TABLE OF CONTENTS

1 Introduction

When we have a large list of genes of interest, such as a list of differentially expressed genes obtained from an RNA-Seq experiment, how do we extract biological meaning from it?

One way to do so is to perform functional enrichment analysis. This method consists of applying statistical tests to verify if genes of interest are more often associated to certain biological functions than what would be expected in a random set of genes. In this appnote you will learn about how to perform enrichment analysis.

The Gene Ontology (GO) is a major bioinformatics initiative to unify the representation of gene and gene product attributes across all species. It is a structured, controlled vocabulary for the classification of gene function at the molecular and cellular level. It is divided in three separate sub-ontologies or GO types: biological process (e.g., signal transduction), molecular function (e.g., ATPase activity) and cellular component (e.g., ribosome). These sub-ontologies are structured as directed acyclic graphs (a hierarchy with multi-parenting) of GO terms.

Kyoto Encyclopedia of Genes and Genomes (KEGG) is a collection of manually drawn pathway maps representing molecular interaction and reaction networks. These pathways cover a wide range of biochemical processes that can be divided into 7 broad categories: Metabolism; Genetic information processing; Environmental information processing; Cellular processes; Organismal systems; Human diseases; Drug development.

GO and KEGG are the most frequently used for functional analysis. They are typically the first choice because of their long-standing curation and availability for a wide range of species.

Other gene sets include but are not limited to Disease Ontology (DO), Disease Gene Network (DisGeNET), wikiPathways, Molecular Signatures Database (MSigDb).

The scripts and test data are available at GitHub:

https://github.com/MGI-APAC-FBS/App-Note_RNA-seq/

2 Gene Ontology Enrichment analysis (clusterProfiler)

This package supports functional characteristics of both coding and non-coding genomics data for thousands of species with up-to-date gene annotation. GO comprises three orthogonal ontologies, i.e. molecular function (MF), biological process (BP), and cellular component (CC).

2.1 Installation

The easiest way is to use Bioconductor for the installation. We just need to find an appropriate Bioconductor version for your R system, i.e., Bioconductor 3.14 is only for R4.1, and 3.10 is for R3.6.

In this case we will use R4.1. Please refer to the following linkage for the installation:

https://bioconductor.org/packages/release/bioc/html/clusterProfiler.html

In some cases, there maybe some software missed in your operating system, i.e., you cannot install XML package on R-Studio Pro, which gives an error as:

```
checking for pkg-config... /usr/bin/pkg-config
checking for xml2-config... /opt/python/3.6.5/bin/xml2-config
USE_XML2 = yes
SED_EXTENDED_ARG: -E
Minor 9, Patch 8 for 2.9.8
Located parser file -I/opt/python/3.6.5/include/libxml2 -I/opt/python/3.6.5/include/
Checking for 1.8:  -I/opt/python/3.6.5/include/libxml2 -I/opt/python/3.6.5/include
Using libxml2.*
checking for gzopen in -lz... yes
checking for xmlParseFile in -lxml2... no
checking for xmlParseFile in -lxml... no
configure: error: "libxml not found"
ERROR: configuration failed for package ‘XML’
* removing ‘/home/juri.kuusik/R/x86_64-pc-linux-gnu-library/3.6/XML’
Warning in install.packages :
  installation of package ‘XML’ had non-zero exit status

The downloaded source packages are in
    ‘/tmp/RtmpCCCTwU/downloaded_packages’
```

It means the libxml2 is missing in your operating system, so what you should do is to install it separately, if the system you are using is CentOS, you should install it in command as:

```
yum install libxml2
```

For GO terms annotation, “org.Hs.eg.db” package is required. If R gives error as:

```
Error in is(OrgDb, "character") : object 'org.Hs.eg.db' not found
```

Please install manually:

```
BiocManager::install('org.Hs.eg.db',force = TRUE)
```

2.2 Supported Organisms

GO analyses (groupGO(), enrichGO() and gseGO()) support organisms that have an OrgDb object available for about 20 species.

If a user has GO annotation data (in a data.frame format with the first column as gene ID and the second column as GO ID), they can use the enricher() and gseGO() functions to perform an over-representation test and gene set enrichment analysis.

