GeneAnswers, Integrated Interpretation of Genes

Gang Feng[‡][*], Pan Du[‡][†], Warren A. Kibbe[‡][‡], Simon Lin[‡][§]

October 3, 2014

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

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 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:

[*]g-feng (at) northwestern.edu
[†]dupan (at) northwestern.edu
[‡]wakibbe (at) northwestern.edu
[§]s-lin2 (at) northwestern.edu

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* pakcage in the Bioconductor developing version, you can use the following command:

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

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.

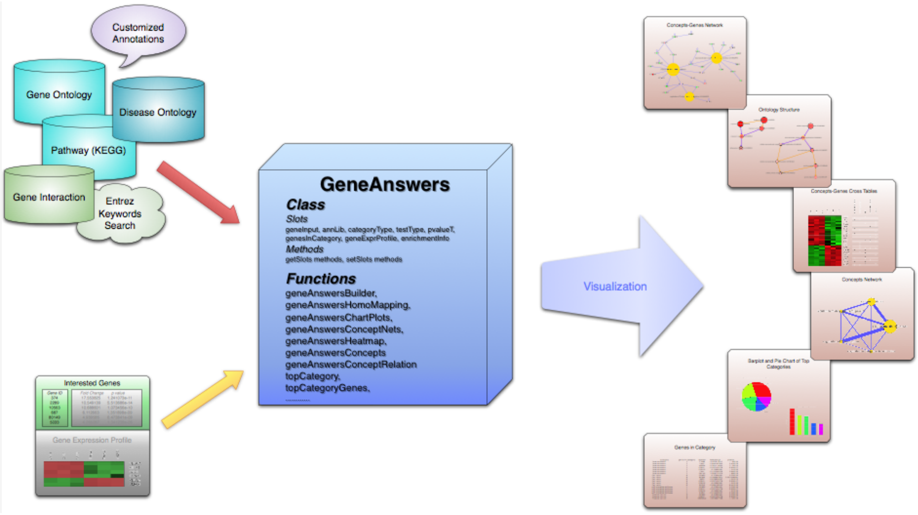


Figure 1: Flow chart of GeneAnswers

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 eUtilis 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. ¡¡load library, eval=T¿¿= library(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.

¡¡build GeneAnswers, echo=T, eval=T¿¿= data('humanGeneInput') data('humanExpr') build a GeneAnswers instance with statistical test based on biological process of GO and saved example data. x ¡- geneAnswersBuilder(humanGeneInput, 'org.Hs.eg.db', categoryType='GO.BP', testType='hyperG', pvalueT=0.1, FDR.correction=TRUE, geneExpressionProfile=humanExpr) class(x) 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', pvalueT=0.1, geneExpressionProfile=humanExpr, verbose=FALSE) 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', pvalueT=0.1, geneExpressionProfile=humanExpr, verbose=FALSE) w ¡- geneAnswersBuilder(humanGeneInput, 'org.Hs.eg.db', categoryType='GO.BP', testType='hyperG', pvalueT=0.1, FDR.correction=TRUE, geneExpressionProfile=humanExpr, level=2, verbose=FALSE) @

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. ¡¡build customized GeneAnswers, echo=T, eval=T¿¿= before running the following codes, make sure that you can connect the internet. keywordsList ¡- list(Apoptosis=c('apoptosis'), CellAdhesion=c('cell adhesion')) entrezIDList ¡- searchEntrez(keywordsList) q ¡- geneAnswersBuilder(humanGeneInput, entrezIDList, testType='hyperG', totalGeneNumber = 45384, pvalueT=0.1, geneExpressionProfile=humanExpr, verbose=FALSE) class(q) getAnnLib(q) getCategoryType(q) @

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 ĜeneAnswersBuilder 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.

¡¡make GeneAnswers readable and generate concept-gene network, echo=T¿¿= mapping gene IDs and category IDs to gene symbols and category terms xx ¡- geneAnswersReadable(x) yy ¡- geneAnswersReadable(y, verbose=FALSE) zz ¡- geneAnswersReadable(z, verbose=FALSE) ww ¡- geneAnswersReadable(w, verbose=FALSE) before running the following codes, make sure that you can connect the internet. q ¡- setAnnLib(q, ’org.Hs.eg.db’) qq ¡- geneAnswersReadable(q, catTerm=FALSE) @

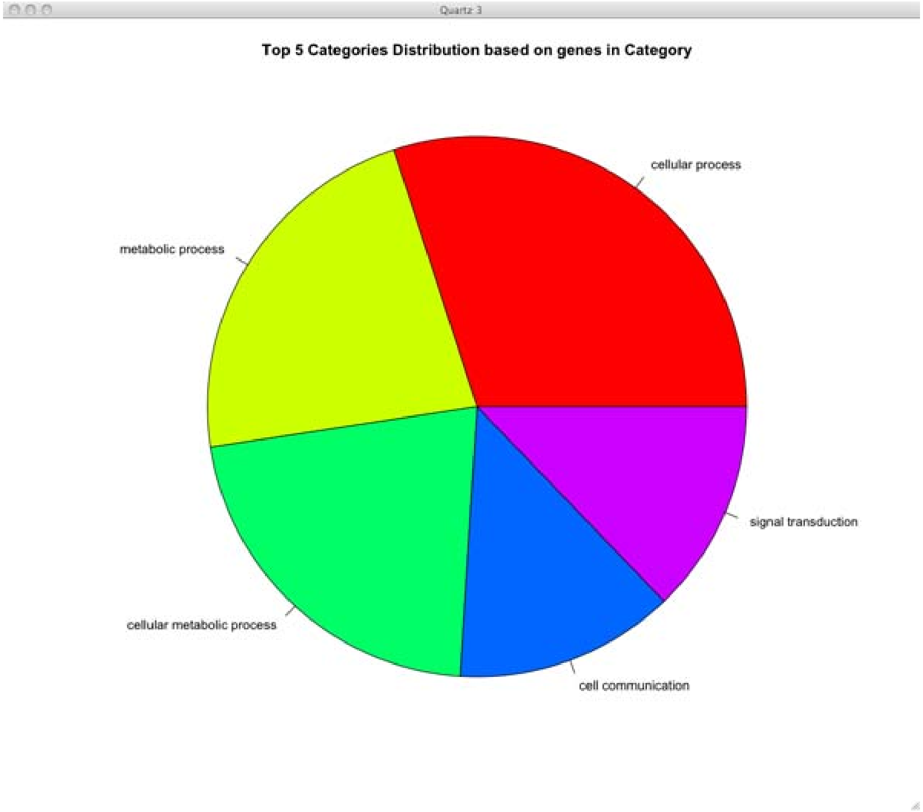
Since function *geneAnswersReadable* implements mapping based on annotation database in slot annLib, we assign ’org.Hs.eg.db’ to customized GeneAnswers instance annLib slot at first for make it readable.

