GeneAnswers, Integrated Interpretation of Genes

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

October 28, 2009

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

Contents

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* 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=
```

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 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.

```
> library(GeneAnswers)
```

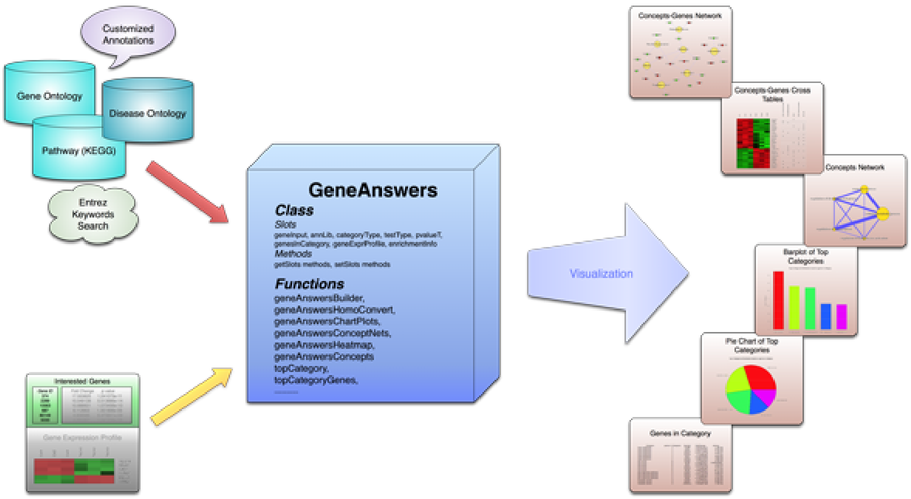

Figure 1: Flow chart of GeneAnswers

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='
```

```
[1] "geneInput has built in ..."
[1] "annLib and categoryType have built in ..."
[1] "genesInCategory has built in ..."
[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 da
> y <- geneAnswersBuilder(humanGeneInput, 'org.Hs.eg.db', categoryType='KEGG', testType='h

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

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

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

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

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

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

[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] "Converting geneInput ..."
[1] "Converting genesInCategory ..."
[1] "Converting enrichmentInfo rownames ..."
[1] "Converting 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] "Converting geneInput ..."
[1] "Converting genesInCategory ..."
[1] "Converting 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')
```

```
> ## plot interactive concept-gene network
> geneAnswersConceptNet(xx, colorValueColumn='foldChange', centroidSize='pvalue', output='
```

