

GeneAnswers, Integrated Interpretation of Genes

Gang Feng^{‡,*}, Pan Du^{†,‡}, Warren A. Kibbe^{‡,‡}, Simon Lin^{‡,§}

May 19, 2010

[‡]Northwestern University Biomedical Informatics Center
Northwestern University, Chicago, IL, 60611, USA

Contents

1	Overview of GeneAnswers	1
2	Citation	2
3	Installation of GeneAnswers package	2
4	Object models of major classes	3
5	Data preprocessing	3
5.1	Build a GeneAnswers instance	4
5.2	Visualization	6
5.3	Multigroup Genes Analysis	26
5.4	Homologous Gene Mapping	26
6	Session Info	31
7	Acknowledgments	32
8	References	32

1 Overview of GeneAnswers

Microarray techniques have been widely employed in genomic scale studies for more than one decade. The standard analysis of microarray data is to filter out a group of genes from thousands of probes by certain statistical criteria. These genes are usually called significantly differentially expressed genes. Recently, next generation sequencing (NGS) is gradually adopted to explore gene transcription, methylation, etc. Also a gene list can be obtained by NGS preliminary data analysis. However, this type of information is not enough to understand

*g-feng (at) northwestern.edu

†dupan (at) northwestern.edu

‡wakibbe (at) northwestern.edu

§s-lin2 (at) northwestern.edu

the potential linkage between identified genes and interested functions. The integrated functional and pathway analysis with gene expression data would be very helpful for researchers to interpret the relationship between the identified genes and proposed biological or medical functions and pathways.

The *GeneAnswers* package provides an integrated solution for a group of genes and specified categories (biological or medical functions, such as Gene Ontology, Disease Ontology, KEGG, etc) to reveal the potential relationship between them by means of statistical methods, and make user-friendly network visualization to interpret the results. Besides the package has a function to combine gene expression profile and category analysis together by outputting concept-gene cross tables, keywords query on NCBI Entrez Gene and application of human based Disease ontology analysis of given genes from other species can help people to understand or discover potential connection between genes and functions.

2 Citation

For the people using *GeneAnswers* package, please cite the following papers in your publications.

* For DOLite:

Du, P., Feng, G., Flatow, J., Song, J., Holko, M., Kibbe, W.A. and Lin, S.M., (2009) 'From disease ontology to disease-ontology lite: statistical methods to adapt a general-purpose ontology for the test of gene-ontology associations', *Bioinformatics* 25(12):i63-8

* For GeneAnswers:

Feng, G., Du, P., Krett, N.L., Tessel, M., Rosen, S., Kibbe, W.A., and Lin, S.M., (submitted) 'Bioconductor Methods to Visualize Gene-list Annotations',
Thanks for your help!

3 Installation of GeneAnswers package

In order to install the *GeneAnswers* package, the user needs to first install R, some related Bioconductor packages. You can easily install them by the following codes.

```
source("http://bioconductor.org/biocLite.R")
biocLite("GeneAnswers")
```

For the users want to install the latest developing version of *GeneAnswers*, which can be downloaded from the developing section of Bioconductor website. Some additional packages might be required to be installed because of the update the Bioconductor. These packages can also be found from the developing section of Bioconductor website. You can also directly install the source packages from the Bioconductor website by specify the developing version number, which can be found at the Bioconductor website. Suppose the developing version is 2.5, to install the latest *GeneAnswers* package in the Bioconductor developing version, you can use the following command:

```
install.packages("GeneAnswers", repos="http://www.bioconductor.org/packages/2.5/bioc", type=
```

4 Object models of major classes

The *GeneAnswers* package has one major class: **GeneAnswers**. It includes the following slots:

1. *geneInput*: a data frame containing gene Entrez IDs with or without any related values. The values could be foldChange, p value, or other values. These data can be used for concept-gene network. Genes with positive values will be represented as red nodes, while negative value genes are green nodes.
2. *testType*: statistical test method. Current version supports hypergeometric test to test relationship between genes and specified categories.
3. *pvalueT*: the cutoff of statistical test. Any categories will not be reported if the p value is more than the cutoff.
4. *genesInCategory*: a list containing genes belonging to categories. The names of the list are categories.
5. *geneExpProfile*: a data frame to store gene expression data. If not available, it could be NULL.
6. *annLib*: annotation database used for statistical test.
7. *categoryType*: functional or medical category used for statistical test.
8. *enrichmentInfo*: a data frame containing filtered categories with statistical results by specified pvalueT.

The figure, 'Flow chart of GeneAnswers', shows how *GeneAnswers* package works. A group of genes are essential. We use unique Entrez gene IDs to represent genes. Any relative feature values of these genes can also be optional input information, like fold changes, p values, etc. If the gene expression profile of these genes are available, it can be considered as input, too. Since we want to find the potential connections between genes and categories, category type is also need to be specified. *GeneAnswers* currently supports Gene Ontology (GO), Pathway (KEGG) and developing Disease Ontology (DOLite) in our team. Furthermore, *GeneAnswers* supports Entrez eUtils so that users can make customized annotation library based on interested keywords. If users have own annotation library, *GeneAnswers* can use it to build relationship between it and given genes.

Besides usual barplot and pie chart of top categories, *GeneAnswers* also provides four types of visualization. One is concepts-genes network, which show the concepts and genes on a network layout. The second one is concepts-genes cross table that integrated gene expression profile and corresponding categories together. The third one is a concepts-network shows connections between categories only. The last one is a table, which contains all of information of categories and genes. Combining all of these presentations can be helpful to find and explain the possible linkages between genes and categories.

5 Data preprocessing

First of all, load the *GeneAnswers* package.

```
> library(GeneAnswers)
```

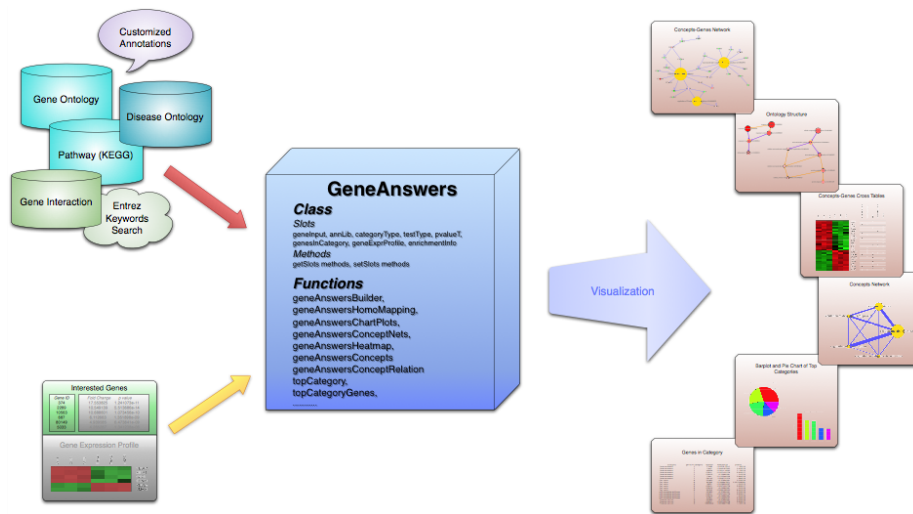


Figure 1: Flow chart of GeneAnswers

5.1 Build a GeneAnswers instance

The key point of *GeneAnswers* package is to build a GeneAnswers instance. The essential input for *GeneAnswers* is an Entrez gene IDs vector (a character vector). However, if users have any interested values associated with genes, these values can also be as optional inputs. In this case, the input, geneInput, could be a matrix or a dataframe. The first column is always for Entrez gene IDs. Other columns are used to store those interested values. Rownames for the matrix or dataframe are not necessary, but colnames are recommended for further usage. We use two internal datasets, one is from human and another is from mouse, as examples to show how to implement *GeneAnswers* package. The human and mouse datasets coming with the *GeneAnswers* package are from human and mouse Illumina beadarray experiments. Each dataset contains two dataframes. For example, humanGeneInput is a dataframe containing Entrez gene IDs with fold changes and p values, while the data frame, humanExpr, includes two types, control and treatment, of gene expression profile of the genes in humanGeneInput.

```
> data('humanGeneInput')
> data('humanExpr')
> ## build a GeneAnswers instance with statistical test based on biological process of GO
> x <- geneAnswersBuilder(humanGeneInput, 'org.Hs.eg.db', categoryType='GO.BP', testType='GO')

[1] "geneInput has built in ..."
[1] "annLib and categoryType have built in ..."
[1] "genesInCategory has built in ..."
[1] "Enrichment test is only performed based on annotated genes"
[1] "testType, pvalueT and enrichmentInfo have built in ..."
[1] "geneExpressionProfile has been built in ..."
[1] "GeneAnswers instance has been successfully generated!"

> class(x)
```

```

[1] "GeneAnswers"
attr(,"package")
[1] "GeneAnswers"

> ## build a GeneAnswers instance with statistical test based on KEGG and saved example data
> y <- geneAnswersBuilder(humanGeneInput, 'org.Hs.eg.db', categoryType='KEGG', testType='hyperG')

[1] "GeneAnswers instance has been successfully generated!"

> ## build a GeneAnswers instance with statistical test based on DOLite and saved example data
> z <- geneAnswersBuilder(humanGeneInput, 'org.Hs.eg.db', categoryType='DOLite', testType='hyperG')

[1] "GeneAnswers instance has been successfully generated!"

> w <- geneAnswersBuilder(humanGeneInput, 'org.Hs.eg.db', categoryType='GO.BP', testType='hyperG')

[1] "GeneAnswers instance has been successfully generated!"

```

We have four GeneAnswers objects, x, y, z and w, containing the statistical test of biological process of GO, KEGG, DOLite and GO (The first two level nodes are removed), respectively. For Gene Ontology, sometimes, users think some nodes are too general and not very relative to their interests. So we provide parameter *level* to determine how many top levels of GO nodes are removed. The instances have included the relationship between given genes and specified categories.