¡¡plot barplot and / or piechart, echo=T, eval=F¿¿= 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. ¡¡plot interactive concept-gene network, echo=T, eval=F¿¿= plot interactive concept-gene network geneAnswersConceptNet(xx, colorValueColumn=’foldChange’, centroidSize=’pvalue’, output=’interactive’) @

The top 5 categories might not be very specific. Users might get a tree view to see relative category terms by calling function geneAnswersConceptRelationif 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, echo=T, eval=F¿¿= plot interactive go structure network geneAnswersConceptRelation(x, direction=’both’, netMode=’connection’, catTerm=TRUE, catID=TRUE, nameLength=15) @

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 with gene interaction, echo=T, eval=F¿¿= plot interactive concept-gene network geneAnswersConceptNet(x, color=’foldChange’,



Quartz 3
Top 5 Categories Distribution based on genes in Category
cellular process
metabolic process
signal transduction
cellular metabolic process
cell communication

Figure 2: Screen shot of pie chart of top categories

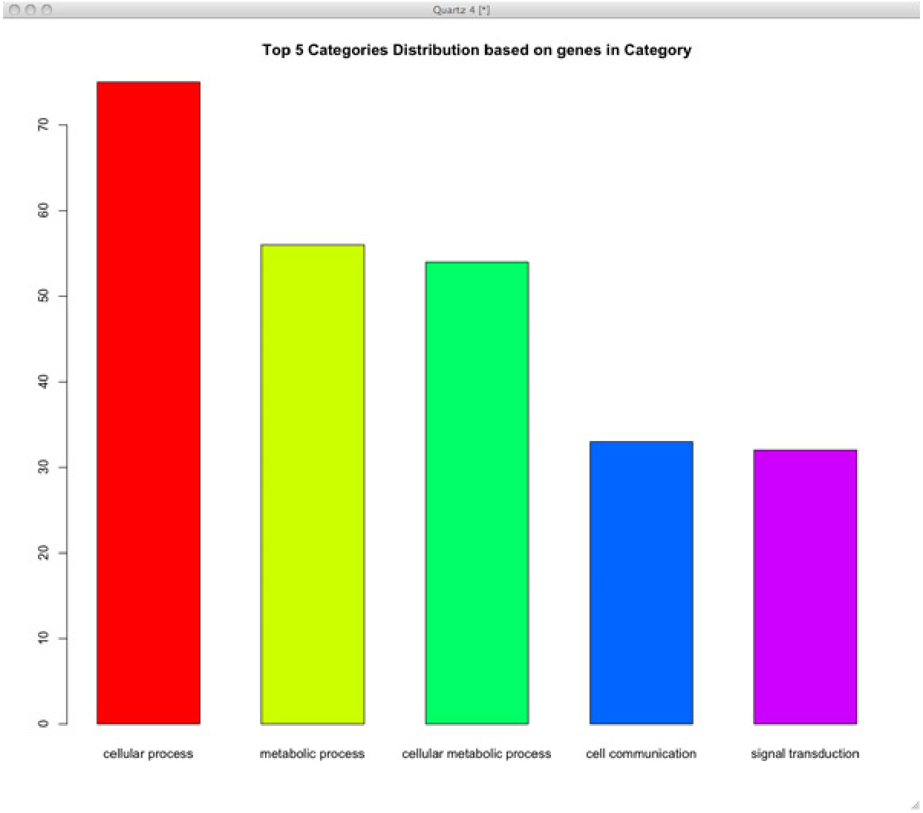


Figure 3: Screen shot of barplot of top categories

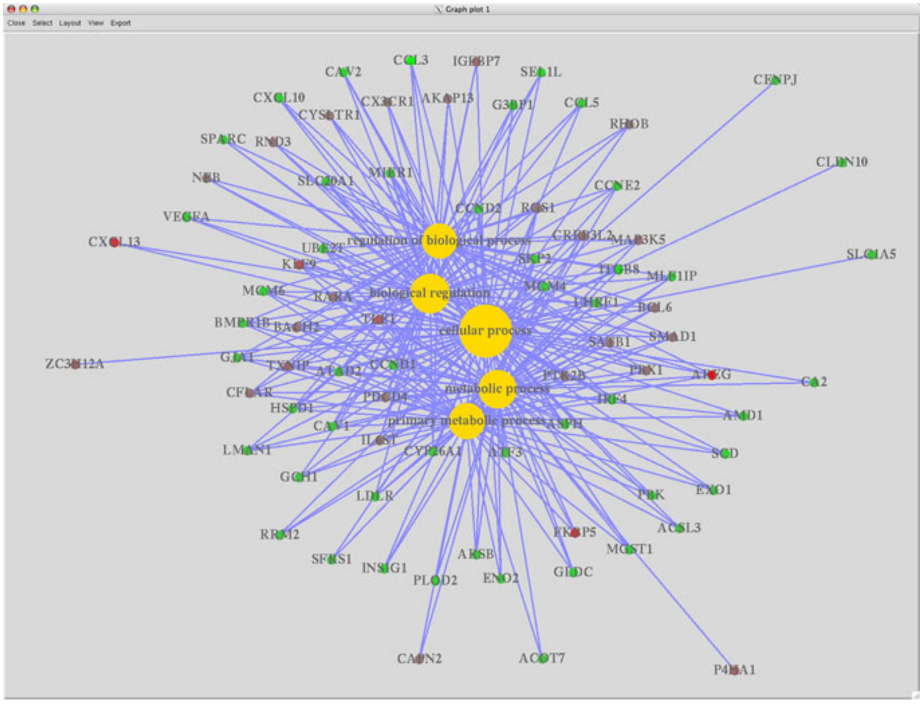


Figure 4: Screen shot of concept-gene network

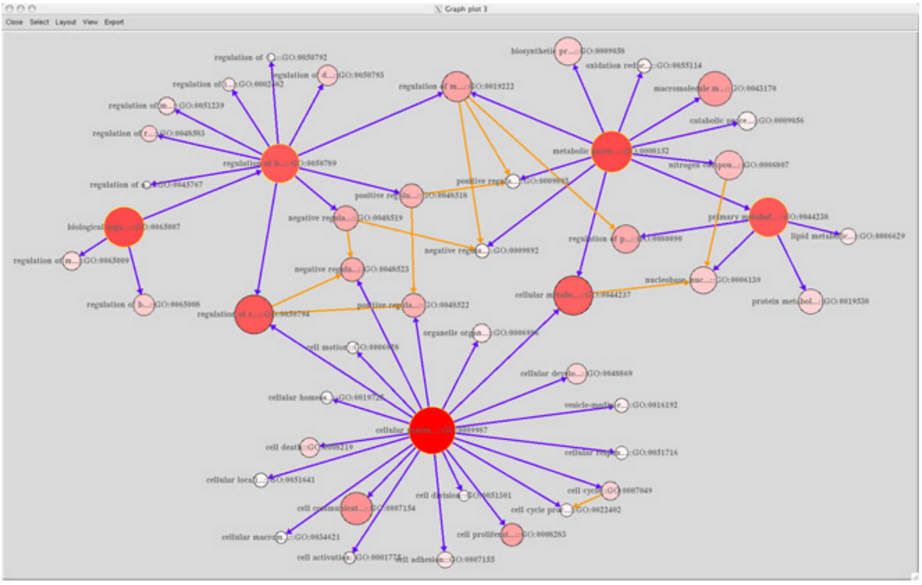

Graph plot 3
Close
Select
Layout
View
Export
regulation of ...::GO:0050792
regulation of d...::GO:0050793
regulation of l...::GO:0002682
regulation of m...::GO:0051239
regulation of r...::GO:0048583
regulation of b...::GO:0050789
regulation of ...::GO:0045767
biological regu...::GO:0065007
regulation of m...::GO:0065009
regulation of b...::GO:0065008
regulation of r...::GO:0050794
regulation of m...::GO:0019222
positive regula...::GO:0048518
negative regula...::GO:0048519
negative regula...::GO:0048523
positive regula...::GO:0048522
positive regula...::GO:0009893
negative regula...::GO:0009892
biosynthetic pr...::GO:0009058
oxidation reduc...::GO:0055114
macromolecule m...::GO:0043170
catabolic proce...::GO:0009056
