FGNet
Functional Gene Networks
derived from biological enrichment analyses

Sara Aibar, Celia Fontanillo, Conrad Droste, and Javier De Las Rivas

Bioinformatics and Functional Genomics Group
Centro de Investigacion del Cancer (CiC-IBMCC, CSIC/USAL)
Salamanca - Spain

March 13, 2014

Version: 1.2

Contents

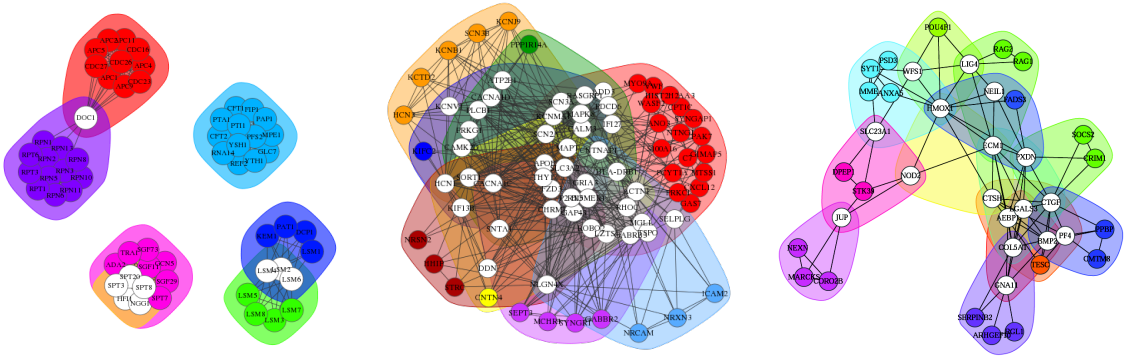
1 Introduction to *Functional Gene Networks*

Functional Gene Networks (FGNet) allows to analize the results from a clustered functional enrichment analysis and transform them into a gene network. In this way, FGNet provides a functional gene network, which is analyzed and presented as a graphical object allowing an overview of the links and overlapping betwen genes and biological functions. The package also includes an interface to perform the enrichment and clustering through *DAVID* and *GeneTerm Linker* tools.

Biological functional analysis
After obtaining a list of genes or proteins from an experiment or omic studies (i.e. microarrays, RNAseq, mass spectrometry, etc), next step is usually to perform a functional analysis of the genes to search for the biological functions or processes in which they are involved. In order to facilitate the analysis of large lists of genes, multiple functional enrichment tools have been developed. These tools search for the genes in biological databases (i.e. GO, Kegg, Interpro), and test whether any biological annotations are over-represented in the query gene list compared to what would be expected in the whole population. However, the raw output from a functional enrichment analysis often provides hundreds of terms, and it still requires a lot of time and attention to go through the whole list of genes and annotations. A way to simplify this task is grouping genes and terms which often appear together and create associated networks. *Functional Gene Networks* algorithm has been designed with this purpose. To do so, it uses the output of a previous functional enrichment analysis. The package has been implemented to support two specific tools:

1. Functional Annotation Clustering from DAVID, which measures relationships among annotation terms based on their co-association with genes within the query gene list [1, 2]. This type of clustering mostly results in groups of highly related terms, such as synonymous annotations from different annotation spaces (i.e. Glycolysis in KEGG and GO-BP), which also share most of their genes.

2. GeneTerm Linker, a post-enrichment tool, which focuses on clearing and sorting the results from a previous enrichment analysis. This is achieved by filtering little informative terms (i.e. *cellular process*) and redundant annotations (i.e. *metabolic process* and *primary metabolic process*). The remaining gene-term sets are grouped into *metagroups* based on their shared genes and terms (*reciprocal linkage*) [3].



Different levels of overlap between gene groups.

Functional network
The *functional network* is the representation of the results from a functional enrichment analysis clustering. Each of the genes is one node, and the genes are linked to each other if they are in the same gene-term set. In the plot, the groups (metagroups or clusters) are also represented by a coloured background. The genes that are only in one group, will also be coloured.

The package includes a *report* function which allows to automatically generate an HTML file with the functional network, the terms included in each group, the intersection network and distance matrix. The *intersection network* is a simplified functional network where all the genes that belong to only one metagroup are clustered into a single node. Therefore, the resulting network contains only the nodes in several groups (*intersection* between groups) linked to the metagroups they belong to. The *distance matrix* represents the similarity between the groups based on the genes they share with each other (binary distance).

A pdf version of this report is available at the end of this vignette.

2 Installation

To install *FGNet* from *Bioconductor*, type in your R console:

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

Note on libraries and dependencies: FGNet requires libraries *igraph* (version 0.6 or older), *hwriter*, *RCurl*, *XML* and *R.utils*.
In addition, it may also use *RColorBrewer* and *png*. These libraries are not required for creating the Functional Networks, but they will enhance the network plots.

