Brittney Shabits
10055320

clusterProfiler Tutorial

Introduction

With the increasing use of high-throughput techniques, such as mass spectrometry and RNA-seq, for systems-level studies comes the new challenge of analyzing the massive amounts of data that is the result of these methods. Without proper analysis and visualization techniques, it can be difficult to extract relevant information resulting in important data being misinterpreted or even overlooked. When analysing large data sets, it is common to use a clustering approach, whereby data is organized into specific groups in which the objects in one group are more similar to each other than to objects in another group. Clustering analysis provides a means to extract information from a cumbersome data set; however, there remains the problem of having a suitable visualization method to properly present the end result. High-throughput data can be inefficiently presented using bulky tables, preventing the audience from understanding key conclusions. Although all the information is there, over-loaded tables are certainly not appealing to the eye in a presentation or paper. These two main problems of (1) managing large data sets by clustering and (2) appropriate visualization techniques are addressed by the R package *clusterProfiler*[1].

Background

Two databases widely used for finding shared traits, themes, and connections between genes/proteins for clustering analysis are the Gene Ontology (GO) Consortium and Kyoto Encyclopedia of Genes and Genomes (KEGG). GO is a database dedicated to consistently and accurately describing all gene products in a species-independent manner[2]. To do this, three structured ontologies were developed; biological processes (BP), cellular components (CC), and, molecular functions (MF). KEGG is a separate database that uses molecular-level information to understand high level functions and therefore provides more detailed information on overall systems and pathways[3].

clusterProfiler is an R package developed for comparing and visualizing large amounts of high-throughput proteomic/genomic data by clustering and comparative analysis[1]. *clusterProfiler* supports 19 species, all of which are supported by the GO and KEGG databases[4]. With *clusterProfiler*, it is simple to group large data sets into clusters with which enrichment analysis can be performed. This type of analysis allows you to get a general idea of what kind of data you have, if there is any enrichment in your samples, and if any of the groupings are significant. The clustering can be based on pathway, location, function, or process, depending on which database is used. Equally as important as the clustering analysis is the visualization techniques by which the data is then presented. The clustered data outputs from *clusterProfiler* are compatible with many different plotting functions to create figures such as bar plots, enrichment maps, and category-gene-network plots, thus providing a wide range of visualization methods to accurately depict the clustering results.

R Code

1. Set-up

Clear the workspace with `rm(list=ls(all=TRUE))` so we have a blank environment to start with.

Install the package "*clusterProfiler*" from Bioconductor. Depending on what organism you are working with, you may also need to install a database for the genome-wide annotation of that specific organism. Here I will demonstrate with the yeast package "org.Sc.sgd.db", which is primarily based on *Saccharomyces* Genome Database (SGD).

```
rm(list=ls(all=TRUE))

source("https://bioconductor.org/biocLite.R")
biocLite("clusterProfiler")
source("http://bioconductor.org/biocLite.R")
biocLite("org.Sc.sgd.db")

require(clusterProfiler)
require(org.Sc.sgd.db)
```

Set your working directory to the folder that contains the data you want to use. Make sure to change all \ symbols to /, as shown below.

```
setwd("C:/Users/VZlab/Documents/Master's/R course/Final Assignment")
```

2. Reading in your data and bitr

The easiest way to upload your data to R Studio for use with *clusterProfiler* is through a .csv file. The dataset that I will be using in this tutorial is "Mass spec data.csv". Use the `read.csv` function to read in your .csv file, indicating if you have a header for each of the columns and if you want to treat strings as factors or not.