metabolic proce...::GO:0008152
nitrogen compou...::GO:0006807
primary metabol...::GO:0044238
lipid metabolic...::GO:0006629
regulation of p...::GO:0080090
nucleobase, nuc...::GO:0006139
cellular metabo...::GO:0044237
protein metabol...::GO:0019538
organelle organ...::GO:0006996
cell motion::GO:0006928
cellular develo...::GO:0048869
cellular homeos...::GO:0019725
vesicle-mediate...::GO:0016192
cellular proces...::GO:0009987
cell death::GO:0008219
cellular respon...::GO:0051716
cellular locali...::GO:0051641
cell division::GO:0051301
cell cycle::GO:0007049
cell communicat...::GO:0007154
cell cycle pro...::GO:0022402
cellular macrom...::GO:0034621
cell prolifera...::GO:0008283
cell activation::GO:0001775
cell adhesion::GO:0007155

Figure 5: Screen shot of go structure network

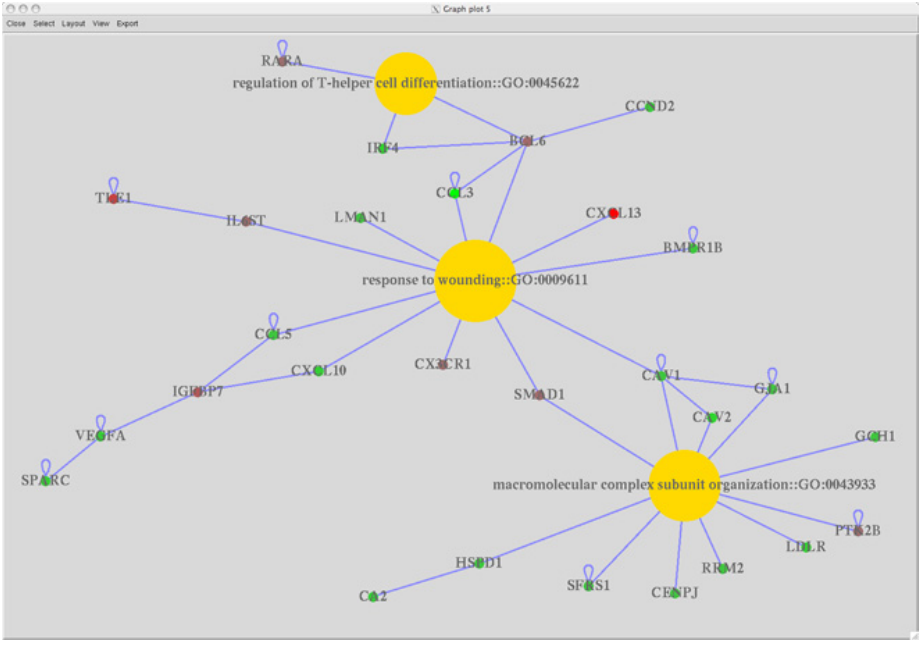


Figure 6: Screen shot of concept-gene network

geneLayer=5, output='interactive', showCats=c("GO:0009611", "GO:0043933", "GO:0045622"), catTerm=TRUE, geneSymbol=TRUE, catID=TRUE) @

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 Gene interaction, echo=T, eval=F¿¿= plot the given gene interaction buildNet(c('444','3638', '5087','55835'), idType='GeneInteraction', layers=2, filterGraphIDs=getGeneInput(x)[,1], filterLayer=2, netMode='connection') @

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 Gene interaction with expression information, echo=T, eval=F¿¿= plot the given gene interaction buildNet(c('444','3638', '5087','55835'), idType='GeneInteraction', layers=2, filterGraphIDs=getGeneInput(x)[,1:2], filterLayer=2, netMode='connection') @

The following one is based on expression p-values, we often use -log2 to transform p-values. ¡¡plot Gene interaction with pvalues, echo=T, eval=F¿¿= plot the given gene interaction buildNet(c('444','3638', '5087','55835'), idType='GeneInteraction', layers=2, filterGraphIDs=cbind(getGeneInput(x)[,1], -log2(getGeneInput(x)[,3])), filterLayer=2, netMode='connection') @

Users can also define customized color scheme and transfer it into 'buildNet'