GeneAnswers package also provides a function *searchEntrez* to retrieve Entrez genes for given keywords by Entrez XML query. National Center for Biotechnology Information (NCBI) provides many powerful online query systems. One of them is Entrez Programming Utilities (eUtils). Users can query NCBI databases by simple keywords logical operations based on XML protocol. This is very helpful to find potential or interested biological functions or pathways. Hence, the retrieved information can be considered as a customized annotation library to test whether the given genes are relative to interested keywords. Here is a case to build a customized GeneAnswers instance.

```

> keywordsList <- list(Apoptosis=c('apoptosis'), CellAdhesion=c('cell adhesion'))
> entrezIDList <- searchEntrez(keywordsList)

[1] "search link: http://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=gene&term=apoptosis"
[1] "search link: http://eutils.ncbi.nlm.nih.gov/entrez/eutils/esearch.fcgi?db=gene&term=cell%20adhesion"

> q <- geneAnswersBuilder(humanGeneInput, entrezIDList, testType='hyperG', totalGeneNumber=10000)

[1] "GeneAnswers instance has been successfully generated!"

> class(q)

[1] "GeneAnswers"
attr(,"package")
[1] "GeneAnswers"

> getAnnLib(q)

```

```
NULL
```

```
> getCategoryType(q)
```

```
[1] "User defined"
```

Customized *GeneAnswers* instances have NULL at *annLib* slot and "User defiend" in *categoryType* slot.

5.2 Visulization

Besides barplot and pie chart, *GeneAnswers* package can generate a network (concept-gene network) show how genes are connected to specified categories as well as general barplot and piechart. Function *GeneAnswersConceptNet* can generate a common R canvas or tcl/tk interactive canvas to draw the network by calling *igraph*. Genes are presented as red nodes, if specified values are positive, and the gene nodes are green with negative values. The category nodes are yellow nodes, the sizes are relative to user-specified values. Currently, if function *GeneAnswersBuilder* successfully returns a *GeneAnswers* instance, the genes are represented as entrez IDs and categories are also category IDs. User can map them to gene symbols and categories terms by function *GeneAnswersReadable*. Function *GeneAnswersReadable* reads slot *annLib* to map Entrez IDs to gene symbols, so make sure slot *annLib* is correct before mapping.

```
> ## mapping gene IDs and category IDs to gene symbols and category terms
> xx <- geneAnswersReadable(x)
```

```
[1] "Mapping geneInput ..."
```

```
[1] "Mapping genesInCategory ..."
```

```
[1] "Mapping enrichmentInfo rownames ..."
```

```
[1] "Mapping geneExprProfile rownames ..."
```

```
> yy <- geneAnswersReadable(y, verbose=FALSE)
```

```
> zz <- geneAnswersReadable(z, verbose=FALSE)
```

```
> ww <- geneAnswersReadable(w, verbose=FALSE)
```

```
> q <- setAnnLib(q, 'org.Hs.eg.db')
```

```
> qq <- geneAnswersReadable(q, catTerm=FALSE)
```

```
[1] "Mapping geneInput ..."
```

```
[1] "Mapping genesInCategory ..."
```

```
[1] "Mapping geneExprProfile rownames ..."
```

Since function *geneAnswersReadable* implements mapping based on annotation database in slot *annLib*, we assign 'org.Hs.eg.db' to customized *GeneAnswers* instance *annLib* slot at first for make it readable.

```
> ## plot barplot and / or piechart
```

```
> geneAnswersChartPlots(xx, chartType='all')
```

The nodes could be represented by different colors for different specified values, like fold change. In this case, overexpressed genes are represented as red nodes while green dots stand for underexpressed genes. If the node is more red, which means its fold change is larger. It's same for green nodes, but different direction.

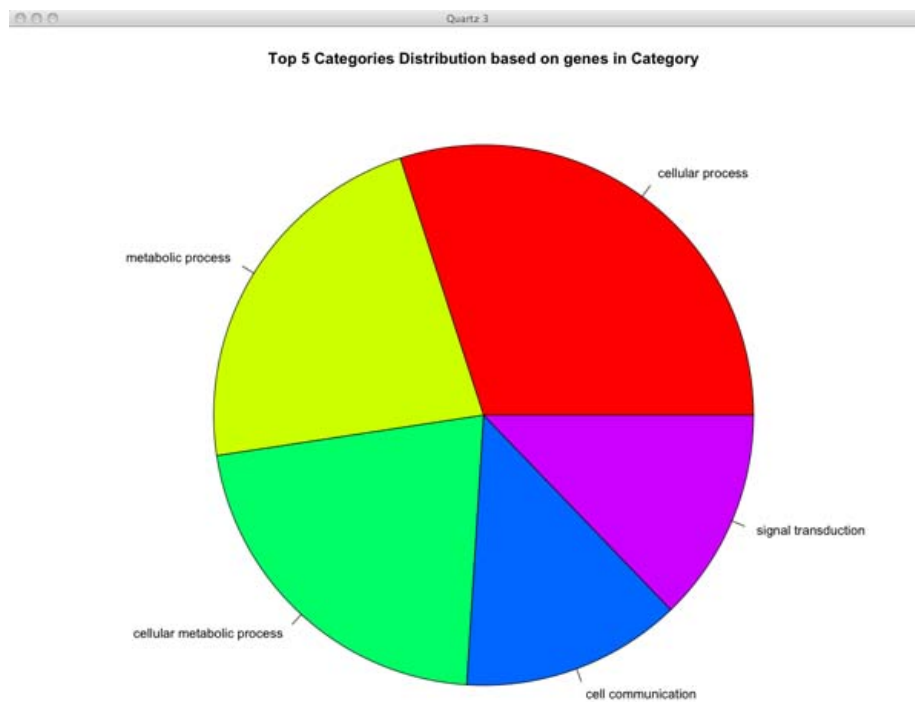


Figure 2: Screen shot of pie chart of top categories

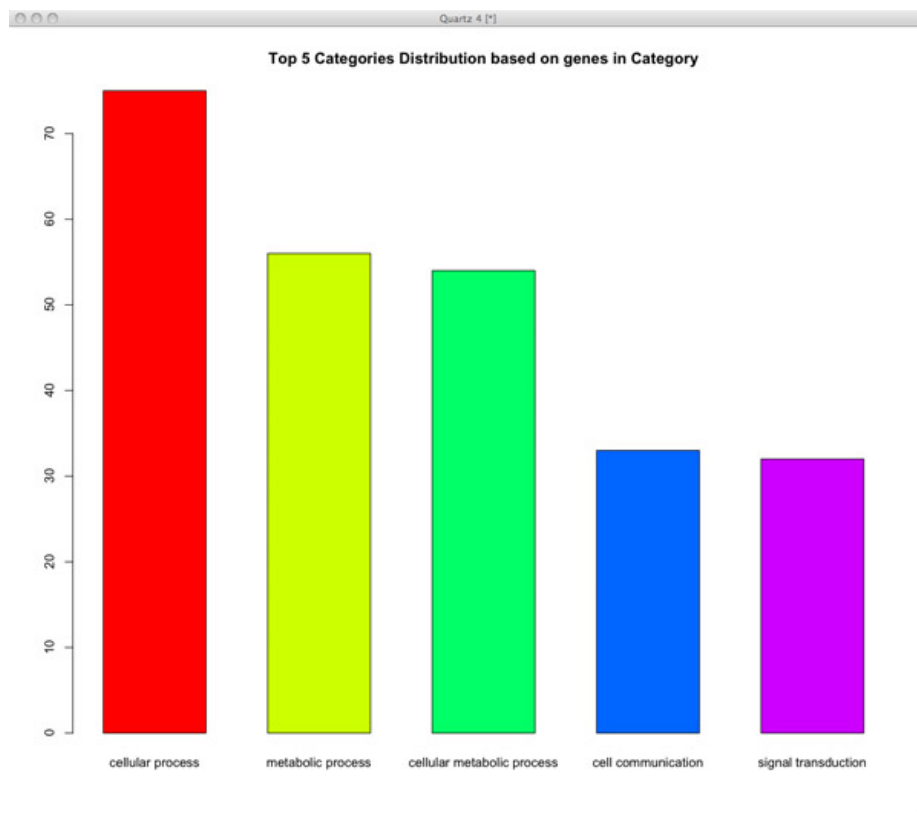


Figure 3: Screen shot of barplot of top categories

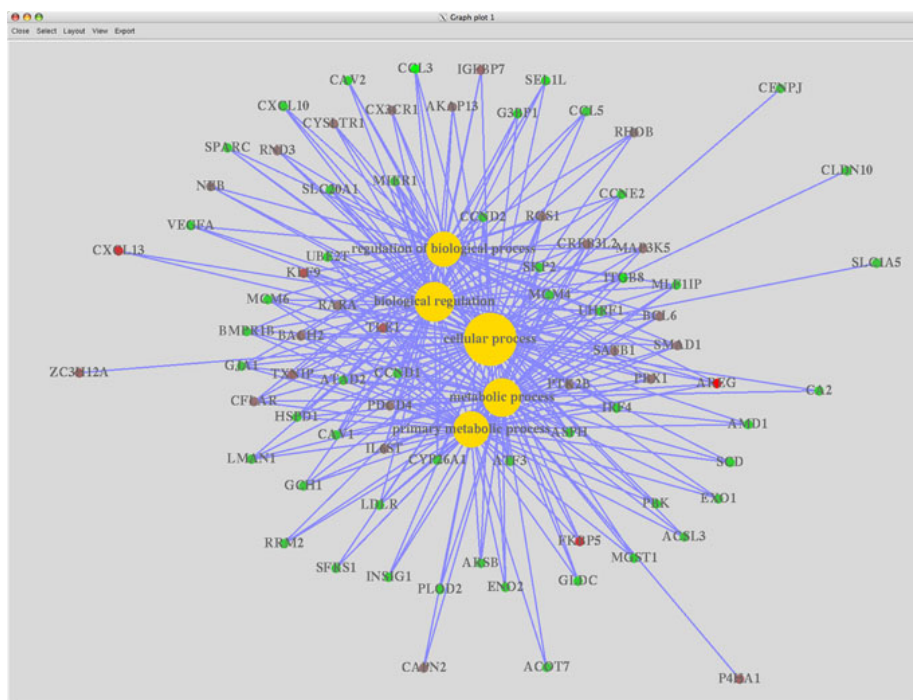


Figure 4: Screen shot of concept-gene network

```
> ## plot interactive concept-gene network
```

```
> geneAnswersConceptNet(xx, colorValueColumn='foldChange', centroidSize='pvalue', output='')
```