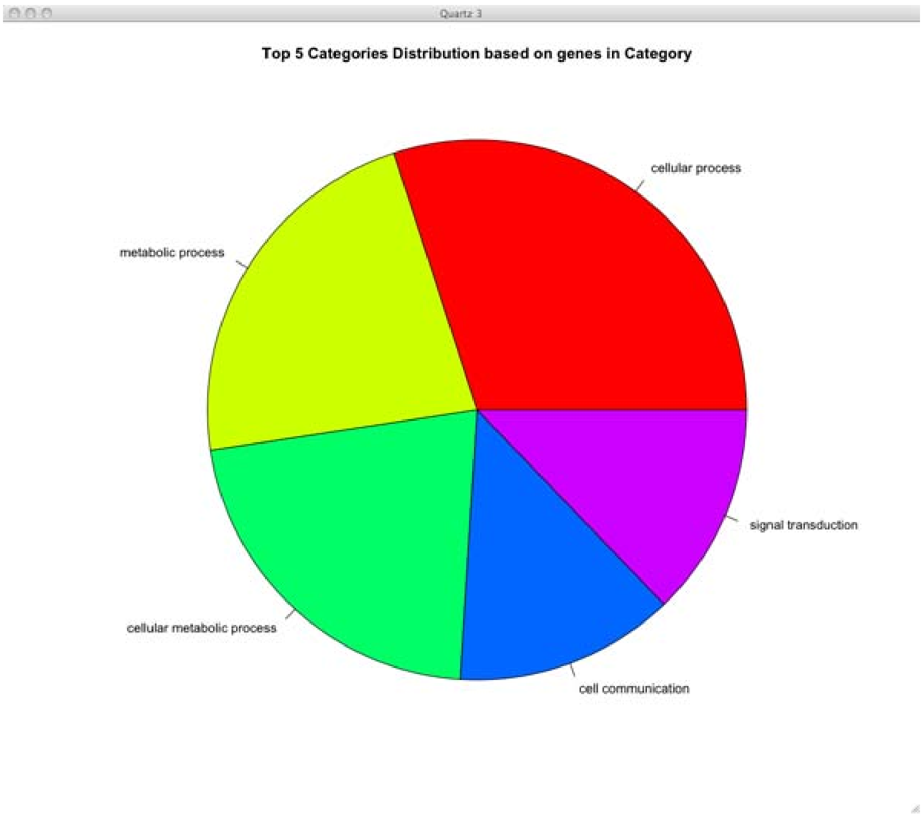

Figure 2: Screen shot of pie chart of top categories

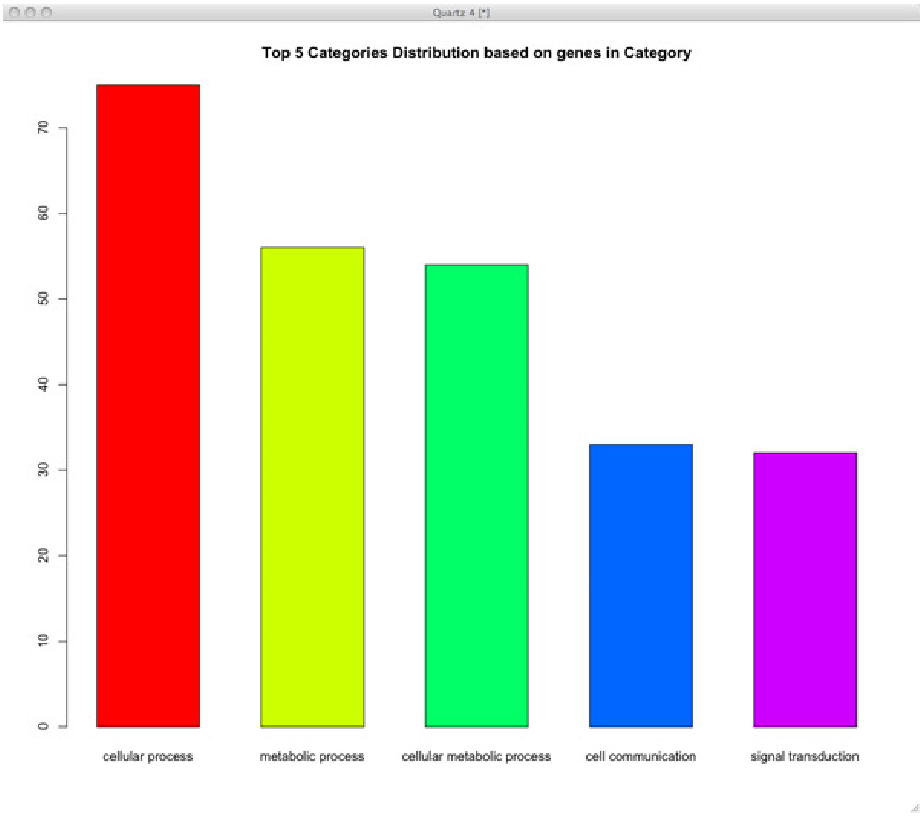

Figure 3: Screen shot of barplot of top categories

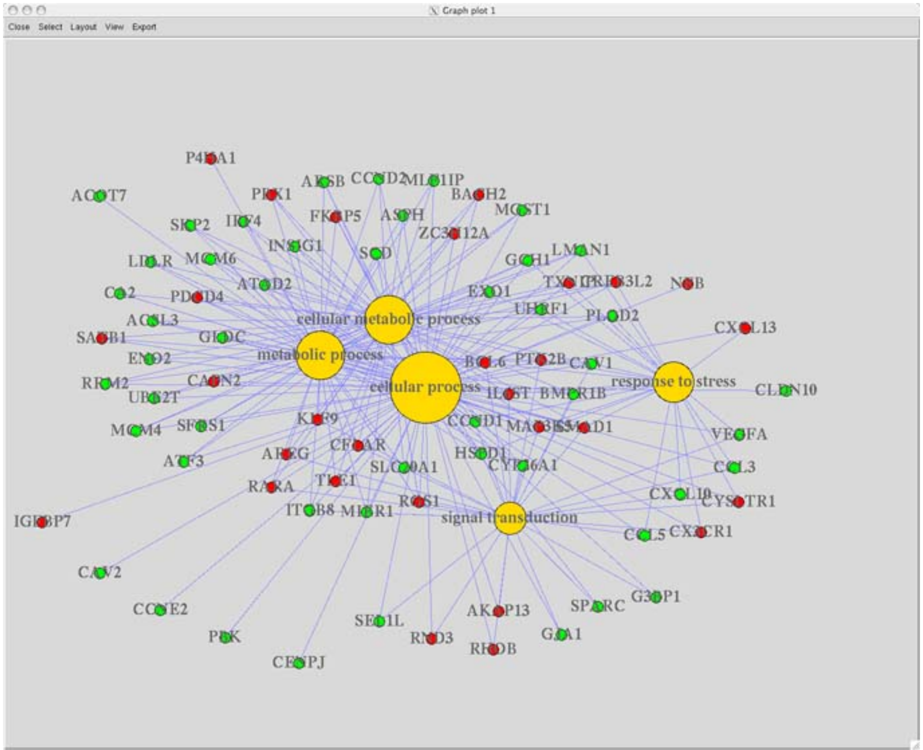

Figure 4: Screen shot of concept-gene network

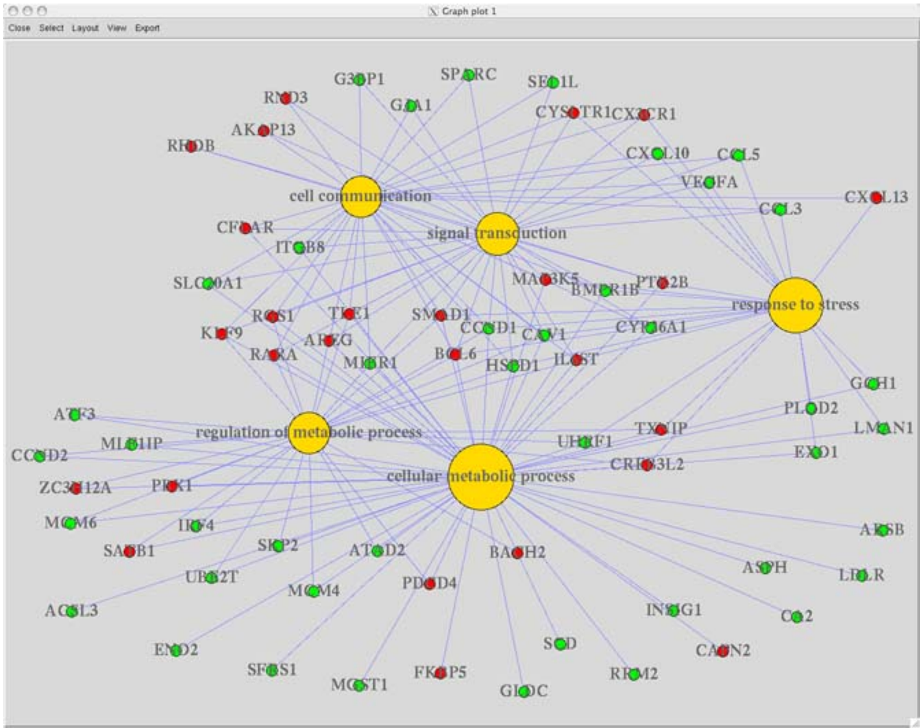

Figure 5: Screen shot of concept-gene network for top 2 GO level nodes removal

```
> ## plot Go-concept network for 2 level nodes removal