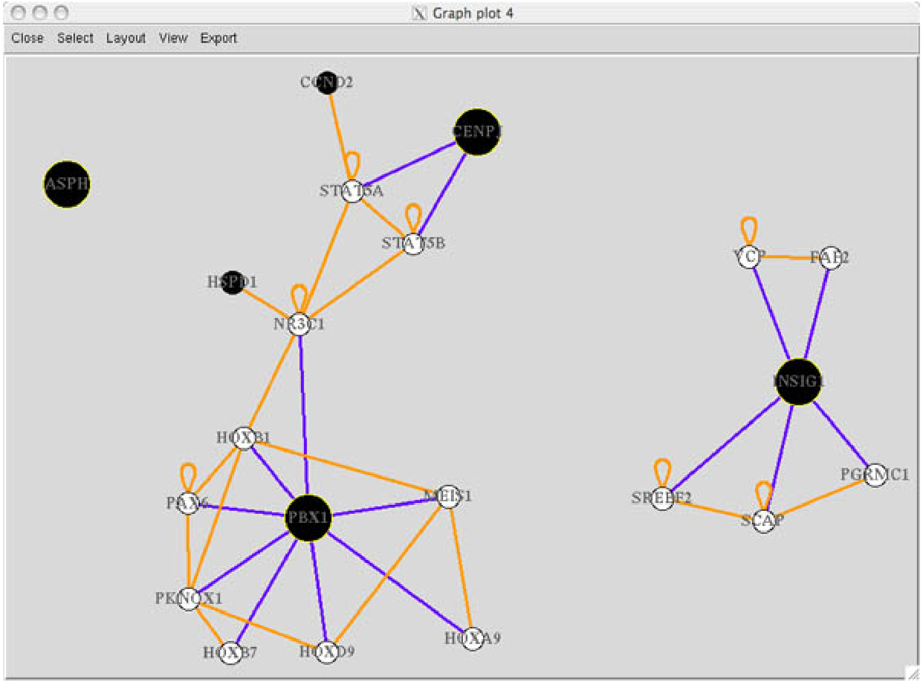


Figure 7: Screen shot of the given gene interaction network

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 Gene interaction for all of given genes, echo=T, eval=F¿¿= plot the given gene interaction buildNet(getGeneInput(x)[,1], idType='GeneInteraction', layers=2, filterGraphIDs=getGeneInput(x)[,1:2], filterLayer=2, netMode='connection') @

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 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, echo=T, eval=F¿¿= plot Go-concept network for 2 level nodes removal geneAnswersConceptNet(ww, colorValueColumn='foldChange', centroidSize='pvalue', output='fixed') @

Also, users can sort enrichment test information and plot it. ¡¡sort GeneAnswers, echo=T, eval=T¿¿= sort enrichmentInfo dataframe by fdr adjusted p value xxx ¡- geneAnswersSort(xx, sortBy='correctedPvalue') yyy ¡- geneAnswersSort(yy, sortBy='pvalue') zzz ¡- geneAnswersSort(zz, sortBy='geneNum') @

¡¡plot concept-gene networks, echo=T, eval=F¿¿= geneAnswersConceptNet(yyy, colorValueColumn='foldChange', centroidSize='geneNum', output='fixed') geneAnswersConceptNet(zzz, colorValueColumn='foldChange', centroidSize='pvalue', output='fixed', showCats=c(10:16)) @

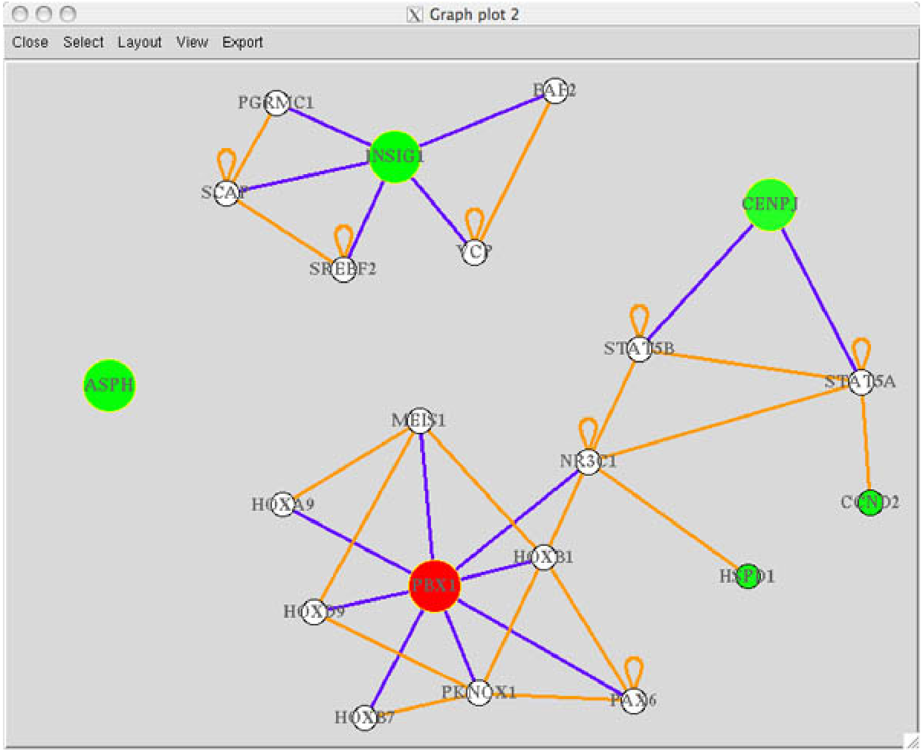


Figure 8: Screen shot of the given gene interaction network with expression information

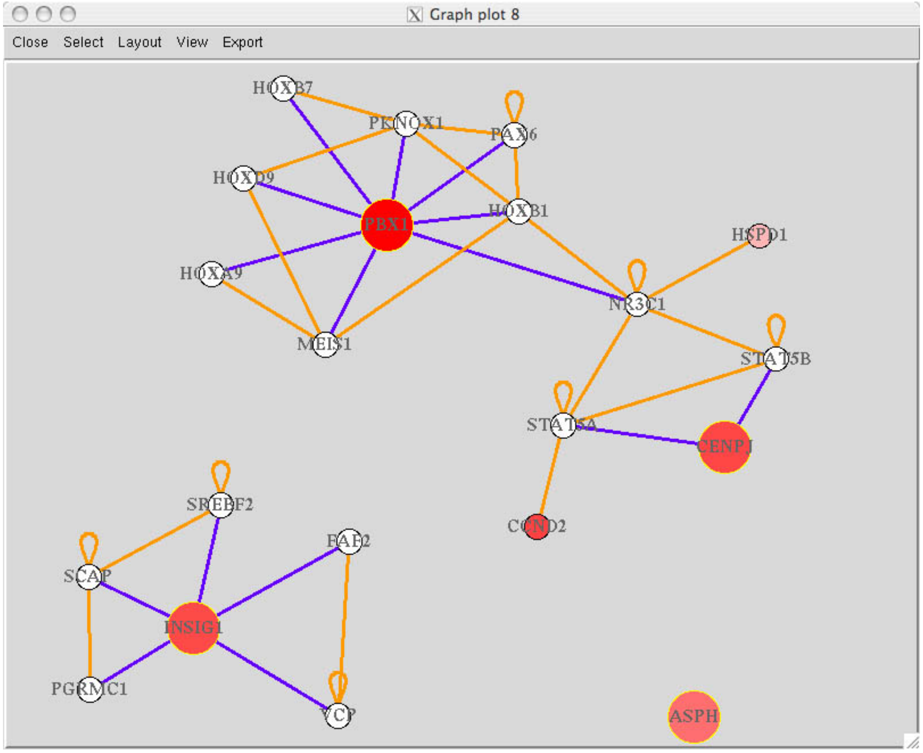


Figure 9: Screen shot of the given gene interaction network with p-value information

