Using GoNetwork

Sara Linker

April 13, 2017

Introduction

The purpose of GoNetwork is to calculate networks based on GO annotation and to identify clusters that are differentially enriched between conditions based on expression data. Networks created using this program are easily plotted with molecular visualization software platforms such as Cytoscape.

Background

Functional annotation is often used as a descriptive tool to examine gene sets identified through upstream analyses. This has been useful in describing what types of pathways and mechanisms are attributable to a given system, however there are limitations in both the ability to describe a system and to quantify these descriptions with many of the current functional annotation methods. GONetwork is a user-friendly method that increases the information returned from functional enrichment analyses by reducing the reliance on

Run

GONetwork is designed to take a list of genes and calculate a similarity matrix based on the hierarchical representation of GO terms and taking into account the sparsity of the dataset. We've provided a test dataset to work with generated from real data examining neural progenitor cells throughout differentiation into neurons. The dataset *differentiation* contains an object *genes* with the significant genes from this analysis.

```
> data(differentiation)
> head(genes)
```

```
[1] "GTF3C2-AS1" "PRMT5-AS1"  "FANCD2OS"   "MED4-AS1"   "LRRC37A16P"
[6] "PKD2L2"
```

Generate GOterm Matrix

Use **getGo()** to create a matrix of GO terms based on the gene list. GoNetwork improves upon previous functional gene enrichment tools by applying weights to the binary GO term matrix based on the number of parents in the GO hierarchy. The time limitation is the connection to the GO term database.

```
> M <- getGo(genes,species = "human")
```

Once the GO term matrix is created, use **GoTheDist()** to calculate the distances between genes. **GoTheDist()** allows users to restrict the scope of GO terms based on minimum number of genes that fall under a GO term and minimum number of parents. Users may also choose whether the distances are calculated using the cosine function, manhattan distance or euclidean method: *"cosine", "manhattan" or "euclidean"*. In this case, the cosine function is used.

```
> D <- GoTheDist(M,method="cosine",Min=6,minparents=15)
> #quantile(D)
```

After distances have been calculated, convert the matrix to Cytoscape format using **cyto()**. Stringency can be adjusted by changing the lower and upper cutoff values. *low.cutoff* is the minimum distance between two genes for the connection to be sent to cytoscape view, whereas *high.cutoff* is the maximum distance. *Note: this may be confusing to users because low.cutoff is used for cosine and high.cutoff must be used for euclidean and manhattan.*

```
> tab<-cyto(D,cutoff = 0.2)
> head(tab)

       origin destination  distance
6     SLC24A5      PKD2L2 0.6259316
91      ASIC1      PKD2L2 0.7453083
756  SLC25A35      SLC3A1 0.5873734
808  SLC22A17      SLC3A1 0.7525478
1166   PKD2L2     SLC24A5 0.6259316
1411    CIITA        SLA2 0.6693514
```

```
> tab2 <-AssignCluster(tab,cutoff = 0.4,return_in_cytotable = TRUE)
> head(tab2)

       origin destination  distance group
6     SLC24A5      PKD2L2 0.6259316    NA
91      ASIC1      PKD2L2 0.7453083    NA
756  SLC25A35      SLC3A1 0.5873734     4
808  SLC22A17      SLC3A1 0.7525478     4
1166   PKD2L2     SLC24A5 0.6259316    NA
1411    CIITA        SLA2 0.6693514     1
```

With the results from **cyto()** you can continue directly to plotting in cytoscape. To continue on a calculate differential representation of gene groups within GO term clusters you can continue on by first assigning clusters with **AssignCluster()**.

```
> k <-AssignCluster(tab,cutoff = 0.4)
> table(k$k)

 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 
16 10  3  2  2  2  2  9 13  2 11  3  2  7  2  2  2  2  2  2 
```

You can then calcualte the significance of those results with **testClusters()**. Test clusters requires an additional variable **group**. **group** is a matrix where the first column contains all of the gene names from **genes** and the second column contains the group designation to be tested

```
> head(group)

       genes group
1 GTF3C2-AS1  high
2  PRMT5-AS1  high
3   FANCD2OS  high
4   MED4-AS1  high
5 LRRC37A16P  high
6     PKD2L2  high

> k.chi.test <- testClusters(k,group)
> findTerms(M,as.character(k[k$k == 14 & !is.na(k$k),"genes"]),proportion.shared = 0.4)

[1] "Rho guanyl-nucleotide exchange factor activity"
[2] "GTPase activator activity"
[3] "protein binding"
[4] "cytosol"
[5] "Rho protein signal transduction"
[6] "regulation of Rho protein signal transduction"
[7] "positive regulation of GTPase activity"
[8] "regulation of small GTPase mediated signal transduction"
```

Other Functions Included

Use **symbolConvert()** to convert alternative gene notations to the desired format. In this case, a list of entrezgene IDs are converted to their respective HGNC gene symbols.

```
> #head(gene.numbers)
> #gene.symbols<-symbolConvert(gene.numbers,"entrezgene","hgnc_symbol")
> #head(gene.symbols)
```

Use **exactTerms()** to identify GO terms containing a specified character string

```
> #e<-exactTerms(colnames(M), term = "vesicle")
> #head(e)
```

Use **findGenes()** to return a list of genes with GO terms containing a specified character string. We can use one of the terms found using the exactTerms() function above:

```
>
> #head(findGenes(M, term = "acrosomal.vesicle"))
>
```

If you would like to identify the GO terms that a group of genes have in common, for instance, a list of genes taken from a cluster seen after plotting in cytoscape, use **findTerms()**:

```
>
> #findTerms(M, geneA = "APOE" ,geneB = "FNBP1L",geneC="TBC1D10A")
```