> geneAnswersConceptNet(ww, 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)
```

```
> 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

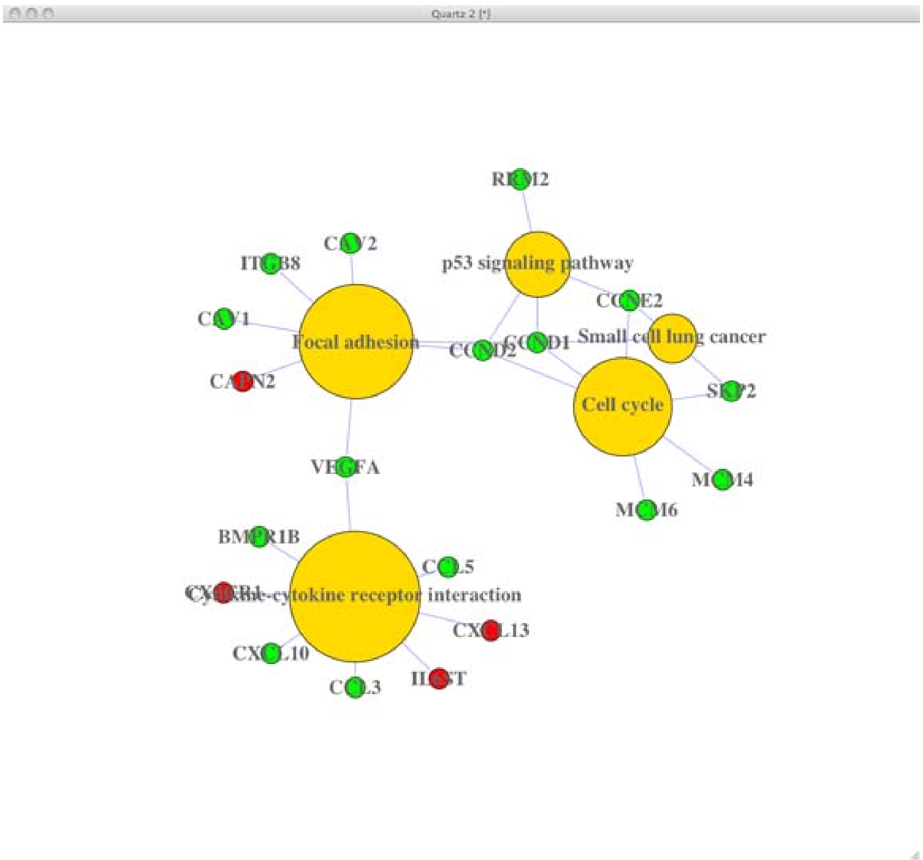

Figure 6: Screen shot of KEGG-gene network

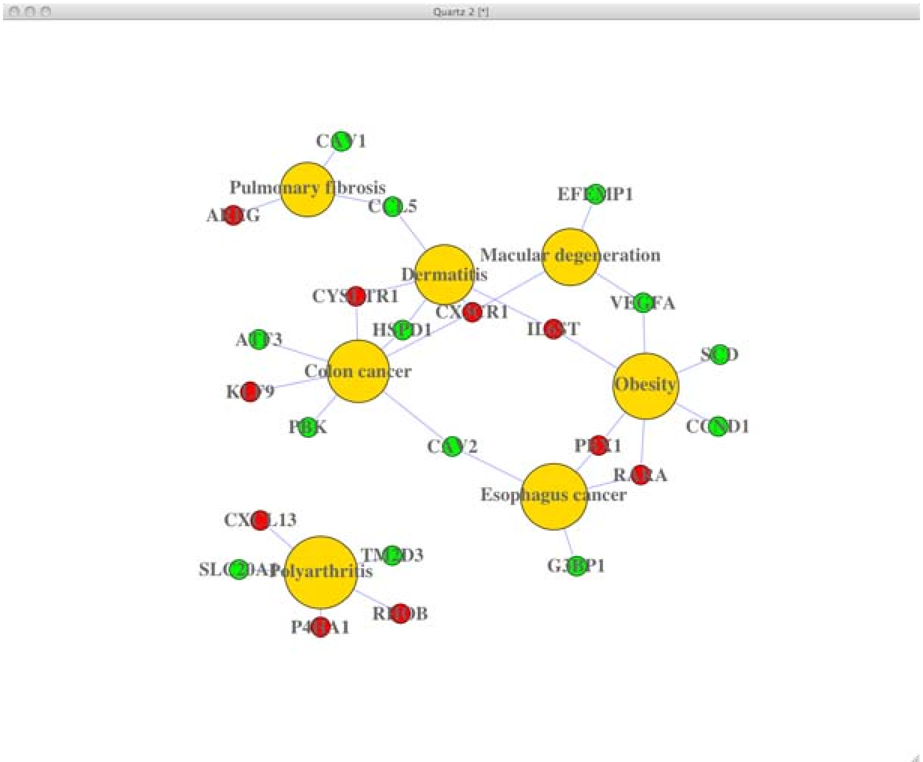

Figure 7: Screen shot of DOLite-gene network

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

```
initial  value 2.060891
iter   5 value 0.563404