The top 5 categories might not be very specific. Users might get a tree view to see relative category terms by calling function `geneAnswersConceptRelationIf` if the category has an ontology structure. The size of nodes is proportional to number of genes in these GO categories. The color of nodes stand for how relative the given genes are to the GO categories. More red, more relative. The given GO categories are yellow framed dots with dark purple edges connetions.

```
> ## plot interactive go structure network
```

```
> geneAnswersConceptRelation(x, direction='both', netMode='connection', catTerm=TRUE, catI
```

Also the new version GeneAnswers integrates gene or protein interaction database from NCBI. The following case is a typical one to show interaction information could be included in basic concepts-genes network.

```
> ## plot interactive concept-gene network
```

```
> geneAnswersConceptNet(x, color='foldChange', geneLayer=5, output='interactive', showCats
```

This function can also be used to show how the given genes interact with each other. For example, for the given genes in the GeneAnswers instance, `x`, users can use the following command to show gene '444','3638', '5087' and '55835' interact with each other and other genes in the given `geneInput` of `x`.

```
> ## plot the given gene interaction
```

```
> buildNet(c('444', '3638', '5087', '55835'), idType='GeneInteraction', layers=2, filterGrap
```

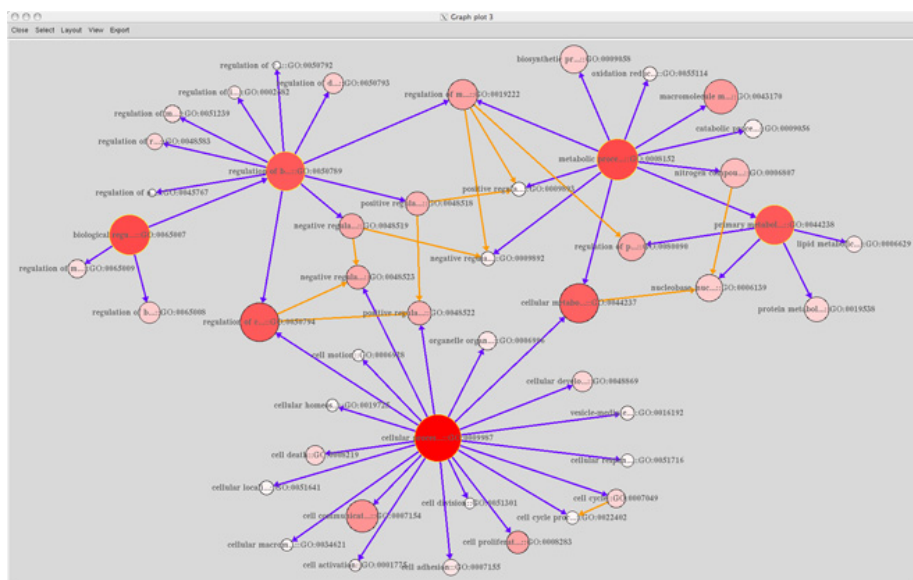


Figure 5: Screen shot of go structure network

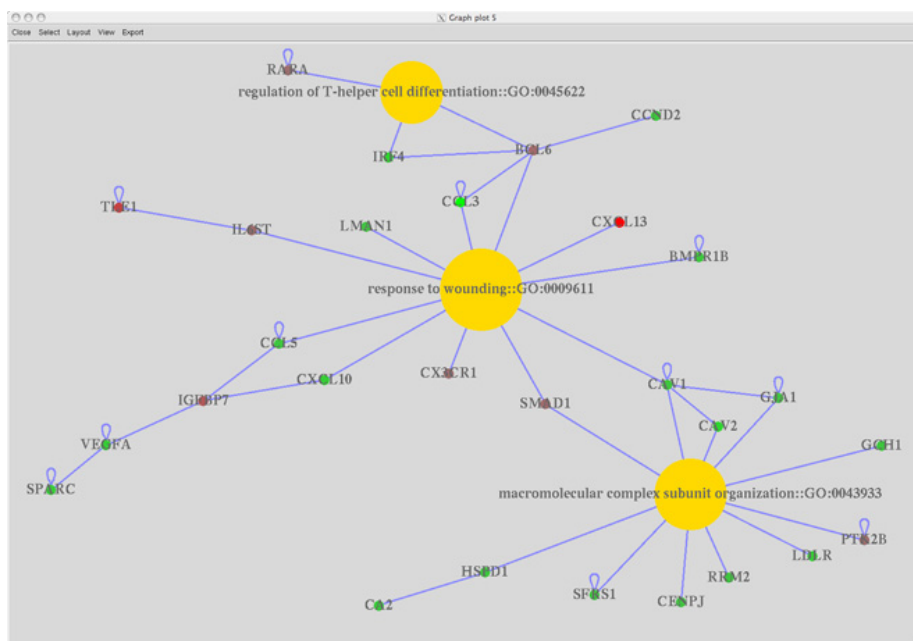


Figure 6: Screen shot of concept-gene network

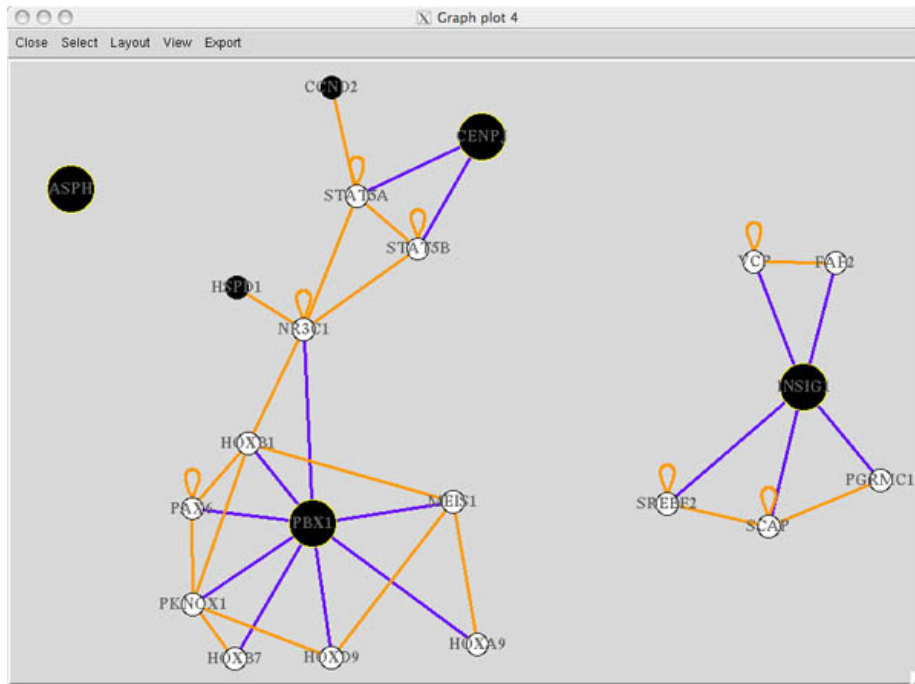


Figure 7: Screen shot of the given gene interaction network

In this case, large black dots with yellow frame stand for the 4 given genes. They also connect to other genes by dark-blue-purple edges. Small black dots represent the other genes from `getGeneInput(x)`. Small white dots are genes that are not in the genes from `getGeneInput(x)`, but interact with these genes.

If there are some certain values associate with genes in `geneInput` of `x`, like the example data, you can represent any one column by colors. For example, if users want to show how genes interaction with gene expression information, the following command can show this:

```
> ## plot the given gene interaction
> buildNet(c('444', '3638', '5087', '55835'), idType='GeneInteraction', layers=2, filterGrap
```

The following one is based on expression p-values, we often use $-\log_2$ to transform p-values.

```
> ## plot the given gene interaction
> buildNet(c('444', '3638', '5087', '55835'), idType='GeneInteraction', layers=2, filterGrap
```

Users can also define customized color scheme and transfer it into 'buildNet' function by parameter 'colorMap'. Details are available in 'buildNet' manual page. If users want to have a overview for the interaction of all of the given genes from `getGeneInput(x)`, the following command could be used:

```
> ## plot the given gene interaction
> buildNet(getGeneInput(x)[,1], idType='GeneInteraction', layers=2, filterGraphIDs=getGene
```

If there are a lot of genes, the network could be very complicated. We strongly recommend to use 'interactive' mode to show the network since it's

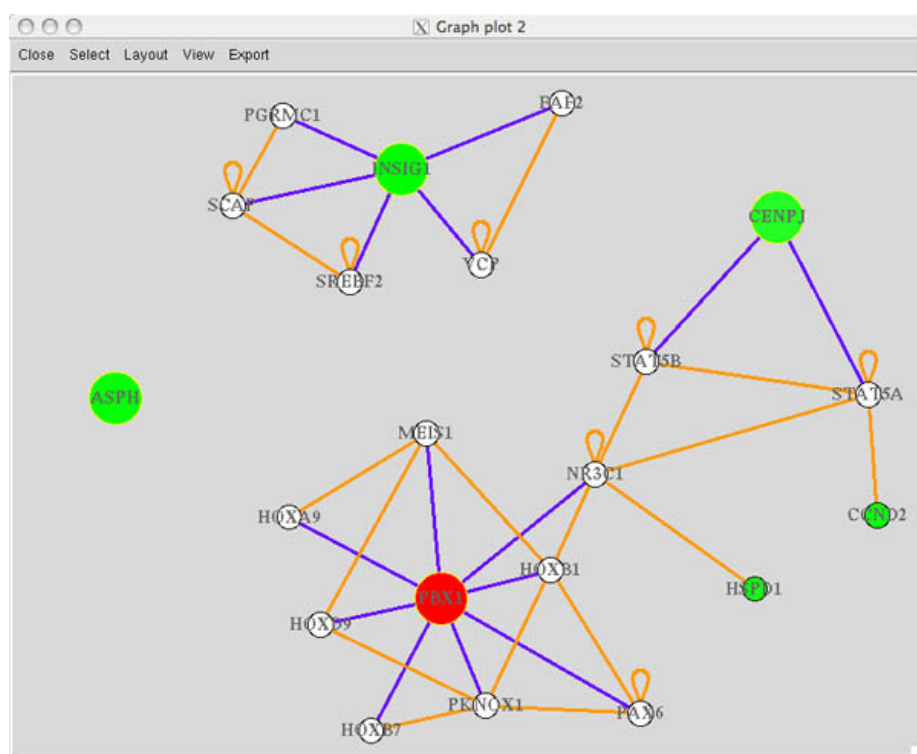


Figure 8: Screen shot of the given gene interaction network with expression information