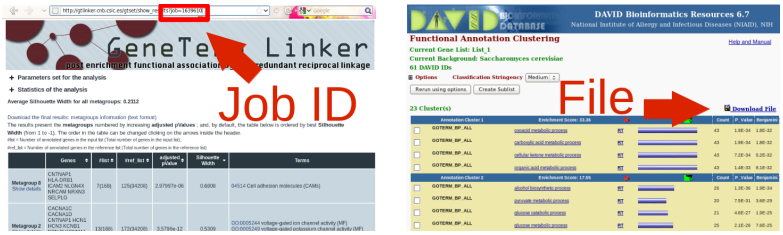
3 Basic example: network from a list of genes/proteins

FGNet can automatically generate an HTML report with the functional analysis and network directly from a gene list. To do so, just load the library, the gene list, and call either *report_david()* or *report_gtLinker()*:

```
> library(FGNet)
> genesYeast <- c("ADA2", "APC1", "APC11", "APC2", "APC4", "APC5", "APC9",
+ "CDC16", "CDC23", "CDC26", "CDC27", "CFT1", "CFT2", "DCP1", "DOC1", "FIP1",
+ "GCN5", "GLC7", "HFI1", "KEM1", "LSM1", "LSM2", "LSM3", "LSM4", "LSM5",
+ "LSM6", "LSM7", "LSM8", "MPE1", "NGG1", "PAP1", "PAT1", "PFS2", "PTA1",
+ "PTI1", "REF2", "RNA14", "RPN1", "RPN10", "RPN11", "RPN13", "RPN2", "RPN3",
+ "RPN5", "RPN6", "RPN8", "RPT1", "RPT3", "RPT6", "SGF11", "SGF29", "SGF73",
+ "SPT20", "SPT3", "SPT7", "SPT8", "TRA1", "YSH1", "YTH1")

> report_david(genesYeast[1:15], geneIdType="GENE_SYMBOL")

> report_gtLinker(genesYeast, organism = "Sc")
```

These report functions are wrappers that include all the steps to query to the functional enrichment tool (DAVID or GtLinker) and generate the functional network (see **Section Workflow**). The first step for creating the functional network is to perform the functional enrichment and clustering. This which can be done directly in R (see following sections) but also through their web sites:

- DAVID: `http://david.abcc.ncifcrf.gov` (Functional Annotation Clustering Tool)
- GeneTerm Linker: `http://gtlinker.dacya.ucm.es`



To use an analysis performed at the web sites, take GeneTerm linker's *job ID* or DAVID's *download file*

To generate the *functional network* out of a previously performed analysis, the jobID (GeneTerm Linker) or the .txt file (DAVID) can be used. *jobName* allows setting the name of the HTML file and the folder where the data will be saved.
i.e. This analysis was performed for an Alzheimer dataset (GSE4757):

```
> report_gtLinker(jobID=1639610, jobName="Alzheimer")
```

When there many overlapping metagroups, a threshold can be set to filter out some metagroups. By default, this threshold is based on the metagroup/cluster *silhouette* (for GeneTerm Linker), or the *enrichment score* (in case of DAVID). (See following sections for more details).

```
> report_gtLinker(jobID=1639610, jobName="Alzheimer_trh0.2",
+ threshold=0.2)
```

A PDF version of this report is available at the end of this vignette.
For help or more details on any functions or their arguments, just set a ? before its name.
i.e. :

```
> ?report_gtLinker
```

4 Advanced example: modifying default parameters and executing the workflow step by step

The report functions are wrappers that include several steps. We will now see each of them:

1. Query the functional enrichment analysis tool (DAVID or GeneTerm Linker)
2. Retrieve the functional enrichment results from the server

3. Create the incidence matrices to build the network
4. Build and plot the networks

The workflows for GeneTermLinker and DAVID are equivalent. They only differ in the two first steps (analysis query and retrieval) since the parameters each of the tools require are different and the results they return are also in different formats. To avoid confusion, we will start with an example with GeneTerm Linker and explain the slight differences when using DAVID. An equivalent example using DAVID is available in section 4.5.