iter  10 value 0.179183
iter  15 value 0.095165
iter  20 value 0.062150
iter  25 value 0.046441
iter  30 value 0.037490
iter  35 value 0.021600
iter  40 value 0.016805
iter  45 value 0.013431
iter  50 value 0.010041
final  value 0.010041
stopped after 50 iterations
initial  value 3.760019
iter   5 value 0.075511
final  value 0.005884
converged
```

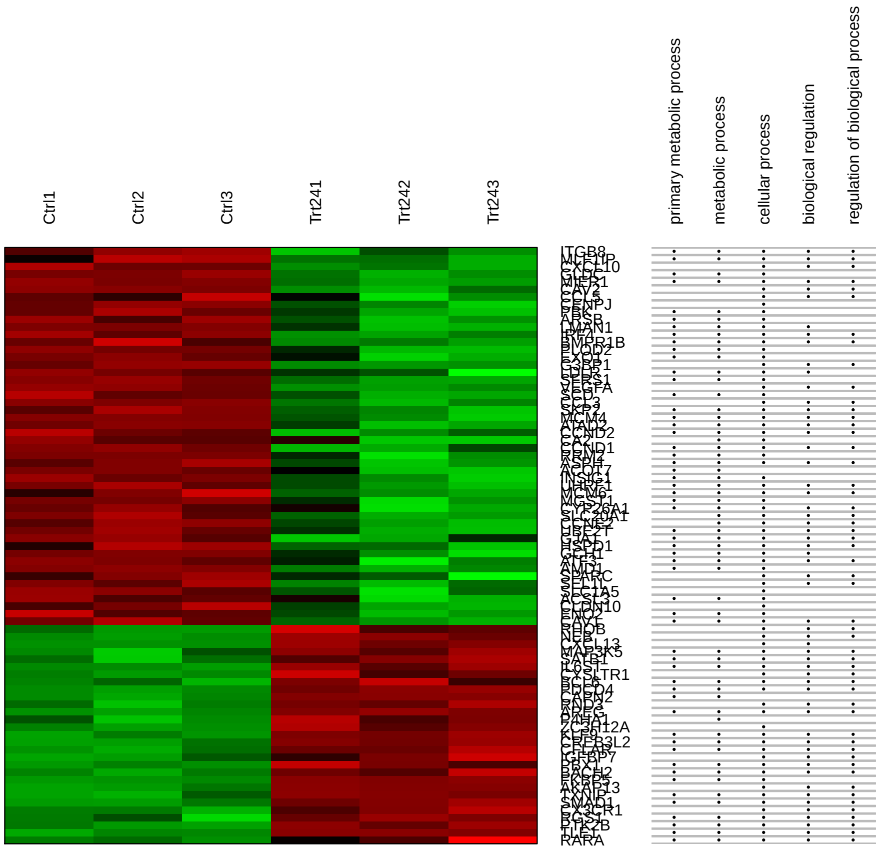

Figure 8: GO-gene cross tabulation