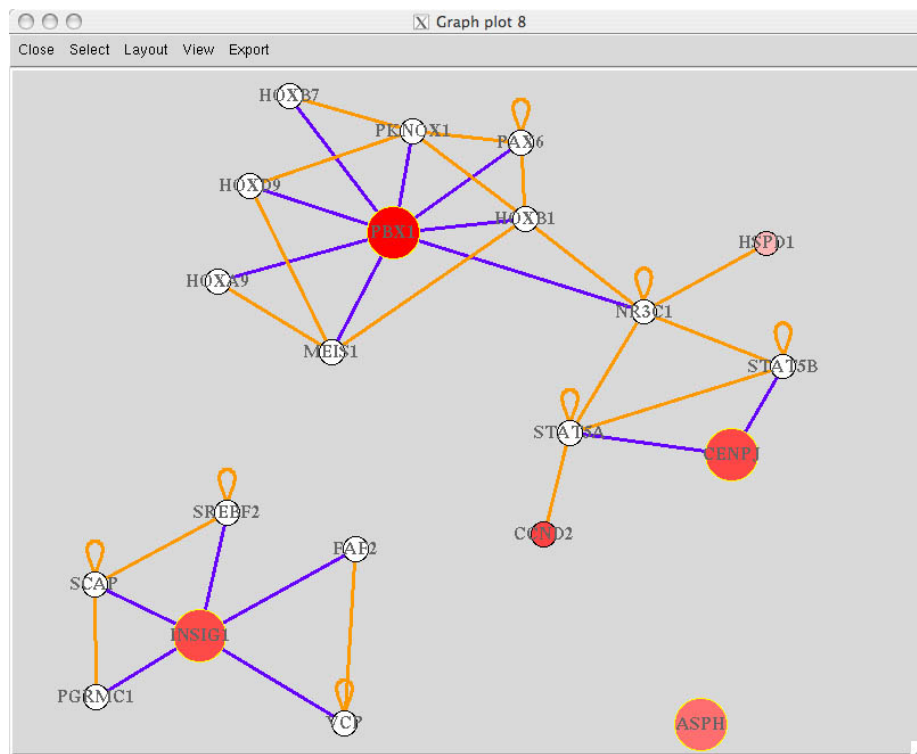


Figure 9: Screen shot of the given gene interaction network with p-value information

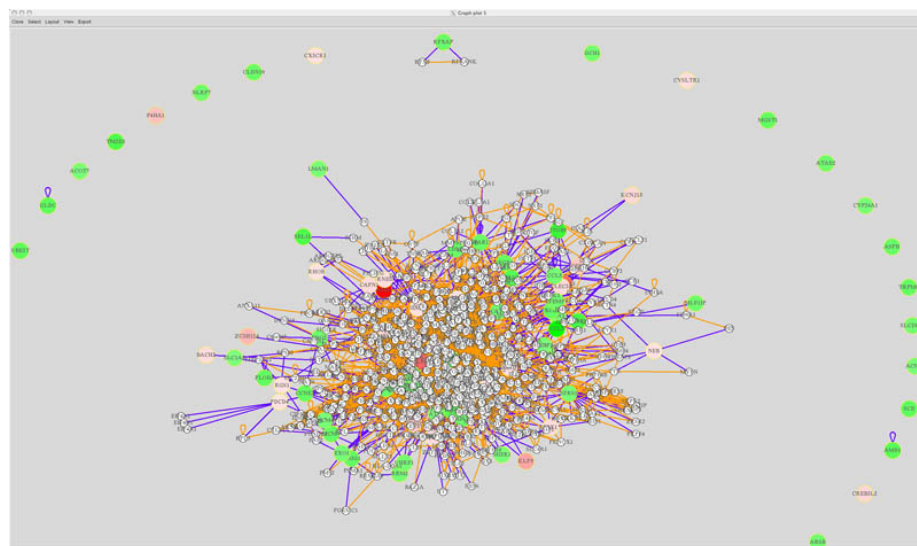


Figure 10: Screen shot of all of the given gene interaction network with expression information

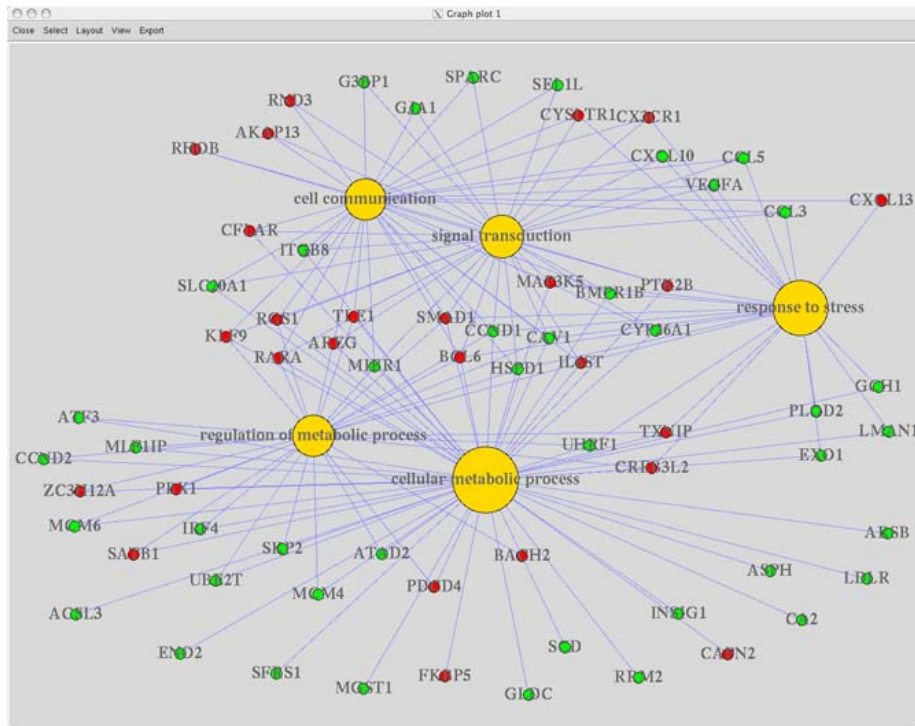


Figure 11: Screen shot of concept-gene network for top 2 GO level nodes removal

easy to manually change layout. The default setting is 'interactive', but users can change it to 'fixed' for a small or simple network.

The following example is to show top 5 GO-gene network for the first 2 level GO nodes removal.

```
> ## plot Go-concept network for 2 level nodes removal
> geneAnswersConceptNet(wv, colorValueColumn='foldChange', centroidSize='pvalue', output='
Also, users can sort enrichment test information and plot it.

> ## sort enrichmentInfo dataframe by fdr adjusted p value
> xxx <- geneAnswersSort(xx, sortBy='correctedPvalue')
> yyy <- geneAnswersSort(yy, sortBy='pvalue')
> zzz <- geneAnswersSort(zz, sortBy='geneNum')

> geneAnswersConceptNet(yyy, colorValueColumn='foldChange', centroidSize='geneNum', output=
> geneAnswersConceptNet(zzz, colorValueColumn='foldChange', centroidSize='pvalue', output=
```

If users provide a gene expression profile, *GeneAnswers* package can generate a table or heatmap labeling relationship between genes and categories with a heatmap of these genes expression. We call this type of representation as concept-gene cross tabulation.

```
> ## generate GO-gene cross tabulation
> geneAnswersHeatmap(x, catTerm=TRUE, geneSymbol=TRUE)
```