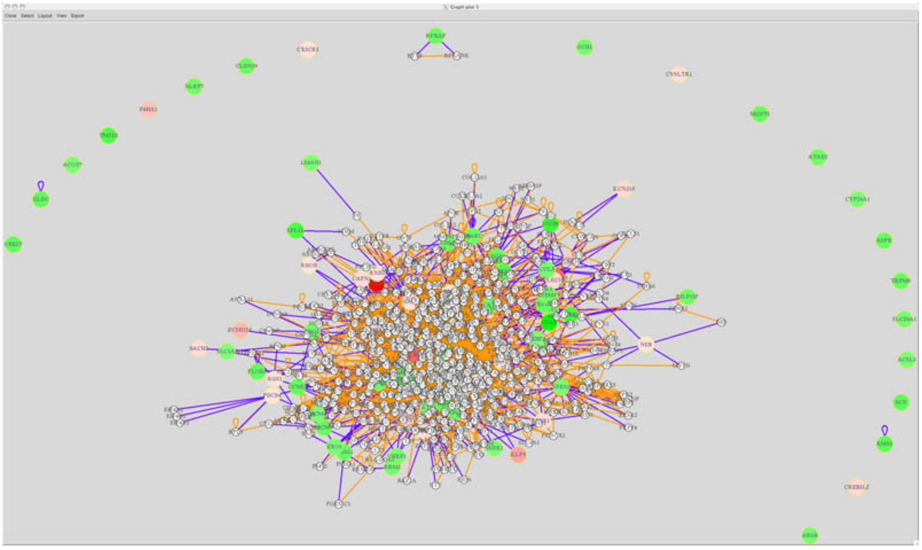


Figure 10: Screen shot of all of the given gene interaction network with expression information

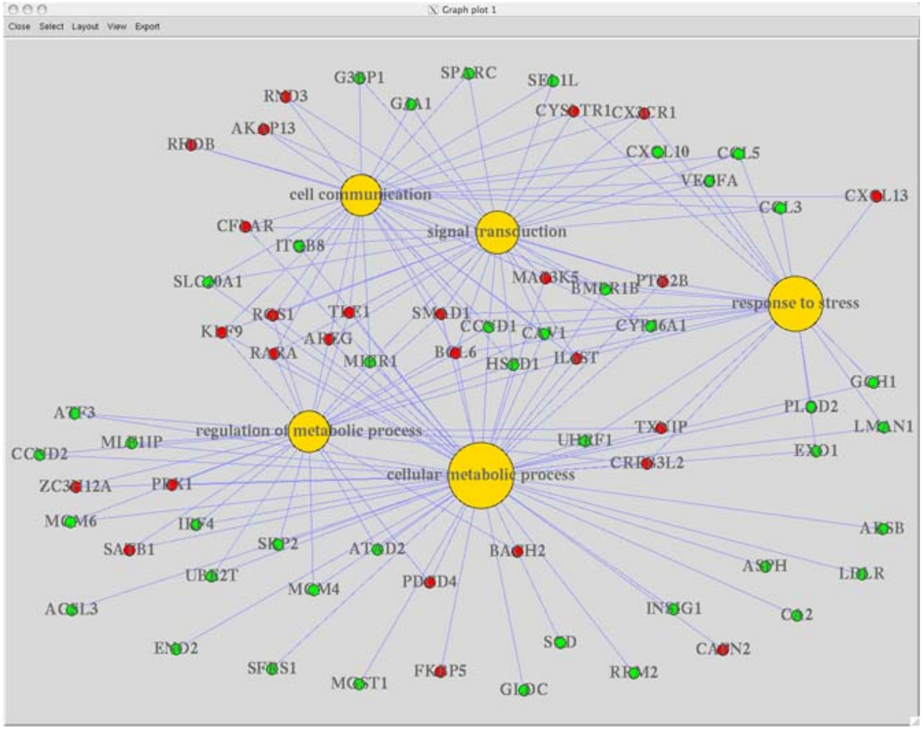


Figure 11: Screen shot of concept-gene network for top 2 GO level nodes removal

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, echo=T, eval=F¿¿= generate GO-gene cross tabulation geneAnswersHeatmap(x, catTerm=TRUE, geneSymbol=TRUE) @

¡¡genrate KEGG-gene cross tabulation echo=T, eval=F¿¿= 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. ¡¡generate DOLite-gene cross tabulation, echo=F, eval=F¿¿= geneAnswersHeatmap(zzz, mapType='heatmap') @

Besides top categories, users can also show interested categories. ¡¡plot customized concept-gene cross tabulation, echo=T, eval=T¿¿= GOBPIDs ¡- c("GO:0043627", "GO:0042493", "GO:0006259", "GO:0007243") GOBPTerms ¡- c("response to estrogen stimulus", "response to drug", "protein kinase cascade", "DNA metabolic process") @

¡¡generate concept-gene cross tabulation, echo=T, eval=F¿¿= generate concept-gene cross tabulation geneAnswersConceptNet(x, colorValueColumn='foldChange', centroidSize='pvalue', output='fixed', showCats=GOBPIDs, catTerm=TRUE, geneSymbol=TRUE) @