4.1 Step 1: Query the functional enrichment analysis tool

To perform the functional enrichment analysis through this package, all is needed is a list of genes. In case the functional analysis tool is GeneTerm Linker, it is also possible to specify the organism (Homo Sapiens by default).

```
> genesYeast <- c("ADA2", "APC1", "APC11", "APC2", "APC4", "APC5", "APC9",
+ "CDC16", "CDC23", "CDC26", "CDC27", "CFT1", "CFT2", "DCP1", "DOC1", "FIP1",
+ "GCN5", "GLC7", "HFI1", "KEM1", "LSM1", "LSM2", "LSM3", "LSM4", "LSM5",
+ "LSM6", "LSM7", "LSM8", "MPE1", "NGG1", "PAP1", "PAT1", "PFS2", "PTA1",
+ "PTI1", "REF2", "RNA14", "RPN1", "RPN10", "RPN11", "RPN13", "RPN2", "RPN3",
+ "RPN5", "RPN6", "RPN8", "RPT1", "RPT3", "RPT6", "SGF11", "SGF29", "SGF73",
+ "SPT20", "SPT3", "SPT7", "SPT8", "TRA1", "YSH1", "YTH1")
> organism <- "Sc"
> jobID <- query_gtLinker(genesYeast, organism=organism)
```

The annotations spaces to perform the enrichment analysis can also be set. Note the annotation IDs are different for DAVID and GeneTermLinker. DAVID's are available at its API description: `http://david.abcc.ncifcrf.gov/content.jsp?file=DAVID_API.html`. GeneTerm Linker allows these five:

```
> annotations <- c("GO_Biological_Process",
+ "GO_Molecular_Function",
+ "GO_Cellular_Component",
+ "KEGG_Pathways",
+ "InterPro_Motifs")
```

To perform the analysis through DAVID, use *query_david* instead. Just note the annotation names are different, there is no need to provide the organism, and it will return the URL to the text file instead of the jobID.

```
> ?query_david
```

4.2 Step 2: Get the functional analysis results

Since the query function only requests the analysis to be performed, now it is needed to download the results of the analysis. To do so, it can be used either the *URL* to DAVID's text file, or the *jobID* returned by GeneTerm Linker:

```
> results <- getResults_gtLinker(jobID, jobName="Set4yeast")
```

```
[1] 0
Levels: 0
```

jobName allows to set a folder to save the results. In case the analysis results are already downloaded, to draw the network with different parameters, just provide the *jobName* or *path*:

```
> results <- getResults_gtLinker(jobName="Set4yeast",
+ alreadyDownloaded=TRUE)
```

The variable *results* now contains the raw results from the functional enrichment and clustering:

```
> names(results)
```

```
[1] "metagroups"   "geneTermSets" "fileName"
```

```
> head(results$metagroups)
```

Since DAVID instead of a job ID, returns the txt file with the results, it uses the function *getResults_david()*:

```
> ?getResults_david
```

4.3 Step 3: Create the incidence matrices to build the network

The raw results can now be transformed into incidence matrices in order to create the network:

```
> incidMat <- toMatrix(results$geneTermSets)
```

These matrices now contain which genes are in each metagroup or cluster and in each gene-term set:

```
> head(incidMat$metagroupGenesMatrix)
```

```
      1 2 3 4 5 6
ADA2  0 1 0 0 0 0
APC1  0 0 1 0 0 0
APC11 0 0 1 0 0 0
APC2  0 0 1 0 1 1
APC4  0 0 1 0 0 0
APC5  0 0 1 0 0 0
```

```
> incidMat$gtSetGenesMatrix[1:5, 14:18]
```

```
      2.4 2.5 2.6 3.1 3.2
ADA2    0   1   1   0   0
APC1    0   0   0   1   1
APC11   0   0   0   1   1
APC2    0   0   0   1   1
APC4    0   0   0   1   1
```

Filtering the metagrops/clusters based on a score threshold can be done in this step. i.e. for filtering metagroups with *silhouette* under 0.2:

```
> incidMatFiltered <- toMatrix(results$geneTermSets,
+ attribute=results$metagroups[,"Silhouette Width", drop=FALSE], threshold=0.2)
```

To see which metagroups/clusters have been filtered out and now are no longer in the incidence matrices:

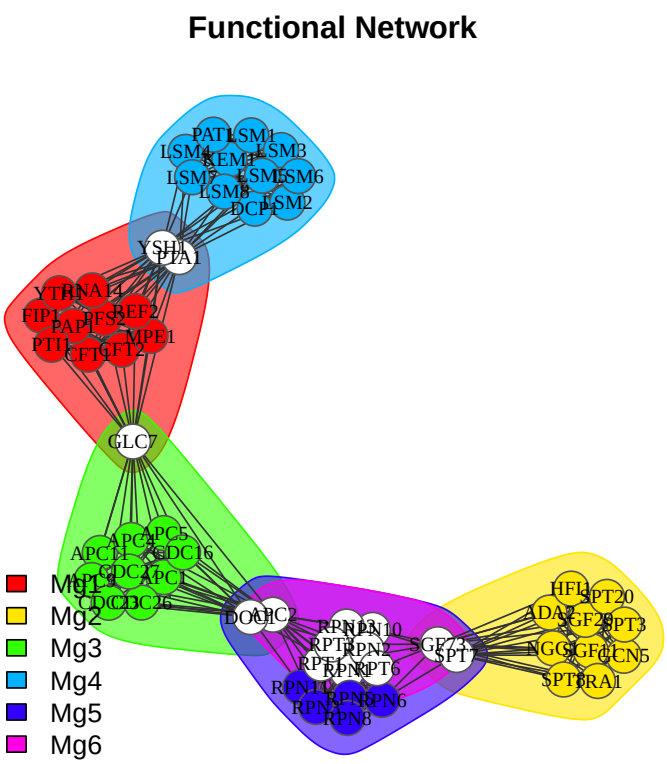
```
> incidMatFiltered$filteredOut
```

```
[1] "5"
```

