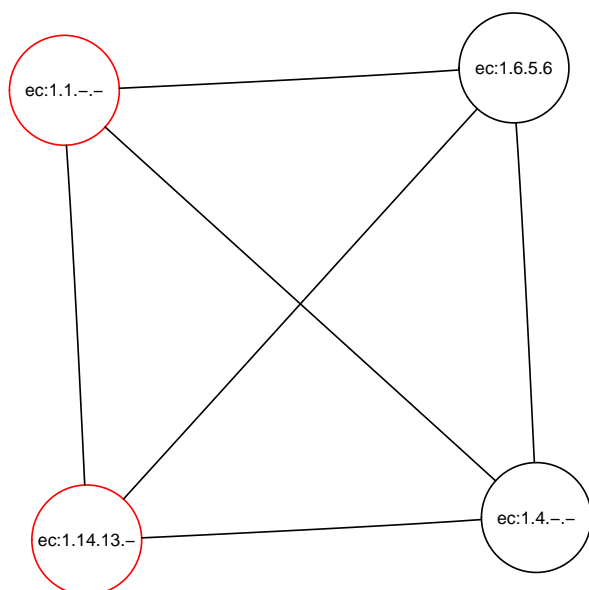
```
> geneList<-getAexample(k=100)
> ann<-getAnn(geneList,graphList=mySubGraph)
> printAnn(ann)[1:10,2:5]
```

	annGeneRatio	annBgRatio	pvalue	qvalue
path:00361_2	2/100	14/25668	0.00132631815811579	0.80878366568216
path:00363_2	2/100	14/25668	0.00132631815811579	0.80878366568216
path:00361_1	2/100	16/25668	0.00174013043157939	0.80878366568216
path:00310_1	2/100	19/25668	0.00246088006132505	0.80878366568216

path:00565_1	2/100	21/25668	0.00300684547427843	0.80878366568216
path:00300_19	1/100	1/25668	0.00389590151160968	0.80878366568216
path:00310_8	1/100	1/25668	0.00389590151160968	0.80878366568216
path:00565_15	1/100	1/25668	0.00389590151160968	0.80878366568216
path:00564_27	2/100	25/25668	0.00425218284044793	0.80878366568216
path:00564_10	2/100	26/25668	0.0045948841590906	0.80878366568216

You can also do:

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



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

8 The extensive application of SubpathwayMiner

8.1 Construct human disease-metabolic subpathway networks

The function is used to construct a disease-metabolic subpathway network(DMSPN) in which nodes represent diseases or metabolic subpathways and they are connected by an

edge if genes of the disease are significantly enriched to the metabolic subpathways. We used the k-cliques subpathway identification method provided by the package to identify statistically significantly enriched disease-causing subpathways. After inputting gene sets (e.g. lung cancer) and distance parameter k, the k-clique method can mine each metabolic subpathway and then identify statistically significantly enriched subpathways. The following describes the step-by-step method for identifying metabolic subpathways. First, each metabolic pathway is converted to an undirected graph with enzymes as nodes. Enzymes in a graph are connected by an edge if their corresponding reactions have a common compound. Secondly, according to parameter k, all sub-pathways (k-cliques) in metabolic pathways can be constructed using k-cliques algorithm on each above graph. For each subpathway (k-clique), distance among all enzymes within it is no greater than the parameter k (a user-defined distance). Gene sets can then be annotated to these subpathways through assigning EC numbers for them and matching them to these subpathways. Finally, the significantly enriched subpathways can be identified using hypergeometric test. To construct disease-metabolic subpathway network (DM-SPN), The disease-causing gene sets need to be used to identify enriched disease-causing subpathways. Then, for each disease, the statistically significantly enriched subpathways were identified by the k-cliques subpathway identification method. Finally, the disease-metabolic subpathway network can be constructed. As an example, the data used by us are obtained from the GAD (Genetic Association Database), which consist of diseases, disease-causing genes (NCBI-geneid), categories (disease classes) that the diseases belong to. The GAD is an NIH supported gene-centered public repository of human association studies examining a wide range of human diseases, including non-mendelian common diseases. Note that, because the function `generateNetwork` is time consuming, the argument `exampleNumber` is used to generate an example of network, which includes the given disease number `exampleNumber`. The default value of `exampleNumber` is -1, meaning that all disease data will be treated. The arguments `pvalue` and `geneNumber` are important for the size of network. For example, if we set up `pvalue=0.001` and `geneNumber=3`, then disease-subpathway associations with $p \geq 0.001$ will be ignored, and those subpathways, within which the disease gene number is less than 3, will be also ignored.

Construct disease-metabolic subpathway networks. Only four diseases are treated as an example of using the function `generateNetwork` to generate networks.

```
> #Use disease-gene associations from GAD.
> path<-paste(system.file(package="SubpathwayMiner"),
+ "/localdata/Dise2G.txt",sep="")
> Dise2G<-read.table(path,header=TRUE,sep = "\t",
+ quote="\"",colClasses="character")
> #generate network
> DMSPN<-generateNetwork(Dise2G,k=3,pvalue=0.005,geneNumber=0,
+ diseCate=TRUE,pathClass=TRUE,exampleNumber=4,verbose=FALSE)
> DMSPN[1:5,c(1,2,3,4,13,7)]
```

	diseaseCates	diseaseName	subpathwayID
1	aging	aging	path:00680_4
2	aging	endocrine	path:00120_6
3	aging	endocrine	path:00120_7
4	aging	longevity	path:00980_1
5	aging	longevity	path:00232_1

	pathwayName
1	Methane metabolism
2	Bile acid biosynthesis
3	Bile acid biosynthesis
4	Metabolism of xenobiotics by cytochrome P450
5	Caffeine metabolism

	pathwayClassNames	pvalue
1	Energy Metabolism	0.00198567143519013
2	Lipid Metabolism	0.00140186894615191
3	Lipid Metabolism	0.00140186894615191
4	Fluorene degradation	1.28510298957885e-07
5	Biosynthesis of Secondary Metabolites	1.1210932856498e-06

Save the network as .txt file.

```
> write.table(DMSPN, "DMSPN.txt", row.names=FALSE, sep="\t")
```

The disease-subpathway associations with $p \geq 0.001$ are ignored, and those subpathways, within which the disease gene number is less than 3, are also ignored.

```
> DMSPN<-generateNetwork(Dise2G,k=4,pvalue=0.001,geneNumber=3,exampleNumber=4)
```

Construct networks with disease class and pathway class.

```
> DMSPN<-generateNetwork(Dise2G,diseCate=TRUE,pathClass=TRUE,exampleNumber=4)
```