```
> 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 1.133016
iter  10 value 0.216047
iter  15 value 0.041517
iter  20 value 0.011044
iter  20 value 0.006056
iter  20 value 0.005719
final  value 0.005719
converged
```

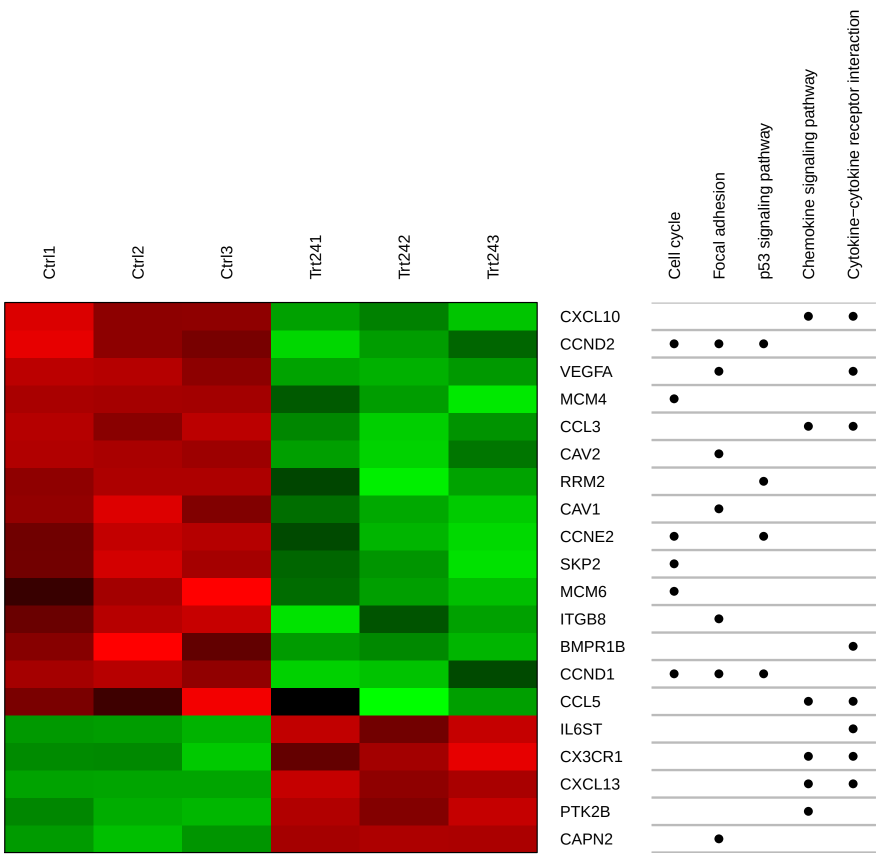

Figure 9: KEGG-gene cross tabulation

```
initial  value 1.962151
iter   5 value 0.463019
iter  10 value 0.104994
iter  15 value 0.083234
iter  20 value 0.051796
iter  25 value 0.024339
iter  30 value 0.016949
iter  35 value 0.011347
final  value 0.008069
converged
initial  value 11.178377
iter   5 value 1.353466
iter  10 value 0.417364
iter  15 value 0.058407
final  value 0.004365
converged
```

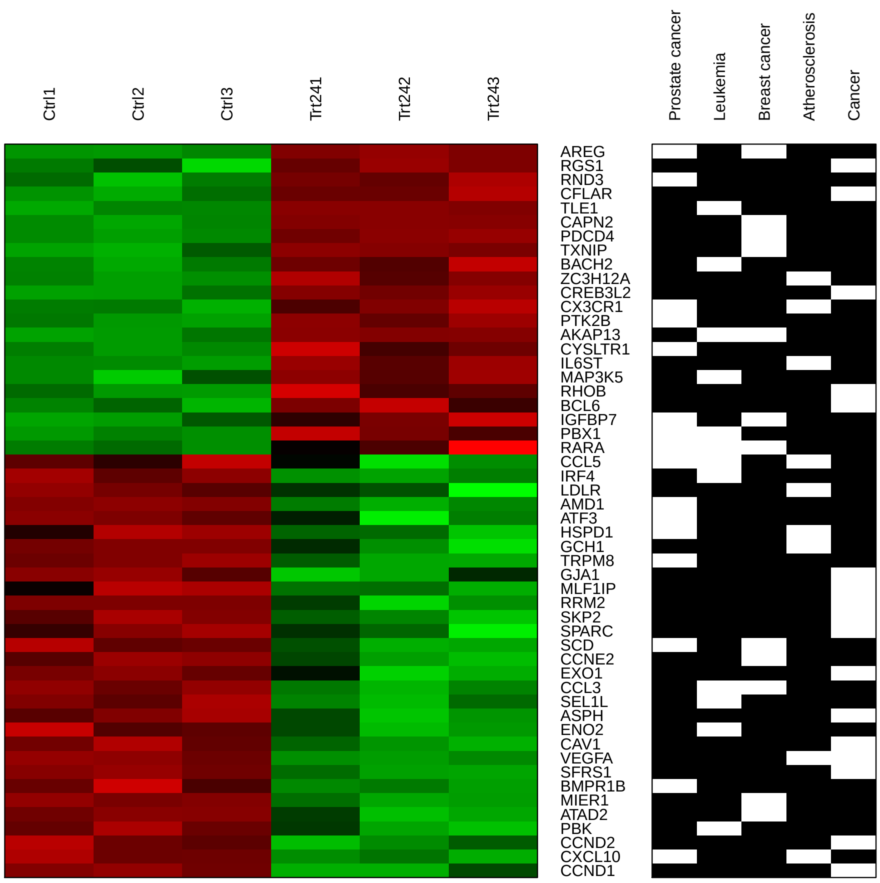

Figure 10: DOLite-gene cross tabulation