4.4 Step 4: Build and plot the networks

Once the incidence matrices are ready, they can be transformed into the network. The function *functionalNetwork()* generates the network and plots it:

```
> functionalNetwork(incidMat)
```



By setting the parameter *PlotType="dynamic"* instead of an static plot, it will create an interactive one.

```
> functionalNetwork(incidMat, plotType="dynamic")
```

By default, *functionalNetwork()* only plots the network. In order to obtain the igraph object with the actual graph/network, set the parameter *returnGraph=TRUE*.

```
> fNw <- functionalNetwork(incidMat, returnGraph=TRUE, plotType="none")
```

Since the returned network is an *iGraph* object, it can be used or analyzed as such:

```
> library(igraph)
> fNw

IGRAPH UNW- 59 362 --
+ attr: name (v/c), weight (e/n)

> str(fNw)

> vcount(fNw)

[1] 59

> ecount(fNw)

[1] 362

> sort(betweenness(fNw), decreasing=TRUE)[1:30]

        GLC7         APC2         PTA1        SGF11        RPN10        SGF73
1082.0000000  780.8880952  343.6500000  296.6404762  281.3571429  270.3571429
        RPN6         SPT7         YSH1        RNA14         LSM8         DOC1
 228.9857143  212.8928571  191.2750000  129.2857143  111.5500000  108.5523810
       SGF29         LSM5         KEM1         RPN3         RPT6         CFT1
  93.5833333   85.4690476   75.2500000   58.8023810   33.9071429    9.1250000
        CFT2         FIP1         PFS2         LSM3         LSM7         SPT8
   9.1250000    9.1250000    9.1250000    1.3190476    1.3190476    0.3333333
        LSM2         LSM4         LSM6         MPE1         PAP1         YTH1
   0.2857143    0.2857143    0.2857143    0.1250000    0.1250000    0.1250000
```

Or transformed into other formats...

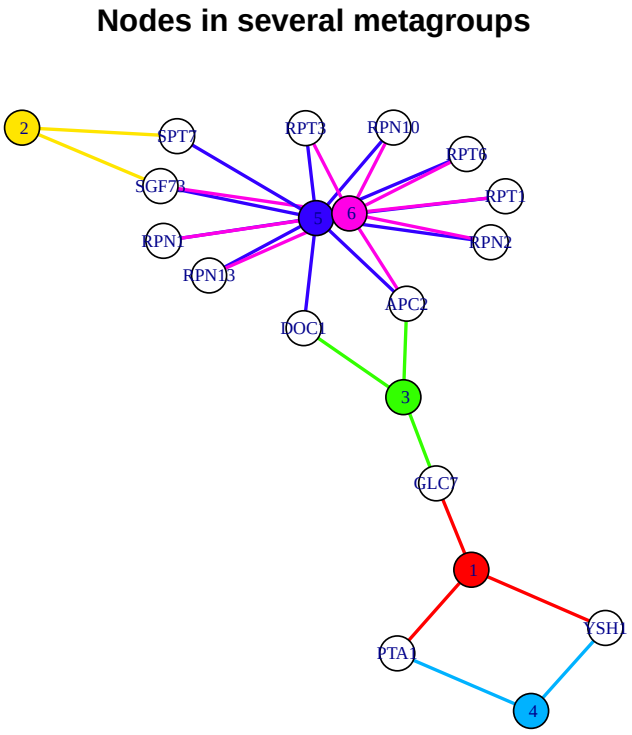
```
> igraph.to.graphNEL(fNw)

A graphNEL graph with undirected edges
Number of Nodes = 59
Number of Edges = 362
```

In dynamic plots (tkplot) it is not possible to draw the metagroup background. To save a dinamically modified network into a static plot with metagroup backgrounds use the `vLayout`. This argument allows to transfer the current layout of a tk plot to a static plot:

```
> functionalNetwork(incidMatFiltered, plotType="dynamic")
> # Modify the layout...
> saveLayout <- tkplot.getcoords(1)   # tkp.id (ID of the tkplot window)
> functionalNetwork(incidMatFiltered, vLayout=saveLayout)
```