Description of the example dataset "Mass spec data.csv" - this file has the entire list of yeast genes (~6600) in the first column called "Yeastgenome". The second column, "Gat1proteins", is a list of the proteins that were identified by mass spectrometry (MS) analysis of samples from affinity pull-downs with my protein of interest, Gat1.

```
data <- read.csv("Mass spec data.csv", header = T, stringsAsFactors = F)
head(data)

##  Yeastgenome Gat1proteins
##1     YAL001C      YDR037W
##2     YAL002W      YPR095C
##3     YAL003W      YGL229C
##4     YAL005C      YBR260C
##5     YAL007C      YLL048C
##6     YAL008W      YIL113W
```

To distinguish the columns as two separate entities, we can save each column with their own name. This will make our code shorter and more simple when using *clusterProfiler* functions.

```
Universe <- data$Yeastgenome
head(Universe)

##[1] "YAL001C" "YAL002W" "YAL003W" "YAL005C" "YAL007C" "YAL008W"

Gat1 <- data$Gat1proteins
head(Gat1)

##[1] "YDR037W" "YPR095C" "YGL229C" "YBR260C" "YLL048C" "YIL113W"
```

A very useful tool that *clusterProfiler* provides is the `bitr` function. This is a Biological Id TRanslator which allows you to convert your list of genes/proteins from one "type" of name to another. This is extremely helpful, for example, if you want your output figures to have the common protein names but your MS data provides only the systematic name or some other ID. The various options of keytypes for each organism can be printed with `keytypes()`.

```
eg <- bitr(Gat1, fromType="ENSEMBL", toType="COMMON", "org.Sc.sgd.db")
head(eg)

##  ENSEMBL                                              COMMON        SGD
##1 YDR037W                                                GCD5 S000002444
##2 YDR037W                                                KRS1 S000002444
##3 YDR037W                                lysine--tRNA ligase KRS1 S000002444
##4 YPR095C Arf family guanine nucleotide exchange factor SYT1 S000006299
##5 YPR095C                                                SYT1 S000006299
##6 YGL229C                                                SAP4 S000003198

keytypes(org.Sc.sgd.db)
```

3. groupGO

The `groupGO` function provides GO classification and clustering based on GO distribution at a specific level. `groupGO` will give you the ID number and description of all clusters at the specified level as well as the names and number of genes/proteins from your data set that fit into each cluster.

By providing your gene/protein list and organism type you can begin clustering. For the `ont` parameter, you must choose one of the three subontologies from GO; either BP, CC, or MF[2].

> BP = describes the organized pathways and larger processes which include the actions of one or more gene products
> CC = describes the component of the cell where these gene products are active
> MF = describes the activities of the gene products at the molecular level

These ontologies are then described by different levels; level 1 is the most general description of the ontology and as the levels increase, so does the classification specificity. The GO level in

which you want to analyze your data can be selected for with the `level` argument. Make the `readable` argument false if you want to keep the naming of your data the same in your output as in your input (if true, the input IDs will be converted to gene symbols – something you may or may not want).

```
ggo <- groupGO(gene = Gat1, organism = "yeast", ont = "BP", level = 4,
readable = F)
head(summary(ggo))

##                   ID                                          Description Count GeneRatio
##GO:0000747 GO:0000747                       conjugation with cellular fusion     4      4/243
##GO:0000909 GO:0000909 sporocarp development involved in sexual reproduction     0      0/243
##GO:0007276 GO:0007276                                    gamete generation     0      0/243
##GO:0007618 GO:0007618                                              mating     0      0/243
##GO:0009566 GO:0009566                                        fertilization     0      0/243
##GO:0034293 GO:0034293                                     sexual sporulation     5      5/243

##                                        geneID
##GO:0000747         YIL113W/YFR022W/YOL117W/YBR040W
##GO:0000909
##GO:0007276
##GO:0007618
##GO:0009566
##GO:0034293 YNL098C/YMR183C/YNL194C/YJL168C/YLL040C
```

4. enrichGO

Similar to `groupGO`, `enrichGO` provides a list of GO clusters and the corresponding proteins; however, `enrichGO` also allows you to see the enrichment of your data against background genes/proteins. Again, the output will give you the ID number and description of all GO classifications at the specified level as well as the names and number of genes/proteins from your data set that were found in each cluster. `enrichGO` output also shows p- and q-values as well as p-values after any adjustment.

