GeneAnswers, Integrated Interpretation of Genes

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

October 30, 2018

†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
‡ wkibbe (at) northwestern.edu
§ s-lin2 (at) northwestern.edu

* For GeneAnswers:

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.

```r
if (!requireNamespace("BiocManager", quietly=TRUE))
  install.packages("BiocManager")
BiocManager::install("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:

```r
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.

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)
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.

```r
> 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)
```

```
[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 created!"
```

```r
> class(x)
```

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

```r
> ## 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)
```

```
[1] "GeneAnswers instance has been successfully created!"
```

```r
> ## 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)
```

```
[1] "GeneAnswers instance has been successfully created!"
```

```r
> w <- geneAnswersBuilder(humanGeneInput, 'org.Hs.eg.db', categoryType='GO.BP', testType='hyperG', pvalueT=0.1, FDR.correction=TRUE, geneExpressionProfile=humanExpr, level=2, verbose=FALSE)
```

```
[1] "GeneAnswers instance has been successfully created!"
```
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.

```r
> # 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 defined" in categoryType slot.

### 5.2 Visualization

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.

```r
> # 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 ..."
Figure 2: Screen shot of pie chart of top categories

```r
> 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
> 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
> geneAnswersConceptNet(xx, colorValueColumn='foldChange', centroidSize='pvalue', output='
Figure 3: Screen shot of barplot of top categories
The top 5 categories might not be very specific. Users might get a tree view to see relative category terms by calling function geneAnswersConceptRelation 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 connections.

```r
> ## plot interactive go structure network
> geneAnswersConceptRelation(x, direction='both', netMode='connection', catTerm=TRUE, catID=TRUE)
```

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.

```r
> ## plot interactive concept-gene network
> geneAnswersConceptNet(x, color='foldChange', 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.

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

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
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

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, 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 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' function by parameter 'colorMap'. Details are available in 'buildNet' manual page. If users want to have an 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=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.
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
The following example is to show top 5 GO-gene network for the first 2 level GO nodes removal.

```r
## 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.

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

> geneAnswersConceptNet(yyy, colorValueColumn='foldChange', centroidSize='geneNum', output='fixed')
> geneAnswersConceptNet(zzz, colorValueColumn='foldChange', centroidSize='pvalue', output='fixed', showCats=c(10:16))
```

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.

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

> geneAnswersHeatmap(yyy)
```
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 1.949469
iter 5 value 0.870924
iter 10 value 0.248809
iter 15 value 0.055166
iter 20 value 0.023174
iter 25 value 0.014625
iter 30 value 0.014625
iter 35 value 0.014625
iter 40 value 0.014625
iter 45 value 0.014625
iter 50 value 0.014625
final value 0.014625
stopped after 50 iterations

initial value 9.129670
iter 5 value 0.400034
iter 10 value 0.021154
final value 0.003471
converged

Figure 14: 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 1.114859
iter 10 value 0.043826
final value 0.005340
converged

Figure 15: KEGG-gene cross tabulation
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: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
> geneAnswersConceptNet(x, colorValueColumn='foldChange', centroidSize='pvalue', output='fixed', showCats=GOBPIDs, catTerm=TRUE, geneSymbol=TRUE)
> 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 lipid::GO:0033993  p value : 2.293e-17 ********"
   GeneID foldChange    pValue
11214  AKAP13  2.062521 2.373847e-08
374   AREG  17.553825 1.241073e-11
  760  CA2   -2.256972 8.992257e-04
  857  CAV1   -2.122246 3.842276e-06
 6348  CCL3  -4.031781 1.147014e-07
[1] "******** response to organic substance::GO:0010033  p value : 3.019e-13 ********"
   GeneID foldChange    pValue
2181   ACSL3  -2.049592 5.282896e-04
11214  AKAP13  2.062521 2.373847e-08
  374   AREG  17.553825 1.241073e-11
  411   ARSB  -2.122207 3.199329e-06
   467   ATF3  -2.014800 5.116121e-05
   GeneID foldChange    pValue
 374    AREG  17.553825 1.241073e-11
  760    CA2   -2.256972 8.992257e-04
  824   CAPN2  2.568772 1.890638e-08
  857    CAV1  -2.122246 3.842276e-06
  858    CAV2  -2.643498 2.625770e-08
[1] "******** response to external stimulus::GO:0009605  p value : 1.976e-11 ********"
   GeneID foldChange    pValue
```
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
Figure 17: Screen shot of customized GO-gene network