In addition to the functional network, when there are nodes in several metagroups, the *intersection network* can also be plotted. This plot is a simplified functional network, containing only the nodes in several metagroups and the metagroups they belong to. The 'metagroup nodes' are the coloured nodes, which can be seen as a cluster of all the genes/proteins that belong only to that metagroup. By default, the intersection network is plotted automatically when requesting an interactive plot, but it can also be plotted by itself:

```
> intersectionNetwork(incidMat,
+  plotType="static", vLayout="kk")
```



The terms in each metagroup can be seen in the raw *results* data frame or using *getTerms()*:

```
> getTerms(results)[1]
```

```
$`Metagroup 1`
     Terms
[1,] "MRNA cleavage and polyadenylation specificity factor complex (CC)"
[2,] "MRNA cleavage (BP)"
[3,] "MRNA cleavage factor complex (CC)"
[4,] "MRNA polyadenylation (BP)"
[5,] "MRNA surveillance pathway"
[6,] "Termination of RNA polymerase II transcription, poly(A)-coupled (BP)"
[7,] "Termination of RNA polymerase II transcription, poly(A)-independent (BP)"
```

4.5 Example using DAVID Functional Annotation Clustering Tool

To generate the functional network with DAVID, the workflow is the same as with GeneTerm Linker. In this example we will use yeast genes from 2 metabolic pathways: *ergosterol biosynthesis* and *sphingolipid metabolism*:

```
> genesMetabolism <- c("YGR175C", "YHR007C", "YMR202W", "YJL167W",
+ "YNL280C", "YGR060W", "YGL001C", "YLR100W", "YLR056W", "YGL012W",
+ "YMR015C", "YML008C", "YHR072W", "YHR190W", "YKL004W", "YBR036C",
+ "YDR294C", "YDR072C", "YKL008C", "YHL003C", "YMR296C", "YDR062W",
+ "YJL134W", "YOR171C", "YLR260W", "YMR298W", "YMR272C", "YPL057C",
+ "YDR297W", "YBR265W", "YPL087W", "YBR183W", "YKR053C")
> # To add the gene label/symbol for the plots...
> library(org.Sc.sgd.db)
> geneLabels <- unlist(as.list(org.Sc.sgdGENENAME)[genesMetabolism])
> names(genesMetabolism) <- geneLabels
```

Option A) Generating the report automatically (getting an initial overview of the networks):

If the analysis has already been performed at the website, it is possible to provide directly the URL or file of the analysis (see Section 3):

```
> report_david(inputFileLocation="http://david... .txt")
```

To analyze a new gene list just provide the list and user email:

```
> report_david(genesMetabolism, email="...", geneLabels=genesMetabolism)
```

Registration is required to use DAVID's web service (`http://david.abcc.ncifcrf.gov/webservice/register.htm`). If the email is not provided, the query will be performed through DAVID's API. The API is limited to 400 genes and does not allow to personalize the clustering arguments (see *?report_david()* for details):

```
> report_david(genesMetabolism)
```

DAVID usually provides many overlapping clusters. In order to simplify the results or explore the overlap between specific clusters, see the following example.

Option B) Executing the steps on the workflow one by one (i.e. to personalize the network):

```
> txtFile <- query_david(genesMetabolism, email="...")
> results <- getResults_david(txtFile, jobName="David_Metabolism", geneLabels=genesM
> incidMat <- toMatrix(results$geneTermSets)
> #functionalNetwork(incidMat)
```

The default gene IDs are Ensembl IDs. The argument *geneLabels* allows to specify an alternative name (i.e. gene symbol) to show in the plots. The full list of annotation categories and IDs for the API is available in `http://david.abcc.ncifcrf.gov/content.jsp?file=DAVID_API.html` and for the web service:

```
> getAllAnnotationCategoryNames(DAVIDWebService$new(email="..."))
> getIdTypes(DAVIDWebService$new(email="..."))
```