To determine the probability that the clustering result occurred by chance, p-values are calculated using the hypergeometric distribution model:

$$p = 1 - \sum_{i=0}^{k-1} \frac{\binom{M}{i}\binom{N-M}{n-i}}{\binom{N}{n}}$$

where *N* is total number of background genes, *M* is the number of genes within the background distribution that are annotated to the cluster of interest, *n* is the size of the list of genes of interest, and *k* is the number of genes within that list which are annotated to the cluster[4].

In our example, the `gene` parameter is our "Gat1" list and the `universe` (background data) is the "Universe" data set (ie. all yeast genes). Again, choose one of the three subontologies and set the `readable` parameter. The `readable` parameter is again set to false so we can keep the naming of the data consistent. `enrichGO` has an additional parameter to set the p-value cut-off

(pvalueCutoff), where no clusters will be shown if they are above this set value. By adjusting the pvaluecutoff you can control which clusters are shown. There are two additional parameters in the enrichGO function (not shown below) that allows you to manipulate the statistical analysis of the output, by providing a q-value cut-off (qvalueCutoff) or a method to adjust the calculated p-values (pAdjustMethod).

```
ego <- enrichGO(gene = Gat1, universe = Universe, organism = "yeast", ont =
"MF", pvalueCutoff = 0.5, readable = F)

head(summary(ego))

##                   ID                                                Description GeneRatio
##GO:0016462 GO:0016462                                    pyrophosphatase activity    34/242
##GO:0016817 GO:0016817                hydrolase activity, acting on acid anhydrides    34/242
##GO:0016818 GO:0016818     hydrolase activity, in phosphorus-containing anhydrides    34/242
##GO:0008094 GO:0008094                                DNA-dependent ATPase activity    11/242
##GO:0001012 GO:0001012             RNA polymerase II regulatory region DNA binding     9/242
##GO:0017111 GO:0017111                          nucleoside-triphosphatase activity    30/242

##            BgRatio       pvalue   p.adjust     qvalue
##GO:0016462 424/5820 0.0001325029 0.01912459 0.01664423
##GO:0016817 424/5820 0.0001325029 0.01912459 0.01664423
##GO:0016818 424/5820 0.0001325029 0.01912459 0.01664423
##GO:0008094  87/5820 0.0008715510 0.08139915 0.07084209
##GO:0001012  63/5820 0.0010611337 0.08139915 0.07084209
##GO:0017111 402/5820 0.0011279327 0.08139915 0.07084209

##            geneID
##GO:0016462
YBL022C/YBR264C/YCL050C/YCR052W/YDR284C/YDR488C/YDR545W/YEL042W/YER047C/YER173W/YFR009W/YFR031C/Y
GL048C/YHR023W/YHR065C/YHR120W/YHR169W/YIL126W/YJR035W/YKR008W/YLL048C/YLR188W/YLR467W/YML065W/YM
L127W/YMR106C/YNL088W/YNL098C/YNL118C/YOL094C/YOR117W/YOR165W/YOR304W/YOR396W
##GO:0016817
YBL022C/YBR264C/YCL050C/YCR052W/YDR284C/YDR488C/YDR545W/YEL042W/YER047C/YER173W/YFR009W/YFR031C/Y
GL048C/YHR023W/YHR065C/YHR120W/YHR169W/YIL126W/YJR035W/YKR008W/YLL048C/YLR188W/YLR467W/YML065W/YM
L127W/YMR106C/YNL088W/YNL098C/YNL118C/YOL094C/YOR117W/YOR165W/YOR304W/YOR396W
##GO:0016818
YBL022C/YBR264C/YCL050C/YCR052W/YDR284C/YDR488C/YDR545W/YEL042W/YER047C/YER173W/YFR009W/YFR031C/Y
GL048C/YHR023W/YHR065C/YHR120W/YHR169W/YIL126W/YJR035W/YKR008W/YLL048C/YLR188W/YLR467W/YML065W/YM
L127W/YMR106C/YNL088W/YNL098C/YNL118C/YOL094C/YOR117W/YOR165W/YOR304W/YOR396W
##GO:0008094
YCR052W/YER173W/YHR120W/YIL126W/YJR035W/YKR008W/YML127W/YMR106C/YNL088W/YOL094C/YOR304W
##GO:0001012
YDL106C/YER028C/YER148W/YGL043W/YKL112W/YMR043W/YNL068C/YOR113W/YOR140W
##GO:0017111
YBL022C/YBR264C/YCR052W/YDR488C/YDR545W/YER047C/YER173W/YFR009W/YFR031C/YGL048C/YHR023W/YHR065C/Y
HR120W/YHR169W/YIL126W/YJR035W/YKR008W/YLL048C/YLR188W/YLR467W/YML065W/YML127W/YMR106C/YNL088W/YN
L098C/YOL094C/YOR117W/YOR165W/YOR304W/YOR396W

##            Count
##GO:0016462    34
##GO:0016817    34
##GO:0016818    34
##GO:0008094    11
##GO:0001012     9
##GO:0017111    30
```