<table>
<thead>
<tr>
<th>GeneID</th>
<th>Gene</th>
<th>foldChange</th>
<th>pValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>2181</td>
<td>ACSL3</td>
<td>-2.049592</td>
<td>5.282896e-04</td>
</tr>
<tr>
<td>411</td>
<td>ARSB</td>
<td>-2.122207</td>
<td>3.199329e-06</td>
</tr>
<tr>
<td>467</td>
<td>ATF3</td>
<td>-2.014800</td>
<td>5.116121e-05</td>
</tr>
<tr>
<td>604</td>
<td>BCL6</td>
<td>2.380046</td>
<td>2.436277e-04</td>
</tr>
<tr>
<td>658</td>
<td>BMPR1B</td>
<td>-2.161504</td>
<td>1.308048e-04</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>GeneID</th>
<th>Gene</th>
<th>foldChange</th>
<th>pValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>6348</td>
<td>CCL3</td>
<td>-4.031781</td>
<td>1.147014e-07</td>
</tr>
<tr>
<td>7422</td>
<td>VEGFA</td>
<td>-2.391115</td>
<td>7.147830e-09</td>
</tr>
<tr>
<td>3627</td>
<td>CXCL10</td>
<td>-2.376811</td>
<td>1.315856e-07</td>
</tr>
<tr>
<td>658</td>
<td>BMPR1B</td>
<td>-2.161504</td>
<td>1.308048e-04</td>
</tr>
<tr>
<td>6352</td>
<td>CCL5</td>
<td>-2.154827</td>
<td>1.092361e-03</td>
</tr>
<tr>
<td>1524</td>
<td>CX3CR1</td>
<td>2.127494</td>
<td>4.412558e-06</td>
</tr>
</tbody>
</table>

> ## 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::04060  genes in Category : 8 ********

<table>
<thead>
<tr>
<th>GeneID</th>
<th>Gene</th>
<th>foldChange</th>
<th>pValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>6348</td>
<td>CCL3</td>
<td>-4.031781</td>
<td>1.147014e-07</td>
</tr>
<tr>
<td>7422</td>
<td>VEGFA</td>
<td>-2.391115</td>
<td>7.147830e-09</td>
</tr>
<tr>
<td>3627</td>
<td>CXCL10</td>
<td>-2.376811</td>
<td>1.315856e-07</td>
</tr>
<tr>
<td>658</td>
<td>BMPR1B</td>
<td>-2.161504</td>
<td>1.308048e-04</td>
</tr>
<tr>
<td>6352</td>
<td>CCL5</td>
<td>-2.154827</td>
<td>1.092361e-03</td>
</tr>
<tr>
<td>1524</td>
<td>CX3CR1</td>
<td>2.127494</td>
<td>4.412558e-06</td>
</tr>
</tbody>
</table>
initial value 2.041651
iter 5 value 0.378648
iter 10 value 0.227176
iter 15 value 0.151545
iter 20 value 0.108771
iter 25 value 0.022936
iter 25 value 0.009794
iter 25 value 0.009586
final value 0.009586
converged

initial value 0.000000
final value 0.000000
converged

Figure 18: Customized concept-gene cross tabulation
Figure 19: Screen shot of customized GO category linkage
<table>
<thead>
<tr>
<th>GeneID</th>
<th>foldChange</th>
<th>pValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>3572</td>
<td>2.213384</td>
<td>8.018338e-08</td>
</tr>
<tr>
<td>10563</td>
<td>10.688601</td>
<td>1.073456e-10</td>
</tr>
</tbody>
</table>