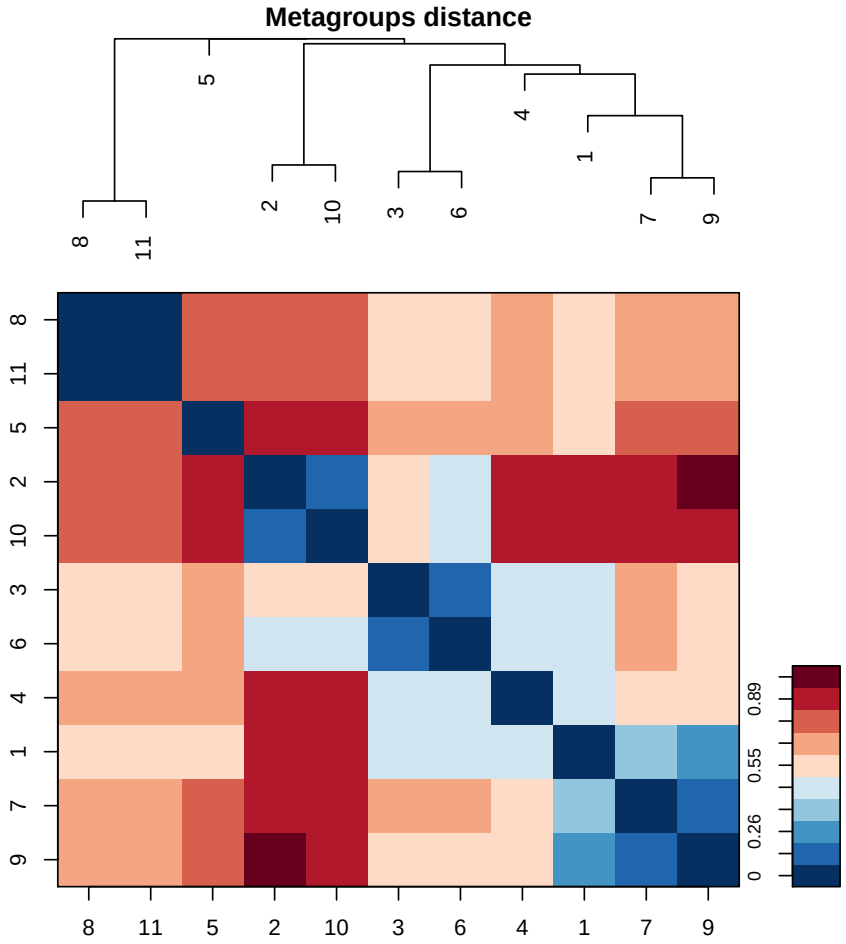
To see the terms in each cluster use *getTerms()*:

```
> getTerms(results)[5]
```

```
$`Cluster 5`
     Terms
[1,] "Ceramide metabolic process"
[2,] "Inositolphosphoceramide metabolic process"
[3,] "Organophosphate metabolic process"
[4,] "Phospholipid metabolic process"
[5,] "Sphingoid metabolic process"
```

DAVID usually provides many overlapping clusters. *plotMetagroupsDistance()* allows to explore the overlap between groups.

```
> distMat <- plotMetagroupsDistance(incidMat$metagroupGenesMatrix)
```



When there are many overlapping clusters it is useful to filter the results or select only a few clusters to plot. The clusters to plot can be selected/filtered based on their Enrichment score, the number of genes they contain, or any other parameter:

a) Filtering based on a **parameter**:

```
> colnames(results$clusters)

[1] "Cluster"         "EnrichmentScore" "nGenes"          "Genes"
[5] "GenesIDs"        "Terms"

> quantile(results$clusters$EnrichmentScore, c(0.10, 0.25, 0.5, 0.75, 0.9))

      10%       25%       50%       75%       90%
 3.169537  3.987255  6.494575 10.085146 15.195045

> incidMatFiltered <- toMatrix(results$geneTermSets,
+  attribute=results$clusters[,"EnrichmentScore", drop=FALSE], threshold=10)
> functionalNetwork(incidMatFiltered)
```

b) Inverse filtering, selection of **lowest values**. i.e. Overlap between clusters with least genes:

```
> quantile(results$clusters$nGenes, c(0.10, 0.25, 0.5, 0.75, 0.9))
> incidMatFiltered <- toMatrix(results$geneTermSets,
+  attribute=-(results$clusters[,"nGenes", drop=FALSE]), threshold=-15)
> functionalNetwork(incidMatFiltered)
```

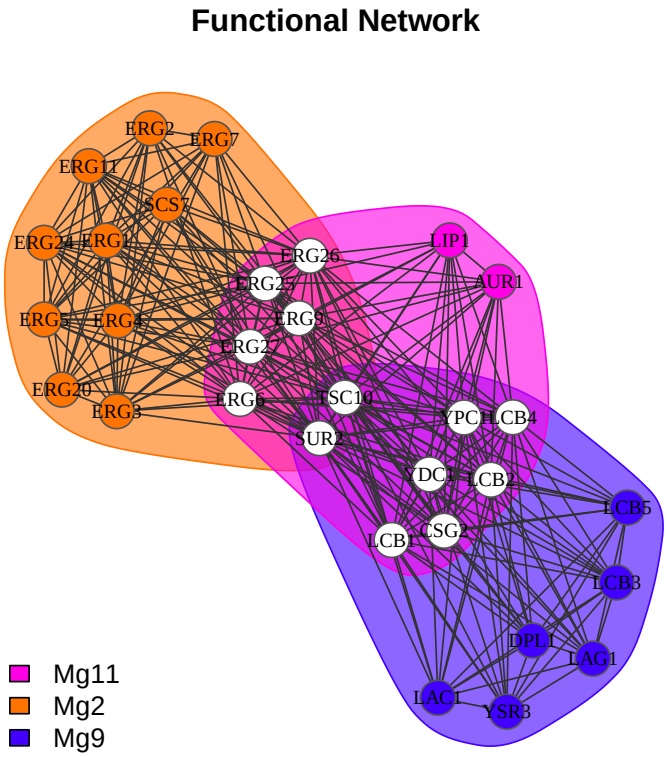
To use any other parameter, add it as column to the to the results$clusters data frame. Then use it to create the incidence matrix and the network.