5. enrichKEGG

This function is very similar to `enrichGO` in that all output variables are the same. However, `enrichKEGG` will give you enrichment of your input of genes in terms of the KEGG database. `enrichKEGG` provides an output of KEGG pathways and lists the genes from your dataset that are enriched in each, all with p- and q-values. Similar to `enrichGO`, `qvalueCutoff` and `pAdjustMethod` (not shown below) are additional parameters for manipulation of the statistical analysis.

```
kk <- enrichKEGG(gene = Gat1, universe = Universe, organism = "yeast",
pvalueCutoff = 0.5, readable = F)
head(summary(kk))
```

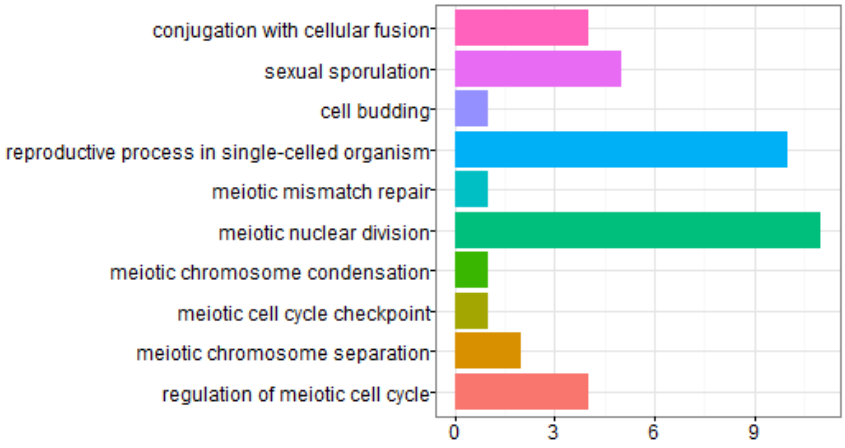
```
##               ID                  Description GeneRatio BgRatio      pvalue  p.adjust    qvalue
##sce00513 sce00513       N-glycan biosynthesis      6/95 30/1971 0.002481596 0.1222744 0.1176776
##sce00600 sce00600     Sphingolipid metabolism      4/95 14/1971 0.003493555 0.1222744 0.1176776
##sce00564 sce00564     Glycerolipid metabolism      6/95 37/1971 0.007392820 0.1724991 0.1660142

##                                                 geneID Count
##sce00513 YJR075W/YGL022W/YLR057W/YJL183W/YBR015C/YJL002C     6
##sce00600                  YMR296C/YDR062W/YDR294C/YHL003C     4
##sce00564 YKR067W/YPL110C/YOR175C/YDR284C/YBL011W/YNL130C     6
```