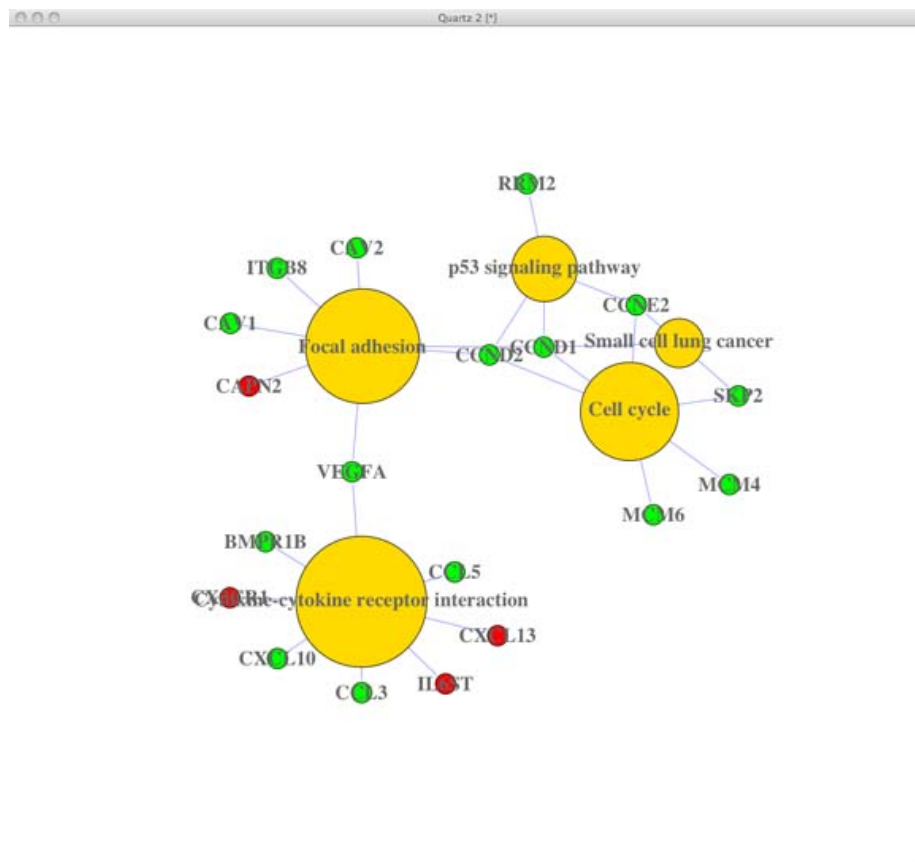


Figure 12: Screen shot of KEGG-gene network

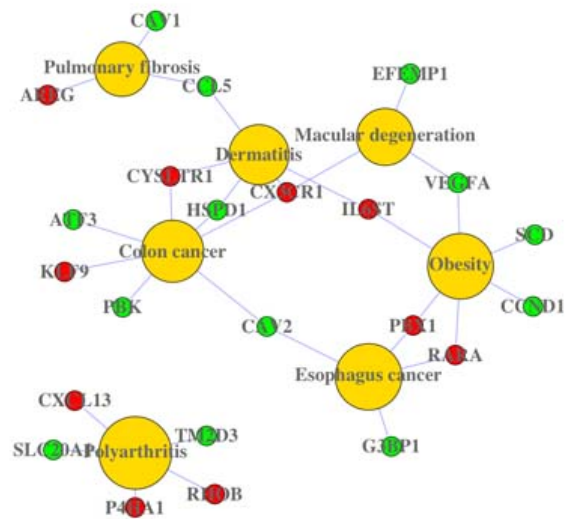


Figure 13: Screen shot of DOLite-gene network


```
> ## generate GO-gene cross tabulation
> geneAnswersHeatmap(x, catTerm=TRUE, geneSymbol=TRUE)
```

```
initial value 2.095268
iter 5 value 0.284632
iter 10 value 0.180610
iter 15 value 0.104728
iter 20 value 0.073371
iter 25 value 0.061866
iter 30 value 0.044067
iter 35 value 0.025023
iter 40 value 0.014669
final value 0.004752
converged
initial value 4.654638
iter 5 value 2.542047
final value 0.000000
converged
```

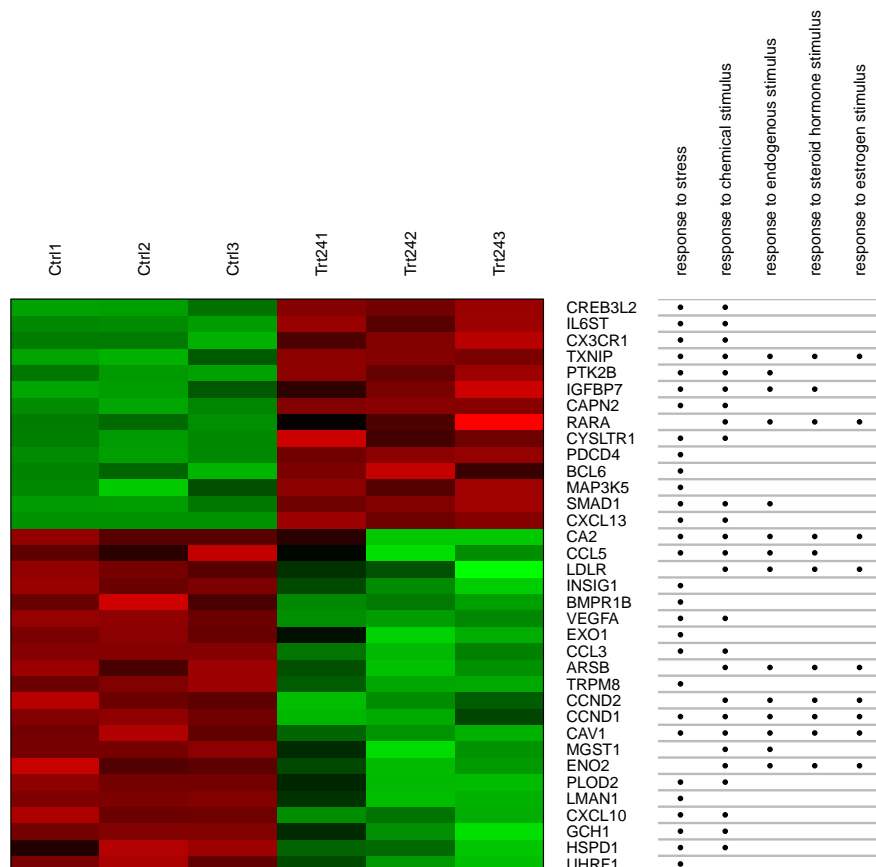


Figure 14: GO-gene cross tabulation

```
> geneAnswersHeatmap(yyy)
```

For cross table, there are two types of representations. One is a table, which is better for few genes, and another one is a two-color heatmap that is adopted for a lot of genes. In the latter, the default setting is that white bar stands for that a gene in that category.

Besides top categories, users can also show interested categories.

```
> GOBPIDs <- c("GO:0007049", "GO:0042592", "GO:0006259", "GO:0016265", "GO:0007243")
> GOBPTerms <- c("cell cycle", "death", "protein kinase cascade", "homeostatic process", "

> ## generate concept-gene cross tabulation
> geneAnswersConceptNet(x, colorValueColumn='foldChange', centroidSize='pvalue', output='f

> geneAnswersHeatmap(x, showCats=GOBPIDs, catTerm=TRUE, geneSymbol=TRUE)
```

Function *geneAnswersConcepts* shows the linkages of specified categories. The width of edge stands for how overlapping between two categories.

```
> ## generate concept-gene cross tabulation
> geneAnswersConcepts(xxx, centroidSize='geneNum', output='fixed', showCats=GOBPTerms)
```

Users can also print top categories and genes on screen and save them in files by specification as well as these two types of visualization. The default file names are "topCategory.txt" and "topCategoryGenes.txt" for top categories with or without corresponding genes, respectively.

```
> ## print top GO categories sorted by hypergeometric test p value
> topGOGenes(x, orderby='pvalue')
```

```
[1] "***** response to estrogen stimulus    p value : 5.69989260507789e-09 *****"
      Symbol foldChange      pValue
411   ARSB  -2.122207 3.199329e-06
760    CA2  -2.256972 8.992257e-04
857   CAV1  -2.122246 3.842276e-06
595  CCND1  -2.257123 5.932920e-07
894  CCND2  -2.239806 1.208734e-06
[1] "***** response to chemical stimulus    p value : 8.4205667864018e-09 *****"
      Symbol foldChange      pValue
411   ARSB  -2.122207 3.199329e-06
760    CA2  -2.256972 8.992257e-04
824   CAPN2  2.568772 1.890638e-08
857   CAV1  -2.122246 3.842276e-06
6348  CCL3  -4.031781 1.147014e-07
[1] "***** response to steroid hormone stimulus    p value : 1.0147596911312e-08 *****"
      Symbol foldChange      pValue
411   ARSB  -2.122207 3.199329e-06
760    CA2  -2.256972 8.992257e-04
857   CAV1  -2.122246 3.842276e-06
6352  CCL5  -2.154827 1.092361e-03
595  CCND1  -2.257123 5.932920e-07
```

```
> geneAnswersHeatmap(yyy)
```

```
initial value 1.716305
iter 5 value 0.407656
iter 10 value 0.112360
iter 15 value 0.072109
iter 20 value 0.052017
iter 25 value 0.029419
final value 0.006019
converged
initial value 4.466884
iter 5 value 1.279433
iter 10 value 0.084821
final value 0.008157
converged
```

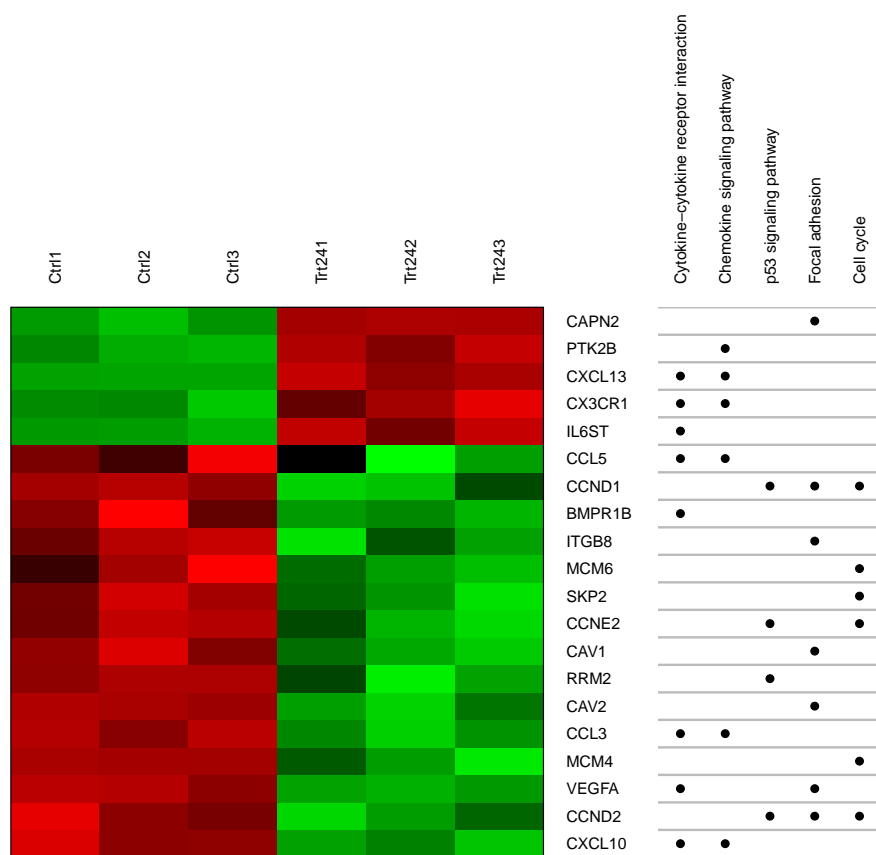


Figure 15: KEGG-gene cross tabulation

```

initial value 1.851350
iter 5 value 0.390287
iter 10 value 0.146730
iter 15 value 0.120260
iter 20 value 0.092560
iter 25 value 0.046348
iter 30 value 0.035369
iter 35 value 0.028131
iter 40 value 0.021161
iter 45 value 0.011081
iter 45 value 0.008695
iter 45 value 0.005071
final value 0.005071
converged
initial value 19.020185
iter 5 value 2.491750
iter 10 value 0.333367
iter 15 value 0.027846
final value 0.002865
converged