[1] "******** Focal adhesion::04510 genes in Category : 7 ********"

GeneID foldChange  pValue
| 3696   | -3.097913  | 1.178489e-06|
| 858    | -2.643498  | 2.625770e-08|
| 7422   | -2.391115  | 7.147830e-09|
| 595    | -2.257123  | 5.932920e-07|
| 894    | -2.239806  | 1.208734e-06|
| 857    | -2.122246  | 3.842276e-06|
| 284    | 2.568772   | 1.890638e-08|

<table>
<thead>
<tr>
<th>GeneID</th>
<th>foldChange</th>
<th>pValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>9134</td>
<td>-2.379943</td>
<td>3.326011e-06</td>
</tr>
<tr>
<td>4175</td>
<td>-2.356668</td>
<td>2.080808e-04</td>
</tr>
<tr>
<td>6502</td>
<td>-2.276824</td>
<td>1.013445e-05</td>
</tr>
<tr>
<td>595</td>
<td>-2.257123</td>
<td>5.932920e-07</td>
</tr>
<tr>
<td>894</td>
<td>-2.239806</td>
<td>1.208734e-06</td>
</tr>
<tr>
<td>4173</td>
<td>-2.082429</td>
<td>8.413734e-06</td>
</tr>
</tbody>
</table>

[1] "******** Cell cycle::04110 genes in Category : 6 ********"

GeneID foldChange  pValue
| 9134   | -2.379943   | 3.326011e-06|
| 4175   | -2.356668   | 2.080808e-04|
| 6502   | -2.276824   | 1.013445e-05|
| 595    | -2.257123   | 5.932920e-07|
| 894    | -2.239806   | 1.208734e-06|
| 4173   | -2.082429   | 8.413734e-06|

[1] "******** Chemokine signaling pathway::04062 genes in Category : 6 ********"

GeneID foldChange  pValue
| 6348   | -4.031781   | 1.147014e-07|
| 3627   | -2.376811   | 1.315856e-07|
| 6352   | -2.154827   | 1.092361e-03|
| 2185   | 2.036789    | 1.970066e-07|
| 1524   | 2.127494    | 4.412558e-06|
| 10563  | 10.688601   | 1.073456e-10|

> ## print and save top 10 DOLites information
> topDOLITEGenes(z, orderby='pvalue', top=5, topGenes='ALL', genesOrderBy='pValue', file=TRUE)

[1] "******** Hyperlipidemia::DOLite:261 p value : 0.001982 ********"

GeneID foldChange  pValue
| 10628  | 3.899457    | 4.296784e-09|
| 6319   | -2.480571   | 6.976419e-07|
| 2181   | -2.049592   | 5.282896e-04|
| 6352   | -2.154827   | 1.092361e-03|

[1] "******** Prostate cancer::DOLite:447 p value : 0.003963 ********"

GeneID foldChange  pValue
| 374    | 17.553825   | 1.241073e-11|
| 262    | -2.538954   | 2.116657e-09|
| 3490   | 3.787547    | 2.688423e-08|
| 5087   | 2.552036    | 1.120693e-07|
| 3627   | -2.376811   | 1.315856e-07|

[1] "******** Alveolar bone loss::DOLite:43 p value : 0.008729 ********"

GeneID foldChange  pValue
| 6348   | -4.031781   | 1.147014e-07|
| 6678   | -2.289622   | 7.751533e-04|

[1] "******** Leukodystrophy NOS::DOLite:307 p value : 0.01148 ********"

GeneID foldChange  pValue
| 6348   | -4.031781   | 1.147014e-07|
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.

```r
> ##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

```r
output<- getConceptTable(gAKEGGL, items='geneNum')
```

Function groupReport can generate a html file including all information.

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.