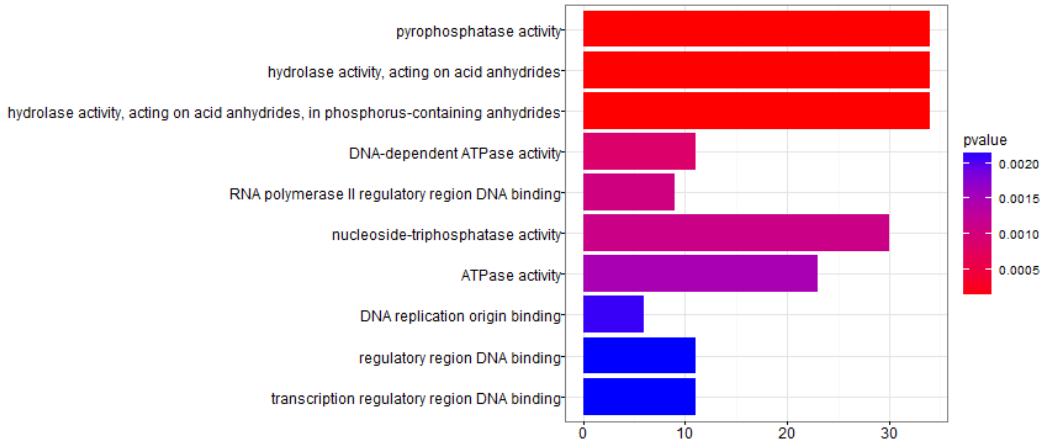
6. Visualization
    a. barplot

The `barplot` output provides a figure to show a set number of clusters and the number of objects in each. Set the `drop` parameter to true if you want to skip over any clusters that have zero objects and only show ones that have one or more. With the `showCategory` parameter, you can select how many clusters you want to be visible.

```
barplot(ggo, drop = T, showCategory = 10)
```



```
barplot(ego, showCategory = 10)
```

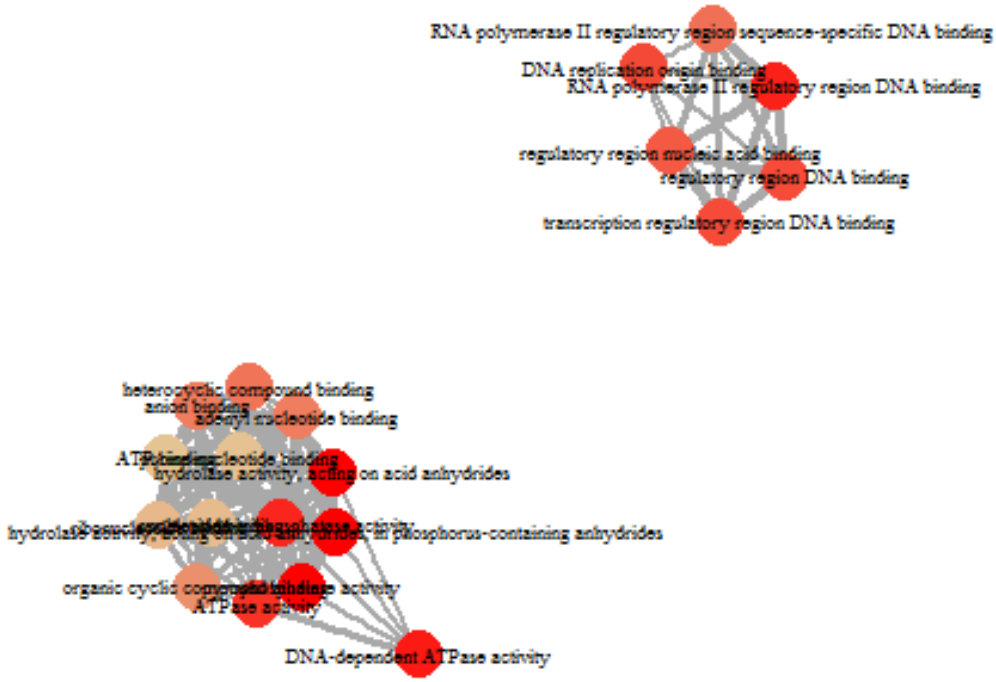


b. enrichMap

The `enrichMap` output gives an enrichment map of the `enrichGO` output, allowing the results to be visualized as a network of clusters. The nodes represent the GO classifications, while the edges represent mutual overlap.