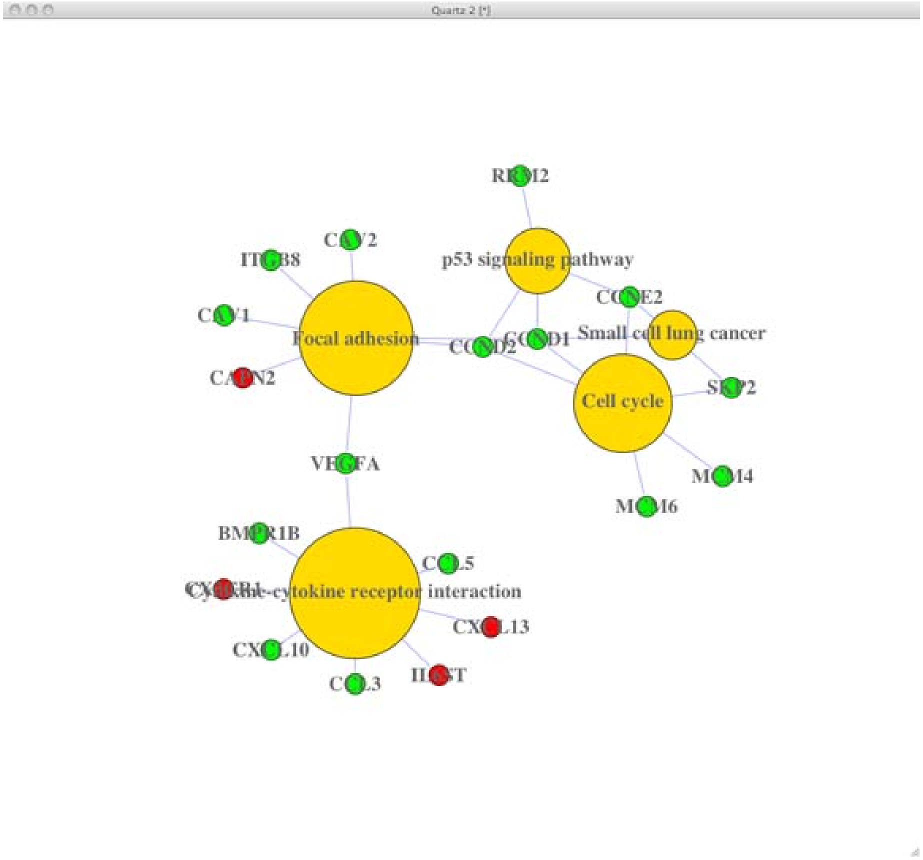


Figure 12: Screen shot of KEGG-gene network

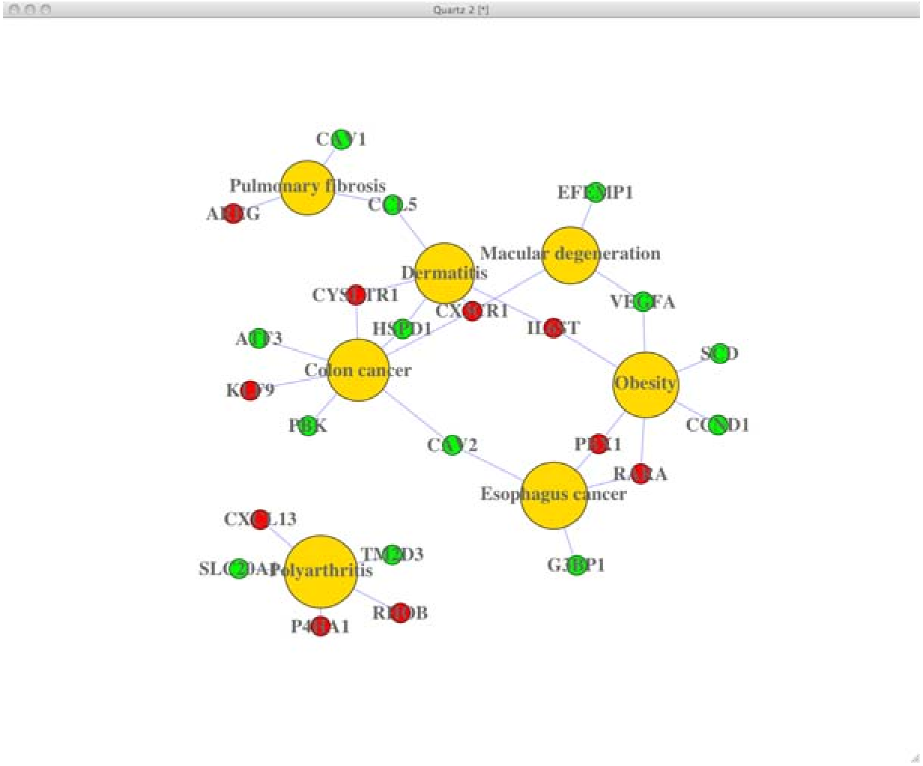


Figure 13: Screen shot of DOLite-gene network

¡¡fig=true, width=8, height=8, echo=T, eval=T¿¿= generate GO-gene cross tabulation geneAnswersHeatmap(x, catTerm=TRUE, geneSymbol=TRUE) @

Figure 14: GO-gene cross tabulation

¡¡fig=true, width=8, height=8, quiet=F, echo=T, eval=T¿¿= geneAnswersHeatmap(yyy) @

Figure 15: KEGG-gene cross tabulation

¡¡fig=true, width=8, height=8, quiet=T, echo=F, eval=T¿¿= geneAnswersHeatmap(zzz, mapType='heatmap') @

Figure 16: DOLite-gene cross tabulation

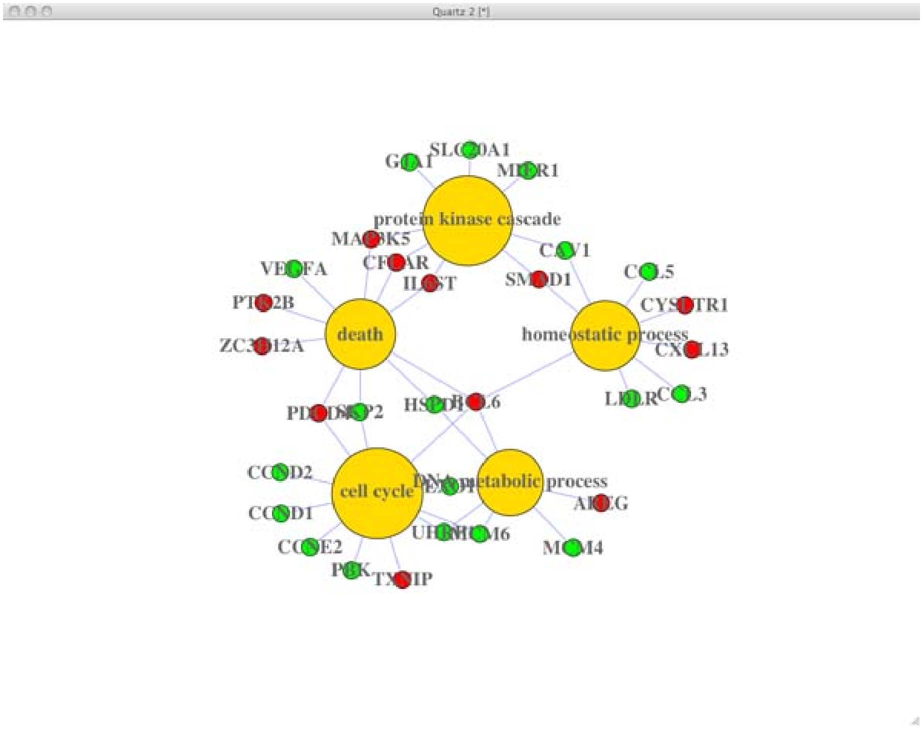


Figure 17: Screen shot of customized GO-gene network

¡¡fig=true, width=8, height=8, quiet=T, echo=F, eval=T¿¿= geneAnswersHeatmap(x, showCats=GOBPIDs, catTerm=TRUE, geneSymbol=TRUE) @