```
[1] "Some specified categories might not be statistical significant! Only show significant
initial  value 1.951039
iter   5 value 0.303949
iter  10 value 0.149066
iter  15 value 0.110179
iter  20 value 0.073996
iter  25 value 0.058564
iter  30 value 0.052496
iter  35 value 0.048771
iter  40 value 0.029830
iter  45 value 0.019509
iter  50 value 0.010840
final  value 0.010840
stopped after 50 iterations
initial  value 0.000000
final  value 0.000000
converged
```

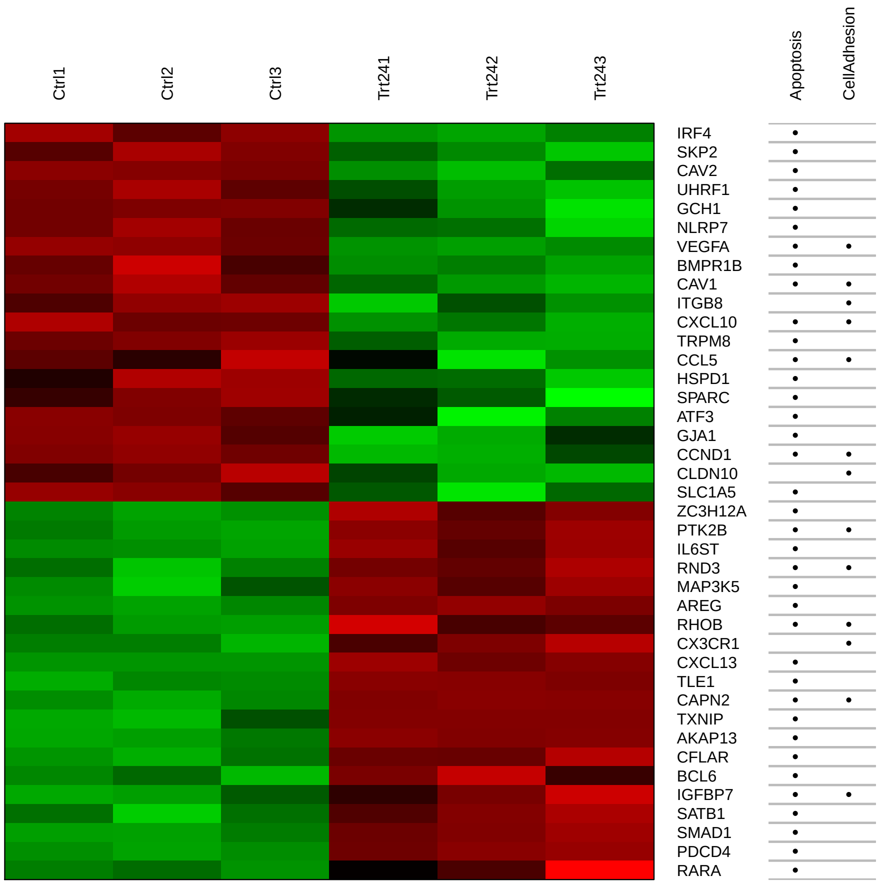

Figure 11: Customized GO-gene cross tabulation

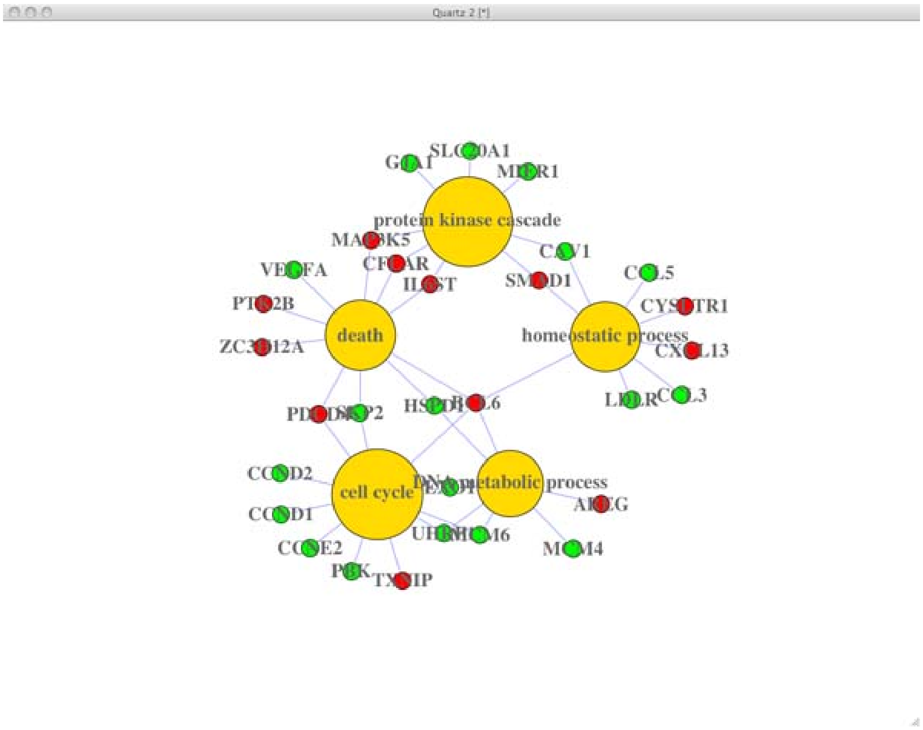

Figure 12: Screen shot of customized GO-gene network

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')
```

```
initial  value 2.112746
iter   5 value 0.414001
iter  10 value 0.208048
iter  15 value 0.156443
iter  20 value 0.091677
iter  25 value 0.056662
iter  30 value 0.047910
iter  35 value 0.035337
iter  40 value 0.017015
final  value 0.008870
converged
initial  value 29.195285
iter   5 value 23.647405
iter   5 value 23.633610
iter   5 value 23.633610
final  value 23.633610
converged
```