c) Selecting **clusters with specific terms or keywords**:

```
> keywords <- c("sphingolipid")
> selectedClusters <- sapply(getTerms(results),
+  function(x)
+  any(grep(paste("(", paste(keywords, collapse="|") ,")",sep=""), tolower(x))))
> resultsSclusters <- results
> resultsSclusters$clusters <- cbind(resultsSclusters$clusters,
+  selectedKeywords=as.numeric(selectedClusters))
> incidMatSelectedGroups <- toMatrix(resultsSclusters$geneTermSets,
+  attribute=resultsSclusters$clusters[,"selectedKeywords", drop=FALSE],
+  threshold=1)
> functionalNetwork(incidMatSelectedGroups)
> getTerms(results)[selectedClusters]
```

d) Selection of **specific groups**. i.e. Selecting clusters 2, 9 and 11:

```
> selectedClusters <- rep(FALSE, nrow(results$clusters))
> selectedClusters[c(2,9,11)] <- TRUE
> resultsSclusters <- results
> resultsSclusters$clusters <- cbind(resultsSclusters$clusters,
+  selectedClusters=as.numeric(selectedClusters))
> incidMatSelectedGroups <- toMatrix(resultsSclusters$geneTermSets,
+  attribute=resultsSclusters$clusters[,"selectedClusters", drop=FALSE],
+  threshold=1)
> functionalNetwork(incidMatSelectedGroups)
```



References

[1] Huang DW, Sherman BT, Lempicki RA. *Systematic and integrative analysis of large gene lists using DAVID Bioinformatics Resources.* Nature Protoc. 2009;4(1):44-57.

[2] Huang DW, Sherman BT, Lempicki RA. *Bioinformatics enrichment tools: paths toward the comprehensive functional analysis of large gene lists.* Nucleic Acids Res. 2009;37(1):1-13.

[3] Fontanillo C, Nogales-Cadenas R, Pascual-Montano A, De Las Rivas J (2011) *Functional Analysis beyond Enrichment: Non-Redundant Reciprocal Linkage of Genes and Biological Terms.* PLoS ONE 6(9): e24289. doi:10.1371/journal.pone.0024289

Automatic HTML report generated by FGNet:

Functional Gene Network

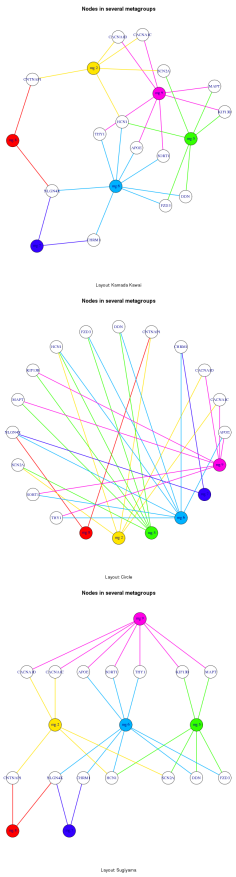
Parameters of the query:

Server/Tool: http://gtlinker.cnb.csic.es
Raw results from functional enrichment and clustering (.txt): [Global overview] [Mg1] [Mg2] [Mg3] [Mg4] [Mg5] [Mg6] [Mg7] [Mg8] [Mg9]
Job ID: 1639610