```

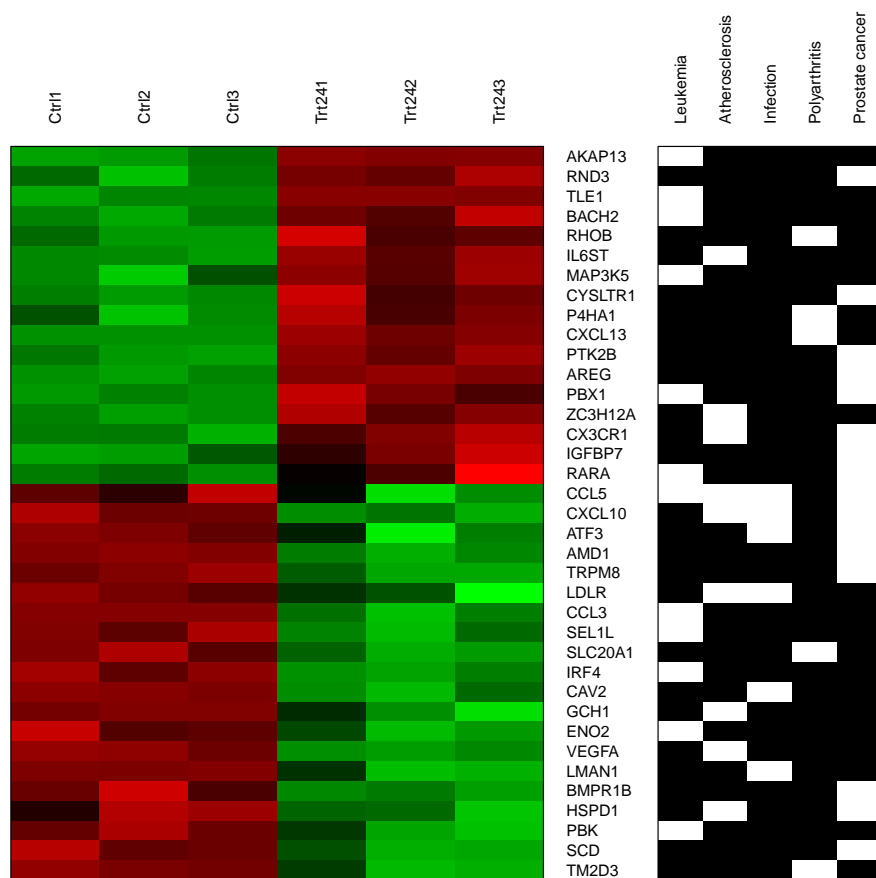


Figure 16: DOLite-gene cross tabulation

```

[1] "Some specified categories might not be statistical significant! Only show significant
initial value 1.955644
iter 5 value 0.250440
iter 10 value 0.129463
iter 15 value 0.075394
iter 20 value 0.040824
iter 25 value 0.016440
iter 30 value 0.013925
final value 0.006865
converged
initial value 0.000000
final value 0.000000
converged

```

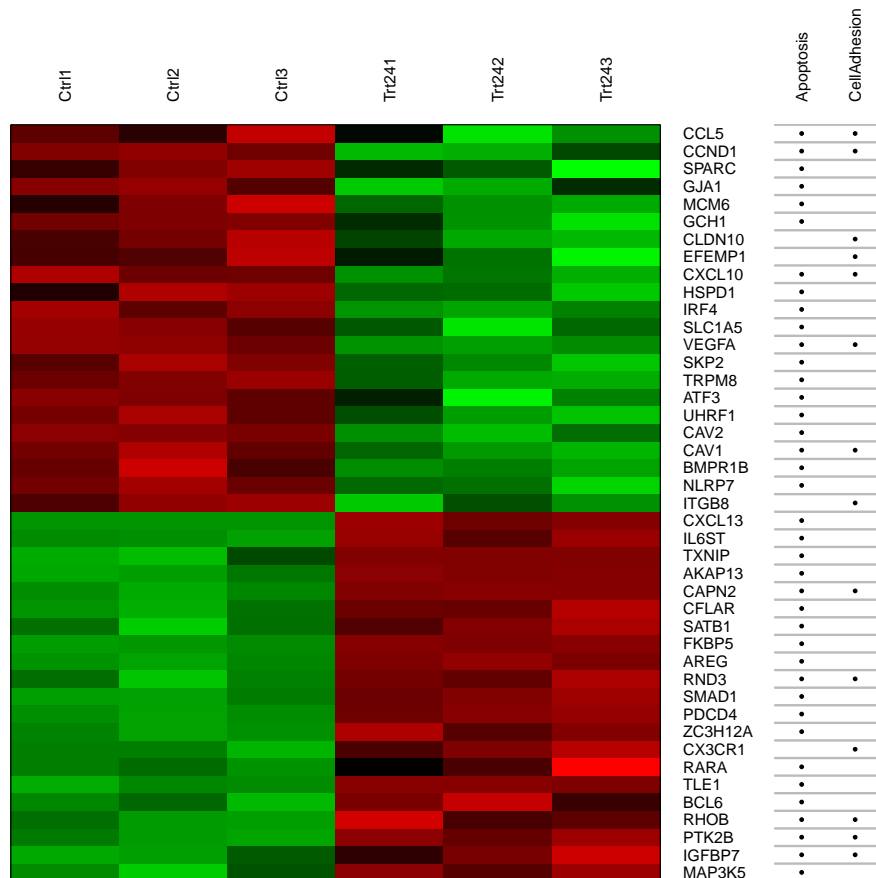


Figure 17: Customized GO-gene cross tabulation

initial value 2.100918
iter 5 value 0.385357
iter 10 value 0.185044
iter 15 value 0.145916
iter 20 value 0.105888
iter 25 value 0.081109
iter 30 value 0.055129
iter 35 value 0.045909
iter 40 value 0.037991
iter 45 value 0.016759
final value 0.006464
converged
initial value 3.320317
iter 5 value 0.045467
final value 0.000864
converged