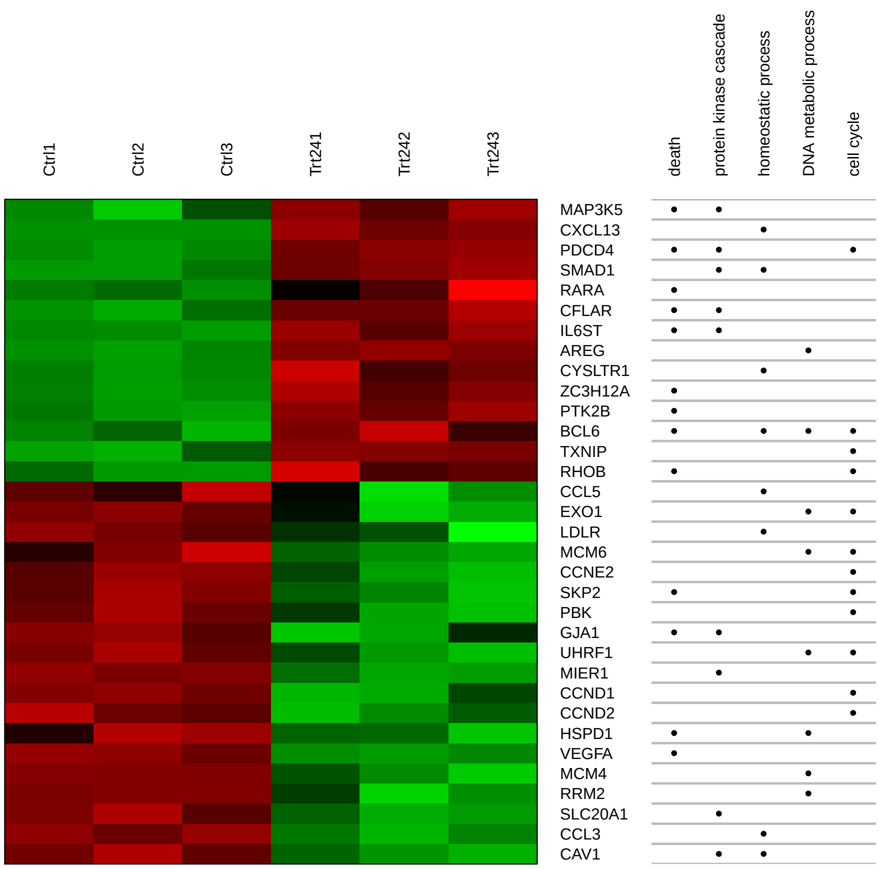

Figure 13: Customized concept-gene cross tabulation

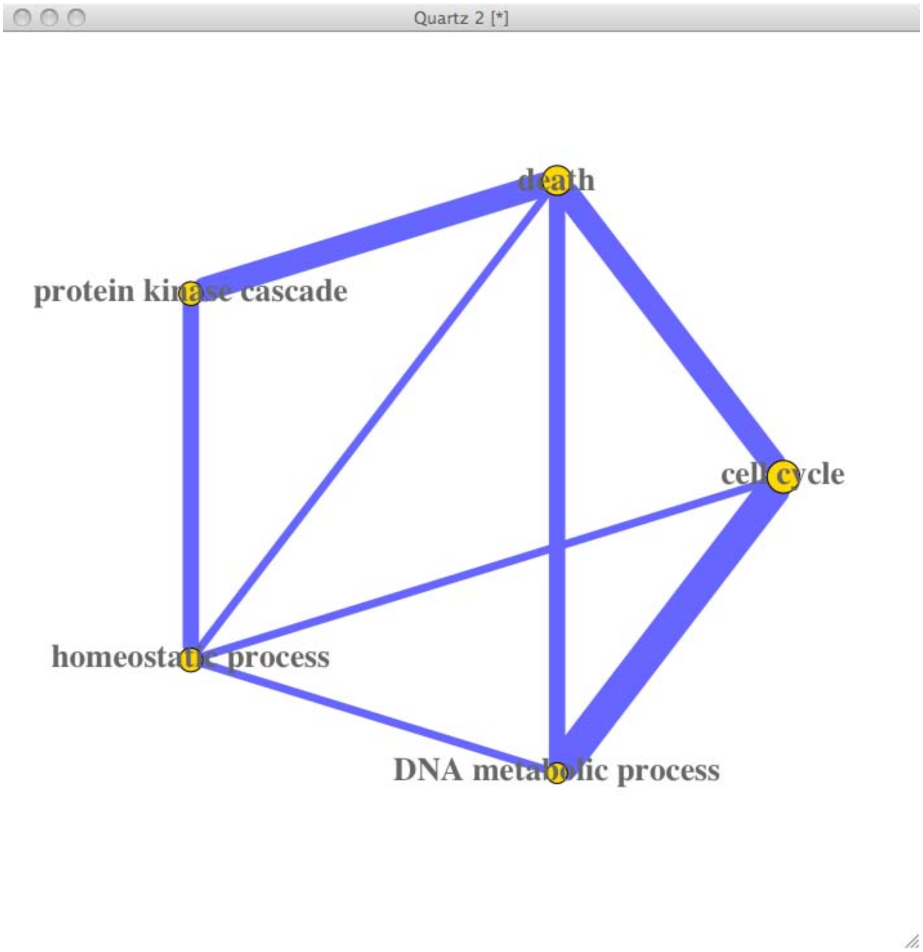

Figure 14: Screen shot of customized GO category linkage

```
[1] "******** cellular process   p value : 7.33663213571693e-34 ********"
      Symbol foldChange       pValue
2181   ACSL3  -2.049592 5.282896e-04
11214 AKAP13   2.062521 2.373847e-08
262     AMD1  -2.538954 1.216657e-09
374     AREG  17.553825 1.241073e-11
411     ARSB  -2.122207 3.199329e-06
[1] "******** biological regulation   p value : 2.59960788724242e-25 ********"
      Symbol foldChange       pValue
11214 AKAP13   2.062521 2.373847e-08
374     AREG  17.553825 1.241073e-11
444     ASPH  -2.337642 5.717355e-06
29028  ATAD2  -2.256887 6.002904e-06
467     ATF3  -2.014800 5.116121e-05
[1] "******** metabolic process   p value : 8.49173382581621e-25 ********"
      Symbol foldChange       pValue
11332  ACOT7  -2.022097 1.604253e-04
2181   ACSL3  -2.049592 5.282896e-04
262     AMD1  -2.538954 1.216657e-09
374     AREG  17.553825 1.241073e-11
411     ARSB  -2.122207 3.199329e-06
[1] "******** primary metabolic process   p value : 3.51338773490462e-23 ********"
      Symbol foldChange       pValue
11332  ACOT7  -2.022097 1.604253e-04
2181   ACSL3  -2.049592 5.282896e-04
262     AMD1  -2.538954 1.216657e-09
374     AREG  17.553825 1.241073e-11
411     ARSB  -2.122207 3.199329e-06
[1] "******** regulation of biological process   p value : 1.49775054208275e-22 ********"
      Symbol foldChange       pValue
11214 AKAP13   2.062521 2.373847e-08
374     AREG  17.553825 1.241073e-11
444     ASPH  -2.337642 5.717355e-06
29028  ATAD2  -2.256887 6.002904e-06
467     ATF3  -2.014800 5.116121e-05

> ## print top KEGG categories sorted by gene numbers and sort genes by fold changes
> topPATHGenes(y, orderby='geneNum', top=4, topGenes=8, genesOrderBy='foldChange')

[1] "******** Cytokine-cytokine receptor interaction   genes in Category : 8 ********"
      Symbol foldChange       pValue
6348    CCL3  -4.031781 1.147014e-07
7422   VEGFA  -2.391115 7.147830e-09
3627  CXCL10  -2.376811 1.315856e-07
658   BMPR1B  -2.161504 1.308048e-04
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] "******** Metabolic pathways   genes in Category : 8 ********"
```

```
     Symbol foldChange       pValue
262    AMD1  -2.538954 1.216657e-09
2731   GLDC  -2.454289 1.695089e-07
2026   ENO2  -2.207372 7.396045e-06
2643   GCH1  -2.151831 1.066956e-05
411    ARSB  -2.122207 3.199329e-06
6241   RRM2  -2.102265 2.339306e-06
2181  ACSL3  -2.049592 5.282896e-04
5033  P4HA1   4.265287 1.341238e-06
[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

> ## print and save top 10 DOLites information
> topDOLiteGenes(z, orderby='pvalue', top=5, topGenes='ALL', genesOrderBy='pValue', file=T

[1] "******** Prostate cancer   p value : 1.45588138562694e-17 ********"
     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] "******** Cancer   p value : 4.34960215018646e-14 ********"
       Symbol foldChange       pValue
7422    VEGFA  -2.391115 7.147830e-09
64764 CREB3L2   2.909868 1.162846e-08
8837    CFLAR   2.850424 2.960108e-08
6426    SFRS1  -2.088303 4.706379e-08
595     CCND1  -2.257123 5.932920e-07
[1] "******** Leukemia   p value : 1.01712450654578e-12 ********"
      Symbol foldChange       pValue
7088    TLE1   5.427252 1.993568e-11
11214 AKAP13   2.062521 2.373847e-08
3662    IRF4  -2.573029 2.583465e-08
6400   SEL1L  -2.722062 9.822205e-08
```

```
5087    PBX1   2.552036 1.120693e-07
[1] "******** Breast cancer   p value : 4.28183925721002e-11 ********"
      Symbol foldChange      pValue
374     AREG  17.553825 1.241073e-11
10628  TXNIP   3.899457 4.296784e-09
824    CAPN2   2.568772 1.890638e-08
11214 AKAP13   2.062521 2.373847e-08
3490  IGFBP7   3.787547 2.688423e-08
[1] "******** Atherosclerosis   p value : 2.23421982569154e-10 ********"
       Symbol foldChange      pValue
80149 ZC3H12A   4.939585 6.473841e-09
7422    VEGFA  -2.391115 7.147830e-09
3572    IL6ST   2.213384 8.018338e-08
3627   CXCL10  -2.376811 1.315856e-07
1524   CX3CR1   2.127494 4.412558e-06
[1] "File topCategoryGenes.txt is successfully generated!"
```

5.3 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.

```
>  ## 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] 66  3

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

[1] "Warning: homogenes 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] 66  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] "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] "Converting genesInCategory ..."
[1] "Converting 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 ..."

> getAnnLib(uu)

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

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

[1] "Converting geneInput ..."
[1] "Converting genesInCategory ..."
[1] "Converting 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] "******** Obesity   p value : 3.61357450643102e-12 ********"
      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] "******** Diabetes mellitus   p value : 5.18742071898125e-12 ********"
      Symbol foldChange       pValue
16598   Klf2   2.913280 1.863840e-10
```