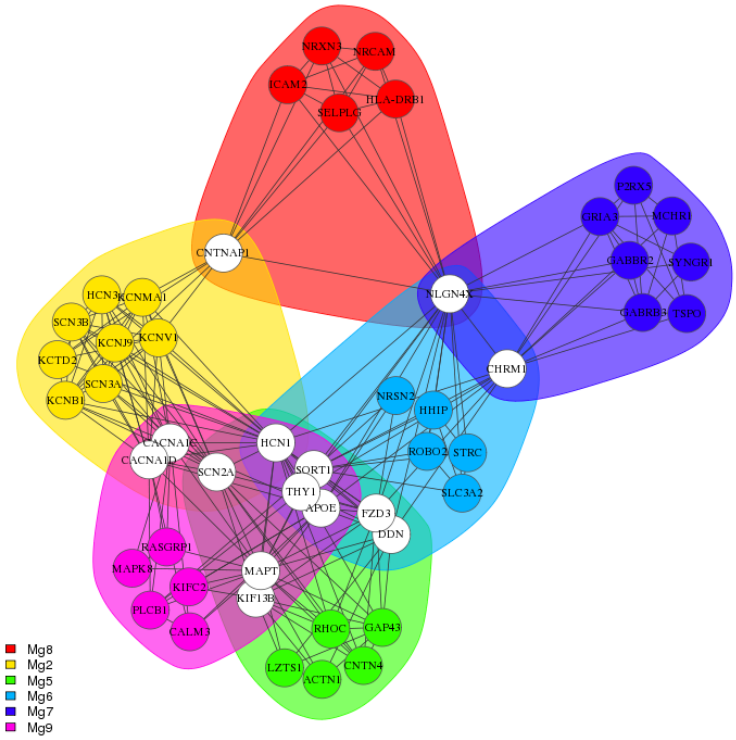
Results:

Number of metagroups: 9
Number of genes included in all metagroups: 77
Filtered metagropus (Silhouette Width < 0.2): Mg1, Mg3, Mg4
Number of genes included in non-filtered metagroups: 49
Functional network in other formats: [iGraph]
Simplified metagroup-terms table (shown below as legend): [Legend]

Nodes in several metagroups:
(Different layouts)



Functional Network



Metagroups (sorted by Silhouette):

Metagroup 8	Silhouette: 0.68	P-value: 3e-06	Num genes: 7	
Cell adhesion molecules (CAMs)				Kegg
Metagroup 2	Silhouette: 0.53	P-value: 3.6e-12	Num genes: 13	
Voltage-gated ion channel activity (MF)				GO
Voltage-gated potassium channel activity (MF)				GO
Voltage-gated potassium channel complex (CC)				GO

Metagroup 5 Silhouette: 0.37 P-value: 3.2e-08 Num genes: 11

Axon (CC) GO

Cell projection (CC) GO

HCN1 SORT1 APOE THY1 DDN CHRM1 FZD3 NLGN4X NRSN2 STRC ROBO2 HHIP SLC3A2

Metagroup 6 Silhouette: 0.36 P-value: 2.8e-07 Num genes: 13

Dendrite (CC) GO

Neuronal cell body (CC) GO

Cell surface (CC) GO

[GO terms tree]

Metagroup 7 Silhouette: 0.28 P-value: 2.1e-06 Num genes: 9

Postsynaptic membrane (CC) GO

Synapse (CC) GO

Neuroactive ligand-receptor interaction Kegg

[GO terms tree]

Metagroup 9 Silhouette: 0.27 P-value: 0.00035 Num genes: 12

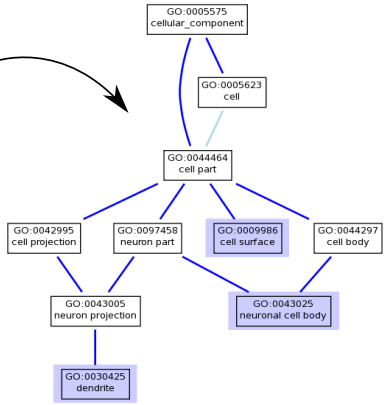
Enzyme binding (MF) GO

Microtubule (CC) GO

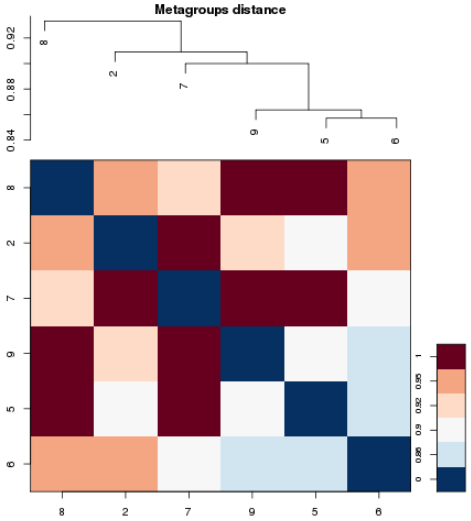
Alzheimer's disease Kegg

MAPK signaling pathway Kegg

[GO terms tree]



Distances between Metagroups:



Distance matrix:

	Mg8	Mg2	Mg7	Mg9	Mg5	Mg6
Mg8	0	0.95	0.93	1	1	0.95
Mg2	0.95	0	1	0.91	0.91	0.96
Mg7	0.93	1	0	1	1	0.9
Mg9	1	0.91	1	0	0.9	0.86
Mg5	1	0.91	1	0.9	0	0.86
Mg6	0.95	0.96	0.9	0.86	0.86	0