Figure 18: Customized concept-gene cross tabulation

¡¡generate customized concept-gene cross tabulation, echo=T, eval=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, echo=T, eval=F¿¿= 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 categories and genes, echo=T, eval=T¿¿= print top GO categories sorted by hypergeometric test p value topGOGenes(x, orderby='pvalue') print top KEGG categories sorted by gene numbers and sort genes by fold changes topPATHGenes(y, orderby='geneNum', top=4, topGenes=8, genesOrderBy='foldChange') print and save top 10 DOLites information topDOLITEGenes(z, orderby='pvalue', top=5, topGenes='ALL', genesOrderBy='pValue', file=TRUE) @

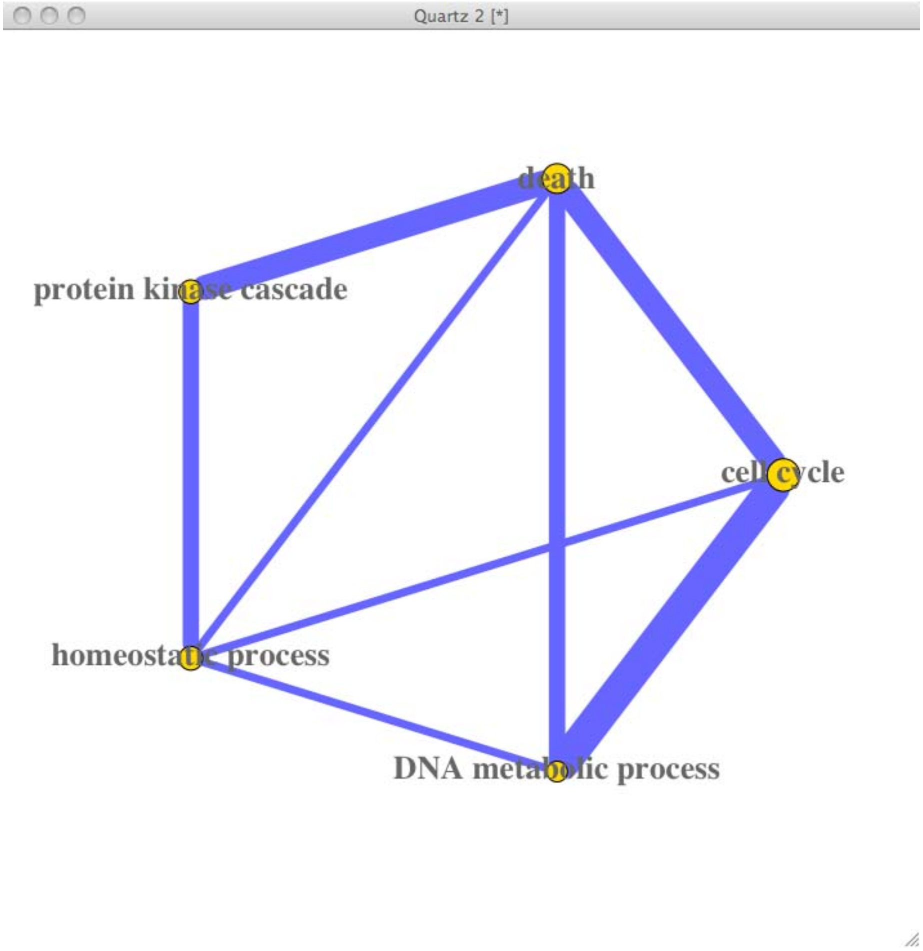
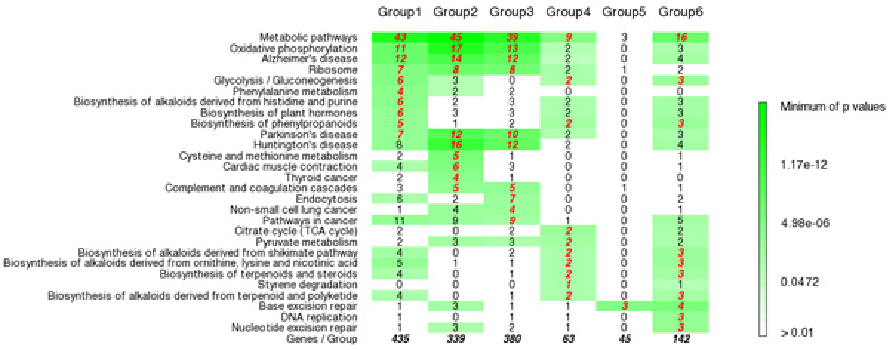


Figure 19: Screen shot of customized GO category linkage



	Group1	Group2	Group3	Group4	Group5	Group6
Metabolic pathways	43	45	39	9	3	16
Oxidative phosphorylation	11	17	13	2	0	3
Alzheimer's disease	12	14	12	2	0	4
Ribosome	7	8	8	2	1	2
Glycolysis / Gluconeogenesis	6	3	0	2	0	3
Phenylalanine metabolism	4	2	2	0	0	0
Biosynthesis of alkaloids derived from histidine and purine	6	2	3	2	0	3
Biosynthesis of plant hormones	6	3	3	2	0	3
Biosynthesis of phenylpropanoids	5	1	2	2	0	3
Parkinson's disease	7	12	10	2	0	3
Huntington's disease	8	16	12	2	0	4
Cysteine and methionine metabolism	2	5	1	0	0	1
Cardiac muscle contraction	4	6	3	0	0	1
Thyroid cancer	2	4	1	0	0	0
Complement and coagulation cascades	3	5	5	0	1	1
Endocytosis	6	2	7	0	0	2
Non-small cell lung cancer	1	4	4	0	0	1
Pathways in cancer	11	9	9	1	0	5
Citrate cycle (TCA cycle)	2	0	2	2	0	2
Pyruvate metabolism	2	3	3	2	0	2
Biosynthesis of alkaloids derived from shikimate pathway	4	0	2	2	0	3
Biosynthesis of alkaloids derived from ornithine, lysine and nicotinic acid	5	1	1	2	0	3
Biosynthesis of terpenoids and steroids	4	0	1	2	0	3
Styrene degradation	0	0	0	1	0	1
Biosynthesis of alkaloids derived from terpenoid and polyketide	4	0	1	2	0	3
Base excision repair	1	3	1	1	3	4
DNA replication	1	0	1	1	0	3
Nucleotide excision repair	1	3	2	1	0	3
Genes / Group	435	339	380	63	45	142

Figure 20: Screen shot of multigroup genes KEGG analysis

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.