- groupGO(): Functional Profile of a gene set at specific GO level. Given a vector of genes, this function will return the GO profile at a specific level.
- enrichGO(): GO Enrichment Analysis of a gene set. Given a vector of genes, this function will return the enrichment GO categories after FDR control.
- gseGO: Gene Set Enrichment Analysis of Gene Ontology

2.3 Prepare your own geneList

GSEA analysis requires a ranked gene list, which contains three features:

- numeric vector: fold change or other type of numerical variable
- named vector: every number has a name, the corresponding gene ID
- sorted vector: number should be sorted in decreasing order

If you import your data from a csv file, the file should contains two columns, one for gene ID (no duplicated ID allowed) and another one for fold change. You can prepare your own geneList via the following command:

Example (using DESeq2 result as input):

```
d = read.csv("DiffGeneExp_DESeq2.csv")
## 1st column is ID
## 2nd column is basemean
## 3rd column is log2FC

## feature 1: numeric vector
geneList = d[,3]
## feature 2: named vector
names(geneList) = as.character(d[,1])
## feature 3: decreasing order
geneList = sort(geneList, decreasing = TRUE)
```

2.4 Run Go Analysis

GO classification

In clusterProfiler, the groupGO() function is designed for gene classification based on GO distribution at a specific level.

GroupGO() Arguments

`gene`	**a vector of entrez gene id.**
`OrgDb`	OrgDb
`keyType`	key type of input gene
`ont`	One of "MF", "BP", and "CC" subontologies.
`level`	Specific GO Level.
`readable`	if readable is TRUE, the gene IDs will mapping to gene symbols.

Example:

```
library(clusterProfiler)
gene <- names(geneList)[abs(geneList) > 2]

# Entrez gene ID
head(gene)

## [1] "4312"  "8318"  "10874" "55143" "55388" "991"
```

```
ggo <- groupGO(gene     = gene,
               OrgDb    = org.Hs.eg.db,
               ont      = "CC",
               level    = 3,
               readable = TRUE)
head(ggo)

##                    ID                Description Count GeneRatio geneID
```

```
## GO:0000133 GO:0000133                 polarisome     0     0/207
## GO:0000408 GO:0000408          EKC/KEOPS complex     0     0/207
## GO:0000417 GO:0000417                HIR complex     0     0/207
## GO:0000444 GO:0000444    MIS12/MIND type complex     0     0/207
## GO:0000808 GO:0000808 origin recognition complex     0     0/207
## GO:0000930 GO:0000930      gamma-tubulin complex     0     0/207
```

GO over-representation analysis

The clusterProfiler package implements enrichGO() for gene ontology over-representation test.

Any gene ID type that is supported in OrgDb can be directly used in GO analyses. Users need to specify the keyType parameter to specify the input gene ID type.

EnrichGO() arguments:

`gene`	**a vector of entrez gene id.**
`OrgDb`	OrgDb
`keyType`	keytype of input gene
`ont`	One of "BP", "MF", and "CC" subontologies, or "ALL" for all three.
`pvalueCutoff`	adjusted pvalue cutoff on enrichment tests to report
`pAdjustMethod`	one of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"
`universe`	background genes. If missing, the all genes listed in the database (eg TERM2GENE table) will be used as background.
`qvalueCutoff`	qvalue cutoff on enrichment tests to report as significant. Tests must pass i) `pvalueCutoff` on unadjusted pvalues, ii) `pvalueCutoff` on adjusted pvalues and iii) `qvalueCutoff` on qvalues to be reported.
`minGSSize`	minimal size of genes annotated by Ontology term for testing.
`maxGSSize`	maximal size of genes annotated for testing
`readable`	whether mapping gene ID to gene Name
`pool`	If ont='ALL', whether pool 3 GO sub-ontologies