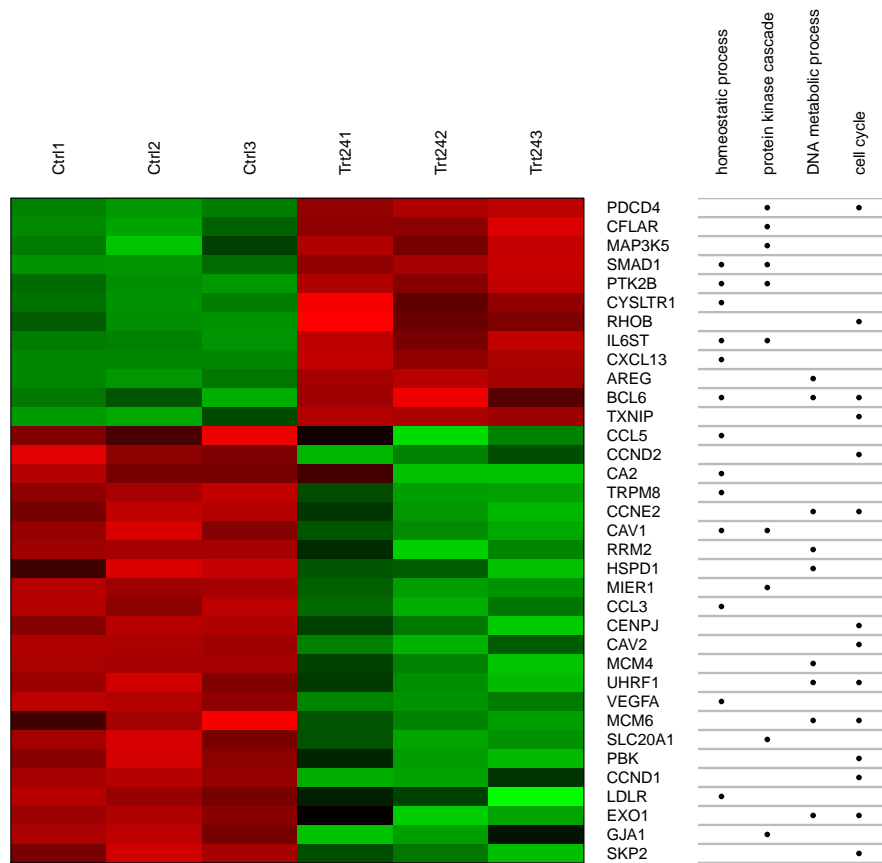


Figure 19: Customized concept-gene cross tabulation

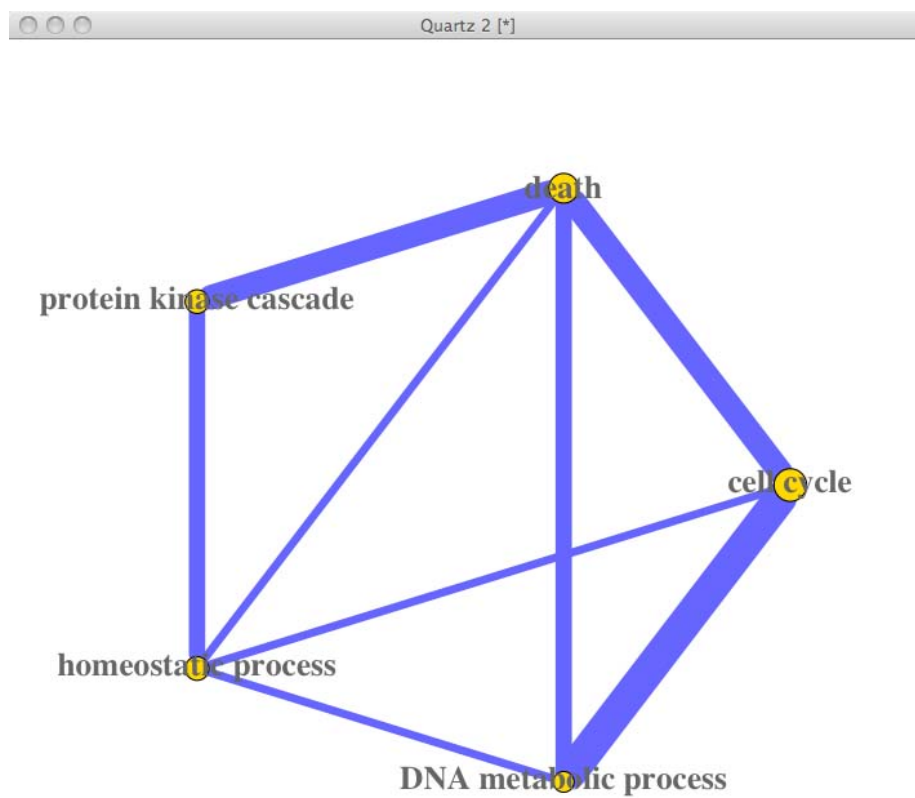


Figure 20: Screen shot of customized GO category linkage


```

6352    CCL5  -2.154827 1.092361e-03
1524   CX3CR1  2.127494 4.412558e-06
3572   IL6ST  2.213384 8.018338e-08
10563 CXCL13 10.688601 1.073456e-10

```

```
[1] "***** Focal adhesion genes in Category : 7 *****"
```

```

Symbol foldChange      pValue
3696   ITGB8  -3.097913 1.178489e-06
858    CAV2   -2.643498 2.625770e-08
7422   VEGFA  -2.391115 7.147830e-09
595    CCND1  -2.257123 5.932920e-07
894    CCND2  -2.239806 1.208734e-06
857    CAV1   -2.122246 3.842276e-06
824    CAPN2   2.568772 1.890638e-08

```

```
[1] "***** Cell cycle genes in Category : 6 *****"
```

```

Symbol foldChange      pValue
9134   CCNE2  -2.379943 3.326011e-06
4175   MCM6   -2.356668 2.080808e-04
6502   SKP2   -2.276824 1.013445e-05
595    CCND1  -2.257123 5.932920e-07
894    CCND2  -2.239806 1.208734e-06
4173   MCM4   -2.082429 8.413734e-06

```

```
[1] "***** Chemokine signaling pathway genes in Category : 6 *****"
```

```

Symbol foldChange      pValue
6348   CCL3   -4.031781 1.147014e-07
3627   CXCL10 -2.376811 1.315856e-07
6352   CCL5   -2.154827 1.092361e-03
2185   PTK2B   2.036789 1.970066e-07
1524   CX3CR1  2.127494 4.412558e-06
10563 CXCL13 10.688601 1.073456e-10

```

```
> ## print and save top 10 DOLites information
```

```
> topDOLiteGenes(z, orderby='pvalue', top=5, topGenes='ALL', genesOrderBy='pValue', file=T
```

```
[1] "***** Hyperlipidemia p value : 0.00198151640638039 *****"
```

```

Symbol foldChange      pValue
10628 TXNIP   3.899457 4.296784e-09
6319   SCD    -2.480571 6.976419e-07
2181   ACSL3  -2.049592 5.282896e-04
6352   CCL5   -2.154827 1.092361e-03

```

```
[1] "***** Prostate cancer p value : 0.00396329360969882 *****"
```

```

Symbol foldChange      pValue
374    AREG   17.553825 1.241073e-11
262    AMD1   -2.538954 1.216657e-09
3490   IGFBP7  3.787547 2.688423e-08
5087   PBX1    2.552036 1.120693e-07
3627   CXCL10 -2.376811 1.315856e-07

```

```
[1] "***** Alveolar bone loss p value : 0.00872917145369052 *****"
```

```

Symbol foldChange      pValue
6348   CCL3   -4.031781 1.147014e-07
6678   SPARC  -2.289622 7.751533e-04

```

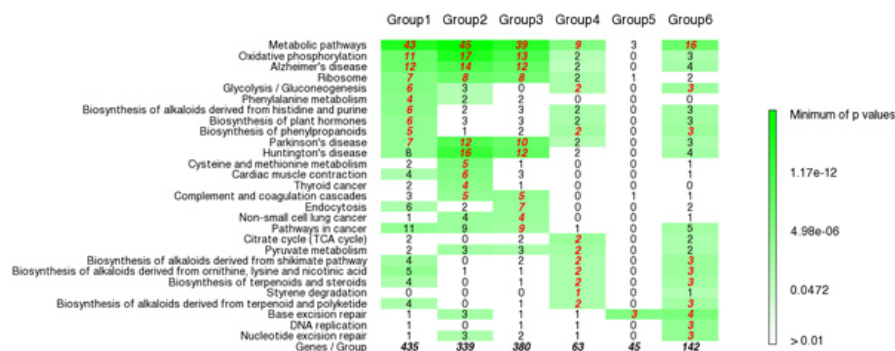


Figure 21: Screen shot of multigroup genes KEGG analysis

```
[1] "***** Leukodystrophy NOS   p value : 0.0114791957044022 *****"
      Symbol foldChange      pValue
411   ARSB    -2.122207 3.199329e-06
3329 HSPD1    -2.144728 7.145728e-05
[1] "***** Bronchiolitis      p value : 0.0114791957044022 *****"
      Symbol foldChange      pValue
6348  CCL3    -4.031781 1.147014e-07
6352  CCL5    -2.154827 1.092361e-03
[1] "File topCategoryGenes.txt is successfully generated!"
```

5.3 Multigroup Genes Analysis

Several groups of genes are used to dynamically study biological processes by different treatment or time points. In this case, *GeneAnswers* provide a solution to integrate enrichment test information of different groups of genes.

```
> ##load multigroup genes sample data
> data(sampleGroupsData)
> ##Build a GeneAnswers List
> gAKEGGL <- lapply(sampleGroupsData, geneAnswersBuilder, 'org.Hs.eg.db', categoryType='KEGG')

[1] "GeneAnswers instance has been successfully generated!"
[1] "GeneAnswers instance has been successfully generated!"
[1] "GeneAnswers instance has been successfully generated!"
[1] "GeneAnswers instance has been successfully generated!"
[1] "GeneAnswers instance has been successfully generated!"
[1] "GeneAnswers instance has been successfully generated!"

> ##Output integrated text table
> output<- getConceptTable(gAKEGGL, items='geneNum')
```

Function groupReport can generate a html file including all information.

5.4 Homologous Gene Mapping

Since DOLite is developed for human, any gene from other species can not take advantage of this novel annotation database. Therefore, *GeneAnswers* package

provides two functions for this type of data interpretation. *getHomoGeneIDs* can map other species gene Entrez IDs to human homologous gene Entrez IDs at first. Then users can perform normal *GeneAnswers* functions. Finally, function *geneAnswersHomoMapping* maps back to original species gene Entrez IDs. Current version supports two types of homologous gene mapping. One is called "direct", which is simple and only works between mouse and human. Since all of human gene symbols are capitalized, while only first letter of mouse homologous gene symbols is uppercase, this method simply maps homologous genes by capitalized mouse gene symbols. Another method adopts *biomaRt* to do mapping. *biomaRt* contacts its online server to mapping homologous genes. Its database include more accurate information, but it might take longer to do that, while 'direct' method can rapidly do conversation though it is possible to miss some information.

```
> ## load mouse example data
> data('mouseExpr')
> data('mouseGeneInput')
> mouseExpr[1:10,]
```

	GeneID	S11	S12	S13	S21	S22	S23
1	93695	11.140745	11.555394	11.199022	13.53989	13.68489	13.52166
2	20750	10.378364	10.780340	10.280152	12.51370	12.77777	12.72755
3	16854	10.576541	10.823445	10.539105	12.52568	12.94808	12.75282
4	20210	10.417790	10.503403	10.603501	12.38010	12.64376	12.45370
5	14282	9.392208	9.574147	9.456061	11.47399	11.24749	11.42666
6	17105	10.599174	11.078450	10.565310	12.47790	12.79757	12.50897
7	17110	12.674773	13.153840	12.672851	14.56094	14.89131	14.57835
8	16002	11.766943	12.268368	11.557304	13.42105	13.62164	13.60838
9	21924	8.874513	9.096380	8.860733	10.46360	10.66965	10.62615
10	269994	10.913894	10.330857	10.853911	9.07294	9.07630	9.04366

```
> mouseGeneInput[1:10,]
```