¡¡multigroup gene analysis, echo=T, eval=T¿¿= load multigroup genes sample data data(sampleGroupsData) Build a GeneAnswers List gAKEGGL ¡- lapply(sampleGroupsData, geneAnswersBuilder, 'org.Hs.eg.db', categoryType='KEGG', pvalueT=0.01, verbose=FALSE) Output integrated text table output¡- getConceptTable(gAKEGGL, items='geneNum') @

¡¡multigroup genes KEGG analysis, echo=F, eval=F¿¿= drawTable(output[[1]], matrixOfHeatmap=output[[2]], mar=c(2,28,3,2), clusterTable=NULL) @

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 moues 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.

¡¡homogene conversation, echo=T, eval=T¿¿= load mouse example data data('mouseExpr') data('mouseGeneInput') mouseExpr[1:10,] mouseGeneInput[1:10,] only keep first one for one to more mapping pickHomo ¡- function(element, inputV) return(names(inputV[inputV == element])[1]) mapping geneInput to

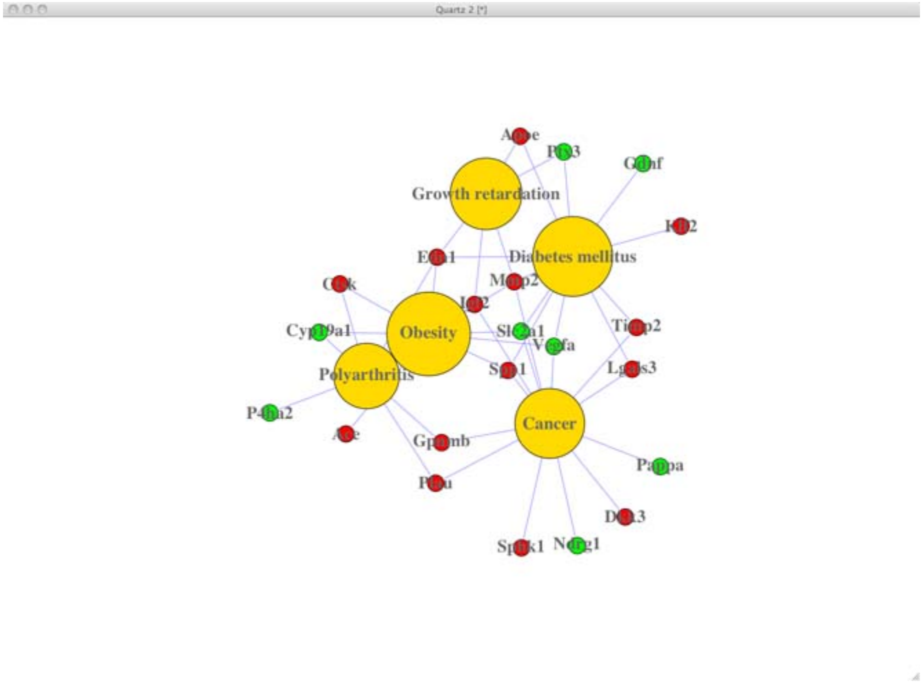


Figure 21: Screen shot of homogene DOLite-gene network

homo entrez IDs. homoLL ¡- getHomoGeneIDs(mouseGeneInput[,1], species='mouse', speciesL='human', mappingMethod='direct') newGeneInput ¡- mouseGeneInput[mouseGeneInput[,1] dim(mouseGeneInput) dim(newGeneInput) newGeneInput[,1] ¡- homoLL[newGeneInput[,1]] mapping geneExpr to homo entrez IDs. homoLLExpr ¡- getHomoGeneIDs(as.character(mouseExpr[,1]), species='mouse', speciesL='human', mappingMethod='direct') newExpr ¡- mouseExpr[as.character(mouseExpr[,1]) newExpr[,1] ¡- homoLLExpr[as.character(newExpr[,1])] dim(mouseExpr) dim(newExpr) build a GeneAnswers instance based on mapped data v ¡- geneAnswersBuilder(newGeneInput, 'org.Hs.eg.db', categoryType='DOLITE', testType='hyperG', pvalueT=0.1, FDR.correct=TRUE, geneExpressionProfile=newExpr) make the GeneAnswers instance readable, only map DOLite IDs to terms vv ¡- geneAnswersReadable(v, geneSymbol=F) getAnnLib(vv) mapping back to mouse genes uu ¡- geneAnswersHomoMapping(vv, species='human', speciesL='mouse', mappingMethod='direct') getAnnLib(uu) make mapped genes readable, DOLite terms are not mapped u ¡- geneAnswersReadable(uu, catTerm=FALSE) sort new GeneAnswers instance u1 ¡- geneAnswersSort(u, sortBy='pvalue') @

¡¡plot concept-gene network, echo=T, eval=F¿¿= plot concept-gene network geneAnswersConceptNet(u, colorValueColumn='foldChange', centroidSize='pvalue', output='fixed') @

¡¡generate homogene DOLite-gene cross tabulation, echo=T, eval=F¿¿= plot homogene DOLite-gene cross tabulation geneAnswersHeatmap(u1) @

¡¡homogene conversation, echo=T, eval=T¿¿= output top information topDOLITEGenes(u, geneSymbol=FALSE, catTerm=FALSE, orderby='pvalue', top=6, topGenes='ALL', genesOrderBy='pValue', file=TRUE) @

¡¡fig=true, width=8, height=8, quiet=T, echo=F, eval=T¿¿= plot homogene DOLite-gene cross tabulation geneAnswersHeatmap(u1) @

Figure 22: homogene DOLite-gene cross tabulation

6 Session Info

¡¡sessionInfo, results=tex, print=TRUE¿¿= toLatex(sessionInfo()) @

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',