Example:

```
ego <- enrichGO(gene          = gene,
                universe      = names(geneList),
                OrgDb         = org.Hs.eg.db,
                ont           = "CC",
                pAdjustMethod = "BH",
                pvalueCutoff  = 0.01,
                qvalueCutoff  = 0.05,
                 readable      = TRUE)
head(ego,3)
```

```
##                    ID                                Description GeneRatio
## GO:0005819 GO:0005819                                    spindle    26/201
## GO:0000779 GO:0000779 condensed chromosome, centromeric region    16/201
## GO:0072686 GO:0072686                             mitotic spindle    17/201
##              BgRatio       pvalue     p.adjust       qvalue
## GO:0005819 306/11853 1.072029e-11 3.151766e-09 2.888837e-09
## GO:0000779 114/11853 7.709944e-11 8.659125e-09 7.936756e-09
## GO:0072686 133/11853 8.835841e-11 8.659125e-09 7.936756e-09
##
geneID
## GO:0005819
CDCA8/CDC20/KIF23/CENPE/ASPM/DLGAP5/SKA1/NUSAP1/TPX2/TACC3/NEK2/CDK1/MAD2L1/KIF18A
/BIRC5/KIF11/TRAT1/TTK/AURKB/PRC1/KIFC1/KIF18B/KIF20A/AURKA/CCNB1/KIF4A
## GO:0000779
CENPE/NDC80/HJURP/SKA1/NEK2/CENPM/CENPN/ERCC6L/MAD2L1/KIF18A/CDT1/BIRC5/TTK/NCAPG/
AURKB/CCNB1
## GO:0072686
KIF23/CENPE/ASPM/SKA1/NUSAP1/TPX2/TACC3/CDK1/MAD2L1/KIF18A/KIF11/TRAT1/AURKB/PRC1/
KIFC1/KIF18B/AURKA
##            Count
## GO:0005819    26
## GO:0000779    16
## GO:0072686    17
```

GO Gene Set Enrichment Analysis

The clusterProfiler package provides the gseGO() function for gene set enrichment analysis using gene ontology.

gseGO() arguments:

`geneList`	order ranked geneList
`ont`	one of "BP", "MF", and "CC" subontologies, or "ALL" for all three.
`OrgDb`	OrgDb
`keyType`	keytype of gene
`exponent`	weight of each step
`minGSSize`	minimal size of each geneSet for analyzing
`maxGSSize`	maximal size of genes annotated for testing
`eps`	This parameter sets the boundary for calculating the p value.
`pvalueCutoff`	pvalue Cutoff
`pAdjustMethod`	pvalue adjustment method
`verbose`	print message or not
`seed`	logical
`by`	one of 'fgsea' or 'DOSE'
`...`	other parameter

```
ego3 <- gseGO(geneList     = geneList,
```

```
              OrgDb         = org.Hs.eg.db,
              ont           = "CC",
              minGSSize     = 100,
              maxGSSize     = 500,
              pvalueCutoff  = 0.05,
              verbose       = FALSE)
```

2.5 Visualization of functional enrichment result

The enrichplot package implements several visualization methods to help interpreting enrichment results. It supports visualizing enrichment results obtained from DOSE (Yu et al. 2015), clusterProfiler (Yu et al. 2012; Wu et al. 2021), ReactomePA (Yu and He 2016) and meshes (Yu 2018). Both over representation analysis (ORA) and gene set enrichment analysis (GSEA) are supported.

Visualize enriched GO terms as a directed acyclic graph

The goplot() function is to plot induced GO DAG of significant terms. It can accept the output of enrichGO and visualize the enriched GO induced graph.

```
goplot(ego)
```

output:

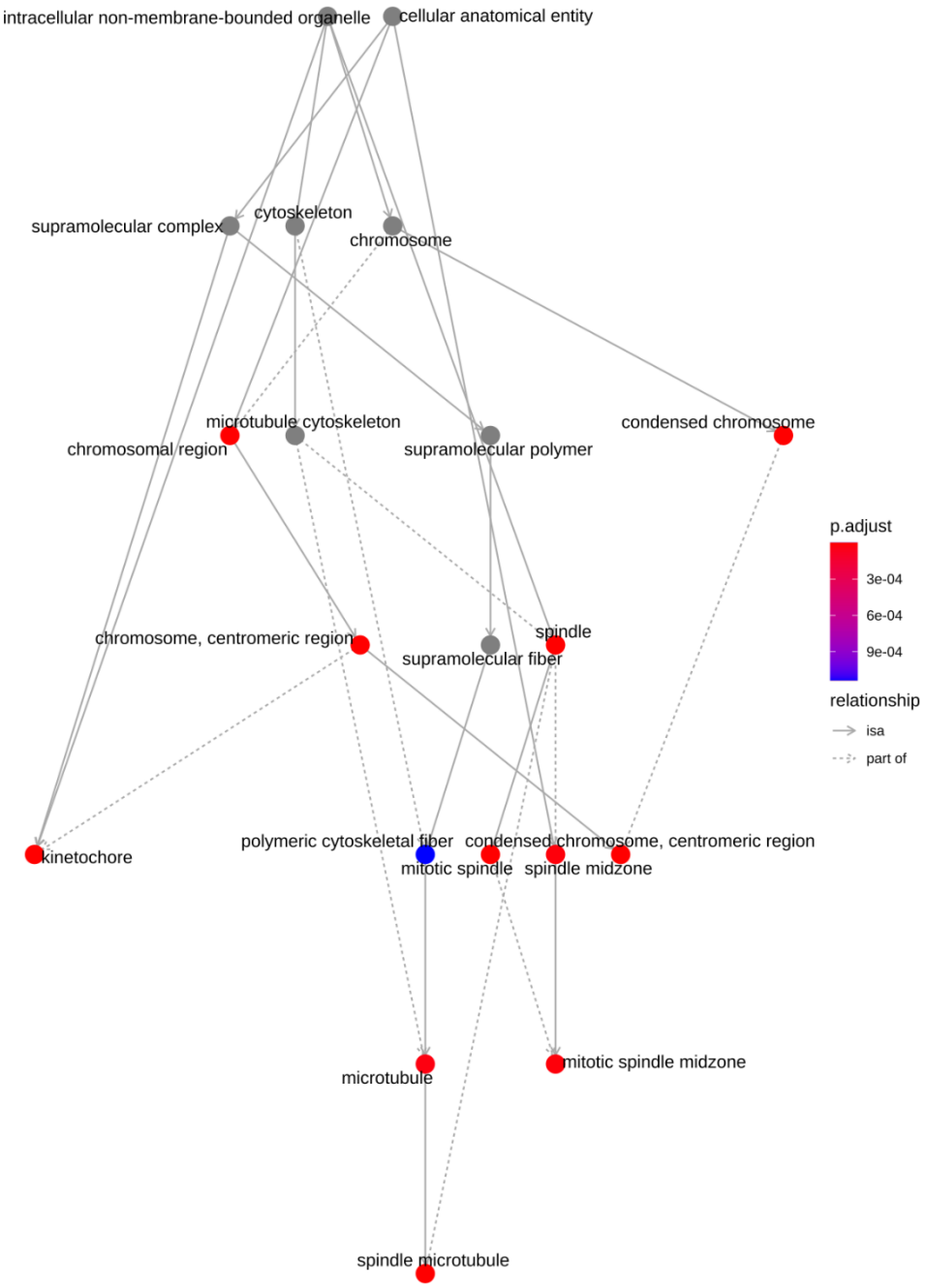

Figure 1: Goplot of enrichment analysis.

For more function, please refer to

https://rdrr.io/bioc/enrichplot/man/goplot.html

Bar Plot

Bar plot is the most widely used method to visualize enriched terms. It depicts the enrichment scores (e.g. p values) and gene count or ratio as bar height and color (Figure 2A). Users can specify the number of terms (most significant) or selected terms (see FAQ) to display via the showCategory parameter.

Example:

```
library(DOSE)
edo <- enrichDGN(gene)
library(enrichplot)
barplot(edo, showCategory=20)
```

Other variables that derived using mutate can also be used as bar height or color as demonstrated in Figure 2B.

```
mutate(edo, qscore = -log(p.adjust, base=10)) %>%
    barplot(x="qscore")
```

Output:

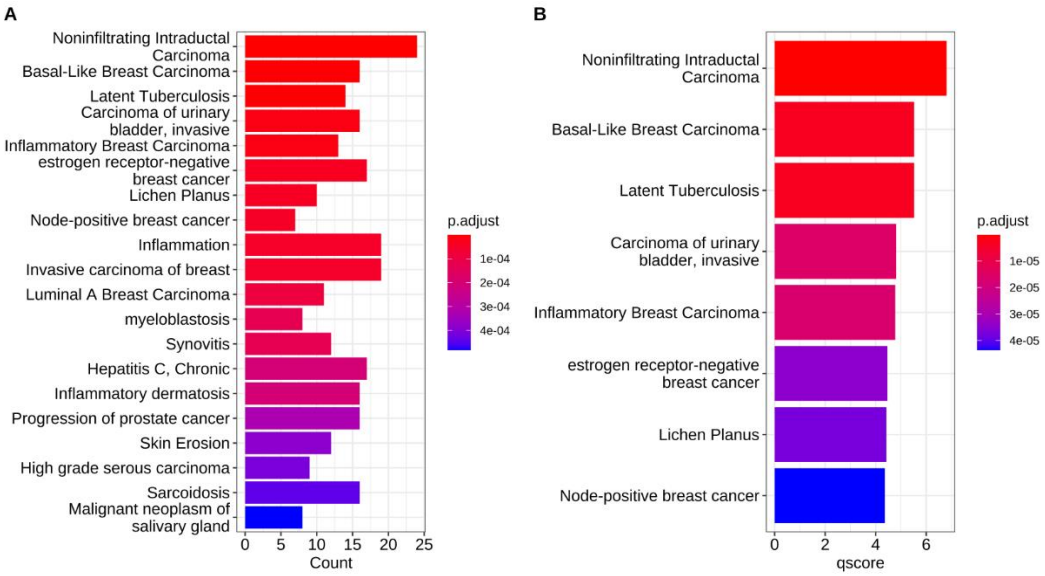

Figure 2: Bar plot of enriched terms.

Dot plot

Dot plot is similar to bar plot with the capability to encode another score as dot size.

```
edo2 <- gseDO(geneList)
dotplot(edo, showCategory=30) + ggtitle("dotplot for ORA")
dotplot(edo2, showCategory=30) + ggtitle("dotplot for GSEA")
```