```r
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 conversion though it is possible to miss some information.

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

  GeneID S11  S12  S13  S21  S22  S23
2 207500 10.378364 10.780340 12.51370 12.777766 12.727550
4  20210 10.417790 10.503403 12.38010 12.643765 12.453697
6  17105 10.599174 10.078450 12.47790 12.797566 12.508973

> mouseGeneInput[1:10,]

   Symbol foldChange pValue
93695    93695  4.869452  1.864011e-08
207500   20750  4.573777  1.224957e-07
16854    16854  4.274721  6.526113e-08
20210    20210  4.956676  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', mappingMethod='direct')
[1] "Warning: homogenes of some input genes can not be found and are removed!!!"

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

25 10
```
> 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='human', mappingMethod='direct')

[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), pickHomo, 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='hyperG', pvalueT=0.1, FDR.correct=TRUE, geneExpressionProfile=newExpr)

[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 created!"

> ## 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='direct')

[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] "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='fixed')

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

> ## output top information
> topDOLITEGenes(u, geneSymbol=FALSE, catTerm=FALSE, orderby='pvalue', top=6, topGenes='ALL', genesOrderBy='pValue', file=TRUE)

[1] "******* Growth retardation  p value : 1.86e-06 *******"

<table>
<thead>
<tr>
<th>Symbol</th>
<th>foldChange</th>
<th>pValue</th>
</tr>
</thead>
<tbody>
<tr>
<td>13614</td>
<td>Edn1</td>
<td>2.504625</td>
</tr>
<tr>
<td>17390</td>
<td>Mmp2</td>
<td>2.932919</td>
</tr>
<tr>
<td>11816</td>
<td>Apoe</td>
<td>2.465045</td>
</tr>
<tr>
<td>16002</td>
<td>Igf2</td>
<td>3.217968</td>
</tr>
</tbody>
</table>

27
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 22: homogene DOLite-gene cross tabulation
19288  Ptx3  -2.100947  4.364358e-05
[1] "******** Obesity  p value : 5.282e-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
  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] "******** Poliarthritis p value : 2.853e-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
  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 : 5.687e-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 3.5.1 Patched (2018-07-12 r74967), x86_64-pc-linux-gnu
- Locale: LC_CTYPE=en_US.UTF-8, LC_NUMERIC=C, LC_TIME=en_US.UTF-8, LC_COLLATE=C, LC_MONETARY=en_US.UTF-8, 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
- Running under: Ubuntu 16.04.5 LTS
- Matrix products: default
- BLAS: /home/biocbuild/bbs-3.8-bioc/R/lib/libRblas.so
- LAPACK: /home/biocbuild/bbs-3.8-bioc/R/lib/libRlapack.so
Acknowledgments

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

References


7 Base packages: base, datasets, grDevices, graphics, methods, parallel, stats, stats4, utils

Other packages: AnnotationDbi 1.44.0, Biobase 2.42.0, BiocGenerics 0.28.0, GO.db 3.7.0, GeneAnswers 2.24.0, Heatplus 2.28.0, IRanges 2.16.0, KEGG.db 3.2.3, MASS 7.3-51, RColorBrewer 1.1-2, RCurl 1.95-4.11, RSQLite 2.1.1, S4Vectors 0.20.0, XML 3.98-1.16, annotate 1.60.0, bitops 1.0-6, igraph 1.2.2, org.Hs.eg.db 3.7.0, org.Mm.eg.db 3.7.0

Loaded via a namespace (and not attached): DBI 1.0.0, RBGL 1.58.0, Rcpp 0.12.19, bit 1.1-14, bit64 0.9-7, blob 1.1.1, compiler 3.5.1, digest 0.6.18, downloader 0.4, graph 1.60.0, magrittr 1.5, memoise 1.1.0, pkgconfig 2.0.2, tools 3.5.1, xtable 1.8-3