	Symbol	foldChange	pValue
93695	93695	4.869452	1.864011e-08
20750	20750	4.573777	1.224957e-07
16854	16854	4.274721	6.526113e-08
20210	20210	3.956676	4.098411e-09
14282	14282	3.754383	3.190981e-09
17105	17105	3.597932	1.088294e-06
17110	17110	3.587662	1.035619e-06
16002	16002	3.217968	5.465650e-06
21924	21924	3.122260	2.337725e-08
269994	269994	-3.106423	1.962161e-06

```
> ## only keep first one for one to more mapping
> pickHomo <- function(element, inputV) {return(names(inputV[inputV == element])[1])}
> ## mapping geneInput to homo entrez IDs.
> homoLL <- getHomoGeneIDs(mouseGeneInput[,1], species='mouse', speciesL='human', mappingM

[1] "Warning: homogenes of some input genes can not be found and are removed!!!"
```

```

> newGeneInput <- mouseGeneInput[mouseGeneInput[,1] %in% unlist(lapply(unique(homoLL), pic
> dim(mouseGeneInput)

[1] 71 3

> dim(newGeneInput)

[1] 67 3

> newGeneInput[,1] <- homoLL[newGeneInput[,1]]
> ## mapping geneExpr to homo entrez IDs.
> homoLLExpr <- getHomoGeneIDs(as.character(mouseExpr[,1]), species='mouse', speciesL='hum

[1] "Warning: homologues of some input genes can not be found and are removed!!!"

> newExpr <- mouseExpr[as.character(mouseExpr[,1]) %in% unlist(lapply(unique(homoLLExpr) ,
> newExpr[,1] <- homoLLExpr[as.character(newExpr[,1])]
> dim(mouseExpr)

[1] 71 7

> dim(newExpr)

[1] 67 7

> ## build a GeneAnswers instance based on mapped data
> v <- geneAnswersBuilder(newGeneInput, 'org.Hs.eg.db', categoryType='DOLite', testType='h

[1] "geneInput has built in ..."
[1] "annLib and categoryType have built in ..."
[1] "genesInCategory has built in ..."
[1] "Enrichment test is only performed based on annotated genes"
[1] "testType, pvalueT and enrichmentInfo have built in ..."
[1] "geneExpressionProfile has been built in ..."
[1] "GeneAnswers instance has been successfully generated!"

> ## make the GeneAnswers instance readable, only map DOLite IDs to terms
> vv <- geneAnswersReadable(v, geneSymbol=F)

[1] "Mapping genesInCategory ..."
[1] "Mapping enrichmentInfo rownames ..."

> getAnnLib(vv)

[1] "org.Hs.eg.db"

> ## mapping back to mouse genes
> uu <- geneAnswersHomoMapping(vv, species='human', speciesL='mouse', mappingMethod='direc

[1] "Change annLib ..."
[1] "Mapping geneInput ..."
[1] "Mapping genesInCategory ..."
[1] "Mapping geneExprProfile ..."

```

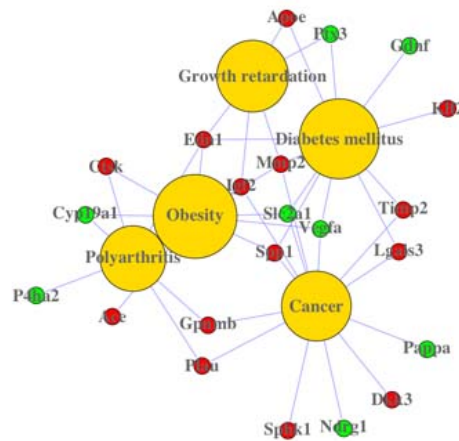


Figure 22: Screen shot of homogene DOLite-gene network

```
> getAnnLib(uu)

[1] "org.Mm.eg.db"

> ## make mapped genes readable, DOLite terms are not mapped
> u <- geneAnswersReadable(uu, catTerm=FALSE)

[1] "Mapping geneInput ..."
[1] "Mapping genesInCategory ..."
[1] "Mapping geneExprProfile rownames ..."

> ## sort new GeneAnswers instance
> u1 <- geneAnswersSort(u, sortBy='pvalue')

> ## plot concept-gene network
> geneAnswersConceptNet(u, colorValueColumn='foldChange', centroidSize='pvalue', output='f

> ## plot homogene DOLite-gene cross tabulation
> geneAnswersHeatmap(u1)

> ## output top information
> topDOLiteGenes(u, geneSymbol=FALSE, catTerm=FALSE, orderby='pvalue', top=6, topGenes='AL

[1] "***** Growth retardation   p value : 1.86018082726357e-06 *****"
      Symbol foldChange      pValue
```

```

initial value 0.561792
iter 5 value 0.185166
iter 10 value 0.055120
iter 15 value 0.051535
iter 20 value 0.035624
final value 0.009132
converged
initial value 16.218142
iter 5 value 9.824398
final value 9.383898
converged

```

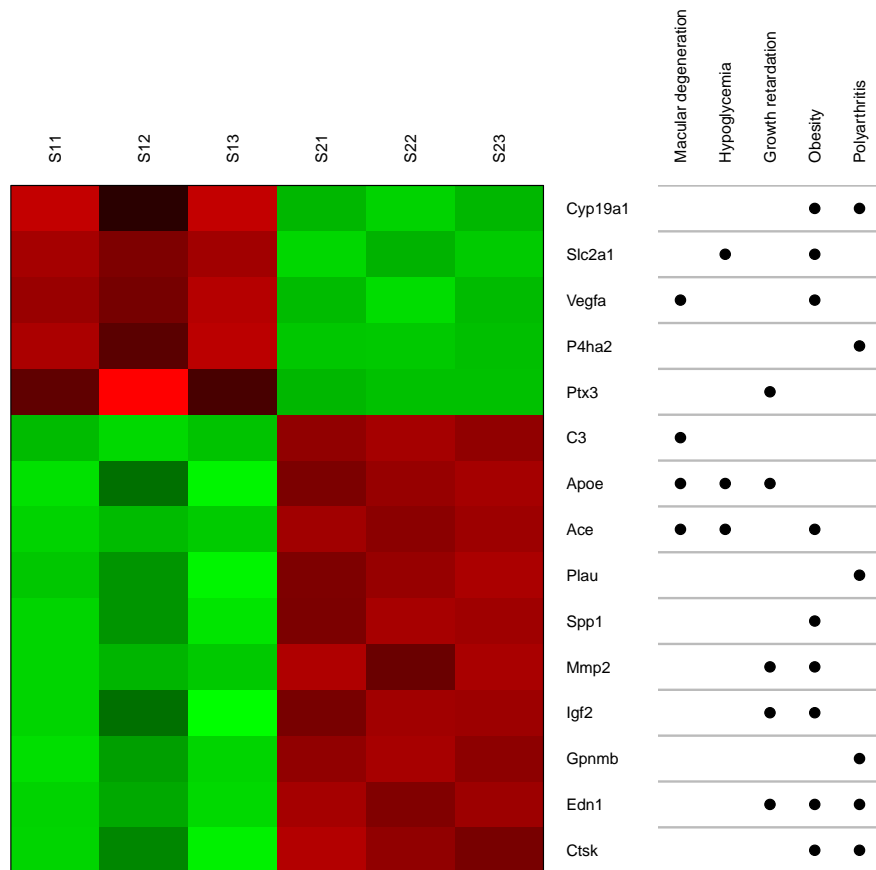


Figure 23: homogene DOLite-gene cross tabulation

```

13614   Edn1    2.504625 1.536084e-08
17390   Mmp2    2.932919 5.771626e-08
11816   Apoe    2.465045 4.135459e-06
16002   Igf2    3.217968 5.465650e-06
19288   Ptx3   -2.100947 4.364358e-05
[1] "***** Obesity   p value : 5.28156537991024e-06 *****"
      Symbol foldChange      pValue
11421   Ace    2.897884 4.811431e-10
20525 Slc2a1  -2.590159 6.290580e-09
13614   Edn1    2.504625 1.536084e-08
22339 Vegfa   -2.535157 3.651889e-08
17390   Mmp2    2.932919 5.771626e-08
[1] "***** Hypoglycemia p value : 9.69236642456046e-06 *****"
      Symbol foldChange      pValue
11421   Ace    2.897884 4.811431e-10
20525 Slc2a1  -2.590159 6.290580e-09
11816   Apoe    2.465045 4.135459e-06
[1] "***** Polyarthriti p value : 2.85339797023959e-05 *****"
      Symbol foldChange      pValue
13614   Edn1    2.504625 1.536084e-08
93695 GpnmB    4.869452 1.864011e-08
18452 P4ha2   -2.584996 3.363470e-07
18792 Plau    2.456354 4.029683e-07
13038 Ctsk    2.220556 1.394015e-06
[1] "***** Macular degeneration p value : 3.4593344673306e-05 *****"
      Symbol foldChange      pValue
11421   Ace    2.897884 4.811431e-10
12266   C3     2.580510 1.185513e-09
22339 Vegfa   -2.535157 3.651889e-08
11816   Apoe    2.465045 4.135459e-06
[1] "***** Capillaries disease p value : 5.6872995821234e-05 *****"
      Symbol foldChange      pValue
13614   Edn1    2.504625 1.536084e-08
22339 Vegfa   -2.535157 3.651889e-08
17390   Mmp2    2.932919 5.771626e-08
[1] "File topCategoryGenes.txt is successfully generated!"

```

6 Session Info

```
> toLatex(sessionInfo())
```

- R version 2.11.0 (2010-04-22), i386-pc-mingw32
- Locale: LC_COLLATE=English_United States.1252,
LC_CTYPE=English_United States.1252,
LC_MONETARY=English_United States.1252, LC_NUMERIC=C,
LC_TIME=English_United States.1252
- Base packages: base, datasets, graphics, grDevices, methods, stats, tools,
utils

- Other packages: annotate 1.26.0, AnnotationDbi 1.10.1, Biobase 2.8.0, bitops 1.0-4.1, DBI 0.2-5, GeneAnswers 1.4.2, GO.db 2.4.1, Heatplus 1.18.0, igraph 0.5.3, KEGG.db 2.4.1, MASS 7.3-5, org.Hs.eg.db 2.4.1, org.Mm.eg.db 2.4.1, RColorBrewer 1.0-2, RCurl 1.4-2, rgl 0.91, RSQLite 0.9-0, XML 3.1-0
- Loaded via a namespace (and not attached): xtable 1.5-6

7 Acknowledgments

We would like to thank the users and researchers around the world contribute to the *GeneAnswers* package, provide great comments and suggestions and report bugs

8 References

Du, P., Feng, G., Flatow, J., Song, J., Holko, M., Kibbe, W.A. and Lin, S.M., (2009) 'From disease ontology to disease-ontology lite: statistical methods to adapt a general-purpose ontology for the test of gene-ontology associations', *Bioinformatics* 25(12):i63-8

Feng, G., Du, P., Krett, N.L., Tessel, M., Rosen, S., Kibbe, W.A., and Lin, S.M., (submitted) 'Bioconductor Methods to Visualize Gene-list Annotations',