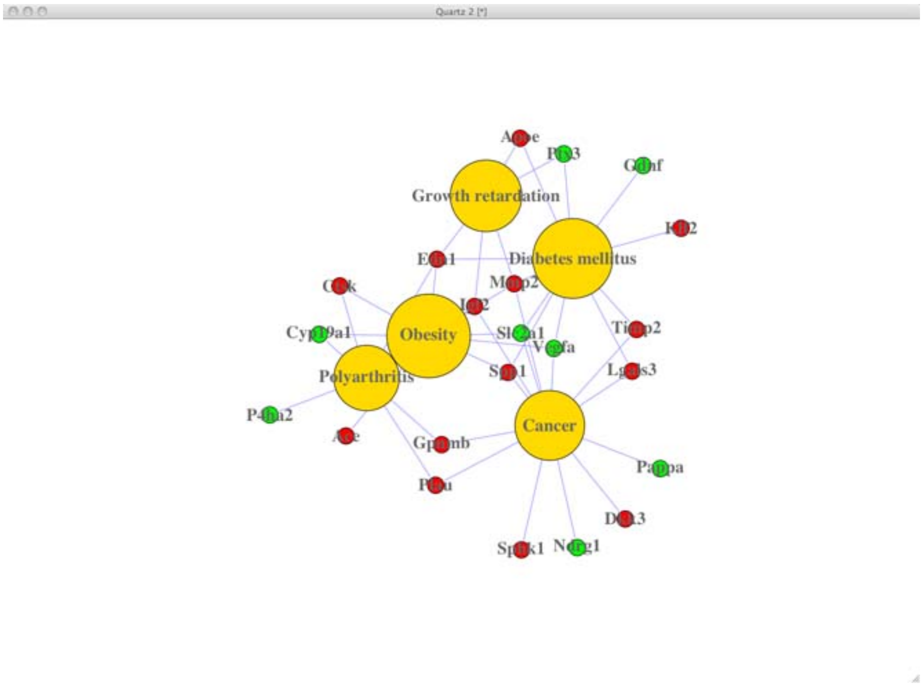

Figure 15: Screen shot of homogene DOLite-gene network

```
20525 Slc2a1  -2.590159 6.290580e-09
13614   Edn1   2.504625 1.536084e-08
22339  Vegfa  -2.535157 3.651889e-08
21858  Timp2   2.195299 4.638384e-08
[1] "******** Cancer    p value : 4.48947063154371e-11 ********"
      Symbol foldChange       pValue
20525 Slc2a1  -2.590159 6.290580e-09
93695  Gpnmb   4.869452 1.864011e-08
22339  Vegfa  -2.535157 3.651889e-08
21858  Timp2   2.195299 4.638384e-08
17390   Mmp2   2.932919 5.771626e-08
[1] "******** Growth retardation    p value : 4.38812418292752e-10 ********"
      Symbol foldChange       pValue
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] "******** Polyarthritis    p value : 1.63827115569347e-09 ********"
      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
```

```
initial  value 0.495569
iter   5 value 0.151207
iter  10 value 0.127116
iter  15 value 0.097958
iter  20 value 0.066870
iter  25 value 0.055071
iter  30 value 0.011353
iter  30 value 0.009738
iter  30 value 0.009291
final  value 0.009291
converged
initial  value 17.277129
iter   5 value 12.546623
final  value 11.897194
converged
```

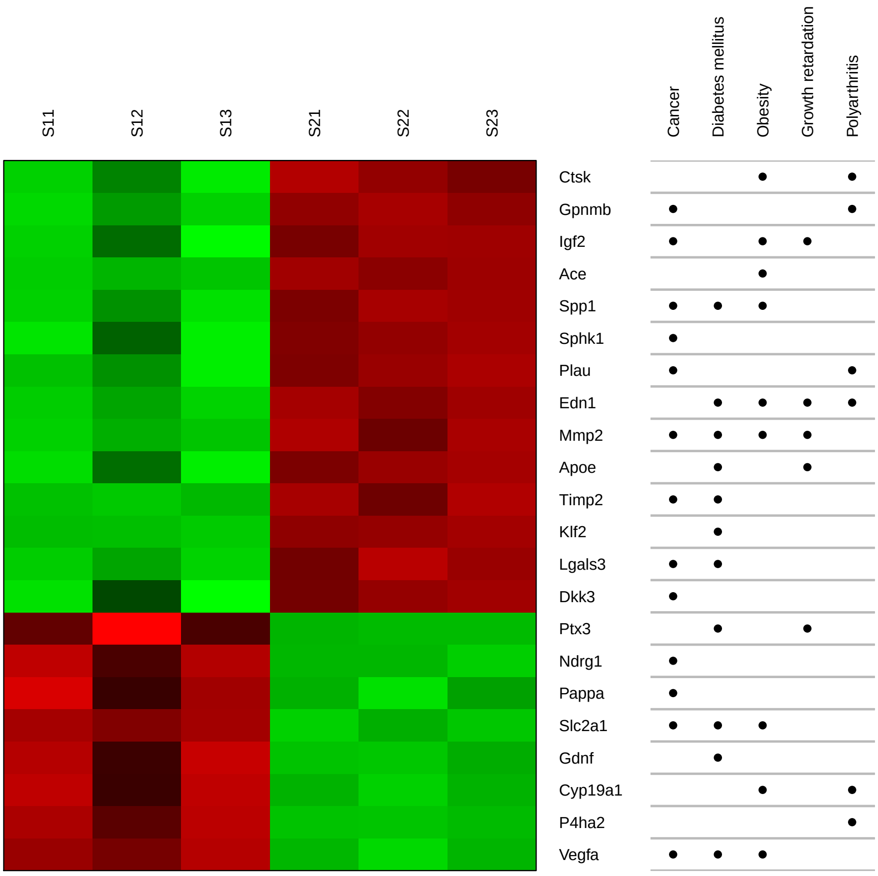

Figure 16: homogene DOLite-gene cross tabulation

```
[1] "******** Rheumatoid arthritis   p value : 5.68914871872988e-09 ********"
      Symbol foldChange       pValue
13614   Edn1   2.504625 1.536084e-08
14955    H19   2.904115 2.570580e-08
17390   Mmp2   2.932919 5.771626e-08
20750   Spp1   4.573777 1.224957e-07
18792   Plau   2.456354 4.029683e-07
[1] "File topCategoryGenes.txt is successfully generated!"
```

6 Session Info

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

- R version 2.10.0 (2009-10-26), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=en_US.UTF-8`, `LC_MONETARY=C`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, graphics, grDevices, methods, stats, tools, utils
- Other packages: annotate 1.24.0, AnnotationDbi 1.8.0, Biobase 2.6.0, bitops 1.0-4.1, DBI 0.2-4, GeneAnswers 1.2.0, GO.db 2.3.5, Heatplus 1.16.0, igraph 0.5.2-2, KEGG.db 2.3.5, MASS 7.3-3, org.Hs.eg.db 2.3.6, org.Mm.eg.db 2.3.6, RColorBrewer 1.0-2, RCurl 1.2-1, RSQLite 0.7-3, XML 2.6-0
- Loaded via a namespace (and not attached): xtable 1.5-5

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',