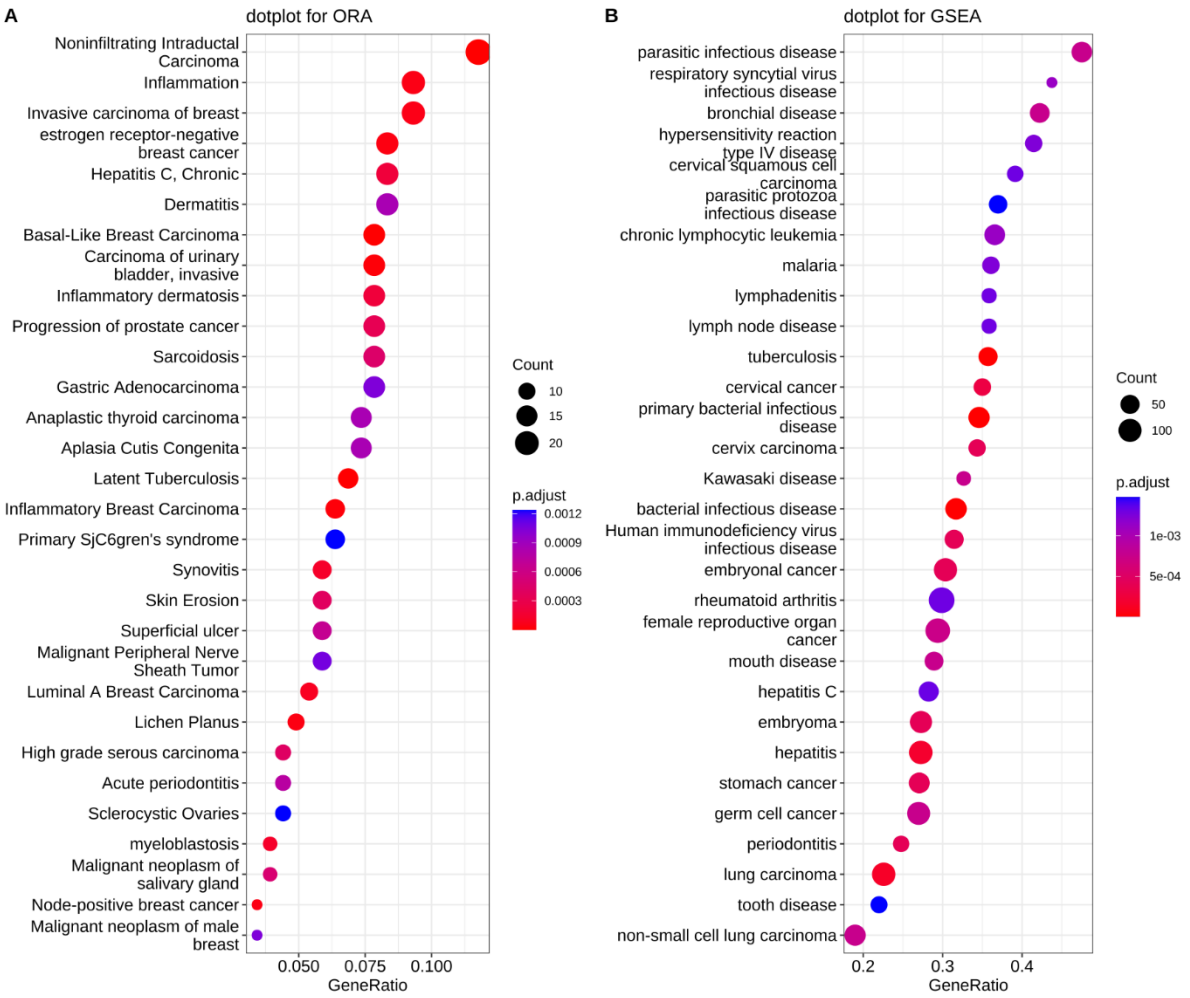

Figure 3: Dot plot of enriched terms.

Showing specific pathways

By default, all the visualization methods provided by enrichplot display most significant pathways. If users are interested to show some specific pathways (e.g. excluding some unimportant pathways among the top categories), users can pass a vector of selected pathways to the showCategory parameter in dotplot(), barplot(), treeplot(), cnetplot() and emapplot() etc.

```
library(DOSE)
library(enrichplot)
de <- names(geneList)[1:100]

x <- enrichDO(de)

## show top 10 most significant pathways and want to exclude the second one
## dotplot(x, showCategory = x$Description[1:10][-2])

set.seed(2020-10-27)
selected_pathways <- sample(x$Description, 10)
selected_pathways
##  [1] "in situ carcinoma"
##  [2] "hypersensitivity reaction type IV disease"
```

```
##  [3] "lymph node disease"
##  [4] "esophageal carcinoma"
##  [5] "atopic dermatitis"
##  [6] "pulmonary sarcoidosis"
##  [7] "esophageal cancer"
##  [8] "Human immunodeficiency virus infectious disease"
##  [9] "ovarian cancer"
## [10] "breast carcinoma"
p1 <- dotplot(x, showCategory = 10, font.size=14)
p2 <- dotplot(x, showCategory = selected_pathways, font.size=14)


cowplot::plot_grid(p1, p2, labels=LETTERS[1:2])
```