The maximum number of nodes shown can be adjusted with `n`, and the font size can be adjusted with `vertex.label.cex`. However, even with these adjustments, the figure can still sometimes look crowded (such as below). Therefore, if the `fixed` parameter was switched to false, R will open up tkplot, a program which allows you to rearrange the nodes and adjust the overall look of the figure (recommended).

```
enrichMap(ego, vertex.label.cex = 0.5, fixed = T, n = 20)
```

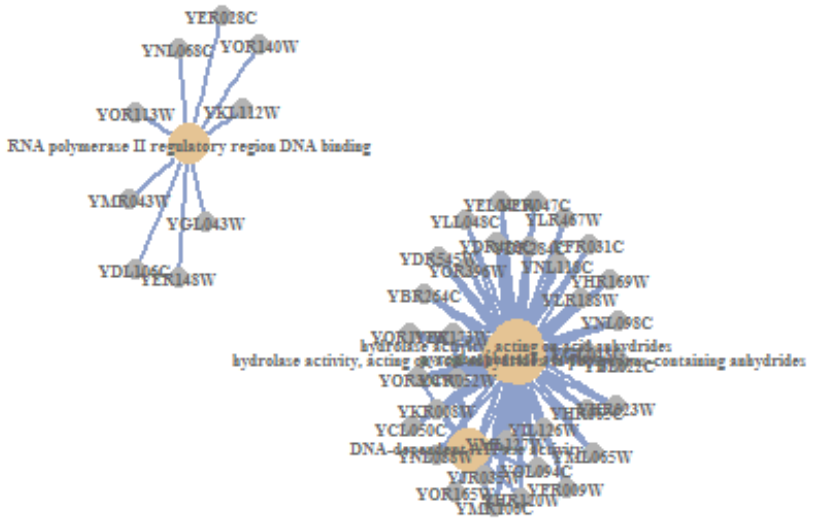


c. cnetplot

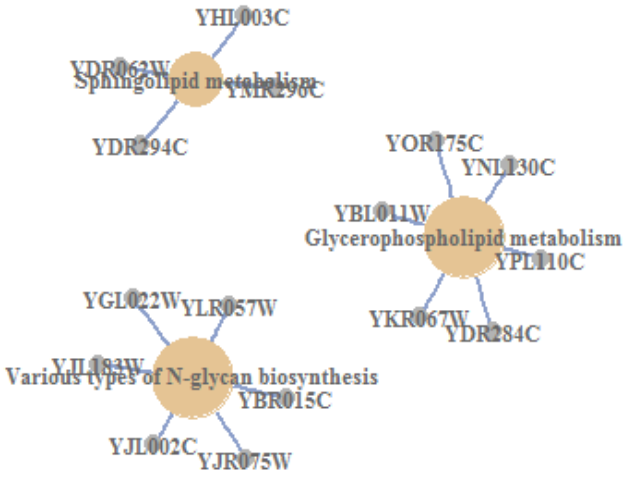
The `cnetplot` output prepares a category-gene-network plot. Here we can see a visualization of the exact genes/proteins that have been enriched in each GO classification or KEGG pathway (represented by nodes). This is important when you want to consider complex association of your data. `cnetplot` is compatible with either the `enrichGO` or `enrichKEGG` output.

Like the enrichment maps, `vertex.label.cex` adjusts the font size, and you can set `fixed` to false if you want to open tkplot to adjust the spacing and modify the figure on your own. Shown below are the unedited figures. `categorySize` can be either "pvalue" or "geneNUM", depending if you want the size of the nodes to be based on number of genes connecting to the concept or the p-value of enrichment test, respectively.

```
cnetplot(ego, categorySize = "pvalue", fixed = T, vertex.label.cex = 0.5)
```




```
cnetplot(kk, categorySize = "geneNum", fixed = T, vertex.label.cex=0.75)
```



References

(1) Yu, G., Wang, L.-G., Han, Y., and He, Q.-Y. (2012) clusterProfiler: an R package for comparing biological themes among gene clusters. *OMICS 16*, 284–7.

(2) Ontology Documentation | Gene Ontology Consortium.

(3) KEGG: Kyoto Encyclopedia of Genes and Genomes.

(4) Yu, G. (2013) Using clusterProfiler to identify and compare functional profiles of gene lists. *R包 16*, 284–287.