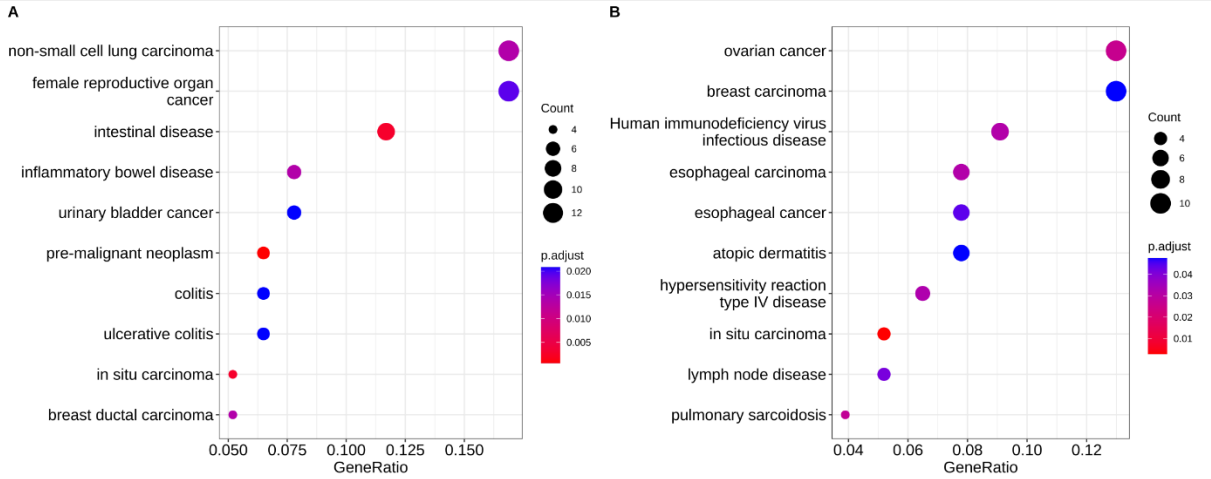

Figure 4: Showing specific pathways. Top ten most significant pathways (A), selected ten pathways (B).

Enrichment Map

Enrichment map organizes enriched terms into a network with edges connecting overlapping gene sets. In this way, mutually overlapping gene sets are tend to cluster together, making it easy to identify functional module.

The emapplot() function supports results obtained from hypergeometric test and gene set enrichment analysis. The cex_category parameter can be used to resize nodes, as demonstrated in Figure 5B, and the layout parameter can adjust the layout, as demonstrated in Figure 5C and D.

Example:

```
edo <- pairwise_termsim(edo)
p1 <- emapplot(edo)
p2 <- emapplot(edo, cex_category=1.5)
p3 <- emapplot(edo, layout="kk")
p4 <- emapplot(edo, cex_category=1.5,layout="kk")
cowplot::plot_grid(p1, p2, p3, p4, ncol=2, labels=LETTERS[1:4])
```

Output:

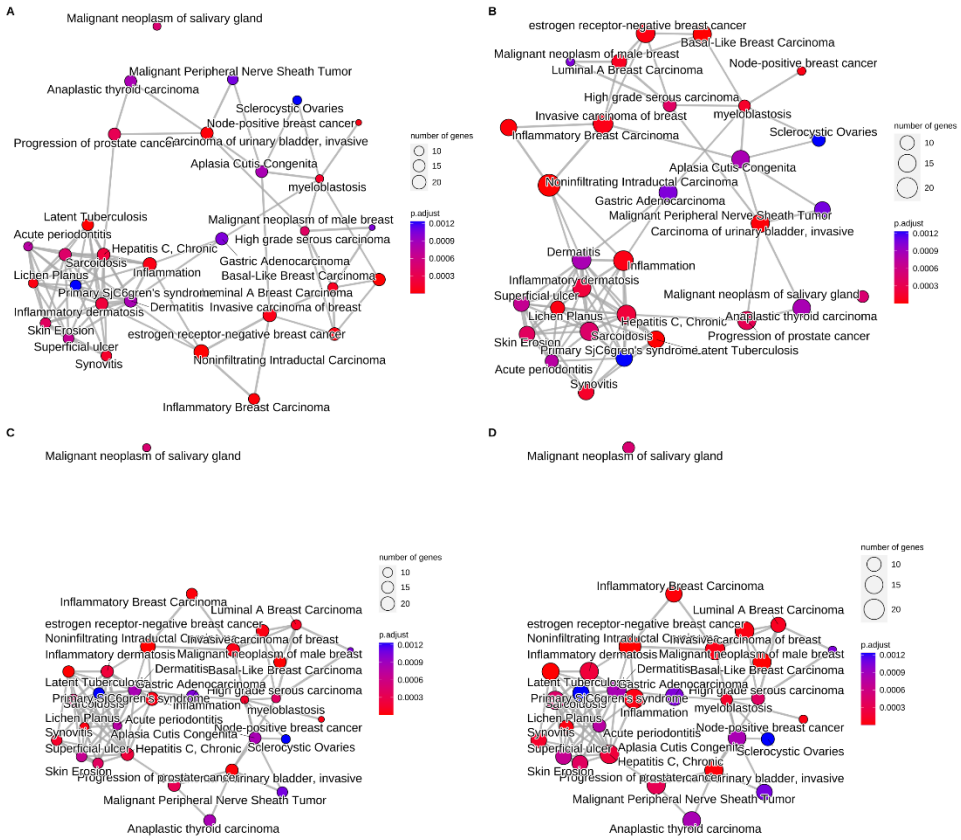

Figure 5: Plot for results obtained from hypergeometric test and gene set enrichment analysis. default (A), cex_category=1.5 (B), layout="kk" (C) and cex_category=1.5,layout="kk" (D).

For more visualization method, you can find the instruction in:

https://yulab-smu.top/biomedical-knowledge-mining-book/enrichplot.html

3 KEGG pathway analysis

The clusterProfiler package supports downloading the latest online version of KEGG data using the KEGG website, which is freely available for academic users. Both the KEGG pathway and module are supported in clusterProfiler.

3.1 Supported organisms

The clusterProfiler package supports all organisms that have KEGG annotation data available in the KEGG database. Users should pass an abbreviation of academic name to the organism parameter. The full list of KEGG supported organisms can be accessed via http://www.genome.jp/kegg/catalog/org_list.html

KEGG Orthology (KO) Database is also supported by specifying organism = "ko".

The clusterProfiler package provides search_kegg_organism() function to help searching supported organisms.

```
library(clusterProfiler)
search_kegg_organism('ece', by='kegg_code')
##      kegg_code                 scientific_name          common_name
## 611        ece Escherichia coli O157:H7 EDL933                 EHEC
## 4519      ecec            Enterococcus cecorum Enterococcus cecorum
ecoli <- search_kegg_organism('Escherichia coli', by='scientific_name')
dim(ecoli)
## [1] 65  3
head(ecoli,3)
##     kegg_code                 scientific_name                 common_name
## 606       eco    Escherichia coli K-12 MG1655 Escherichia coli K-12 MG1655
## 607       ecj     Escherichia coli K-12 W3110  Escherichia coli K-12 W3110
## 608       ecd     Escherichia coli K-12 DH10B  Escherichia coli K-12 DH10B
```

3.2 Run KEGG pathway analysis

KEGG pathway over-representation analysis

enrichKEGG() function is used to do KEGG Enrichment Analysis of a gene set. Given a vector of genes, this function will return the enrichment KEGG categories with FDR control.

enrichKEGG() arguments:

`gene`	a vector of entrez gene id.
`organism`	supported organism listed in 'http://www.genome.jp/kegg/catalog/org_list.html'
`keyType`	one of "kegg", 'ncbi-geneid', 'ncib-proteinid' and 'uniprot'
`pvalueCutoff`	adjusted pvalue cutoff on enrichment tests to report
`pAdjustMethod`	one of "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr", "none"
`universe`	background genes. If missing, the all genes listed in the database (eg TERM2GENE table) will be used as background.
`minGSSize`	minimal size of genes annotated by Ontology term for testing.
`maxGSSize`	maximal size of genes annotated for testing

`qvalueCutoff`	qvalue cutoff on enrichment tests to report as significant. Tests must pass i) `pvalueCutoff` on unadjusted pvalues, ii) `pvalueCutoff` on adjusted pvalues and iii) `qvalueCutoff` on qvalues to be reported.
`use_internal_data`	logical, use KEGG.db or latest online KEGG data

Input ID type can be kegg, ncbi-geneid, ncbi-proteinid or uniprot (converting biological). Unlike enrichGO(), there is no readable parameter for enrichKEGG(). However, users can use the setReadable() function if there is an OrgDb available for the species.

R script:

```
## KEGG pathway over-representation analysis
## following GO analysis, genelist preparation is same as in 2.3

kk <- enrichKEGG(gene         = gene,
                 organism     = 'hsa',
                 pvalueCutoff = 0.05)
head(kk,3)


# Output:
##                ID                                                 Description
## hsa04110 hsa04110                                                  Cell cycle
## hsa04114 hsa04114                                             Oocyte meiosis
## hsa04218 hsa04218                                       Cellular senescence
##          GeneRatio  BgRatio       pvalue     p.adjust       qvalue
## hsa04110     11/94 126/8113 1.895271e-07 0.0000392321 3.850392e-05
## hsa04114     10/94 131/8113 2.444436e-06 0.0002529992 2.483033e-04
## hsa04218     10/94 156/8113 1.171114e-05 0.0008080690 7.930705e-04
##                                                    geneID Count
## hsa04110 8318/991/9133/890/983/4085/7272/1111/891/4174/9232    11
## hsa04114    991/9133/983/4085/51806/6790/891/9232/3708/5241    10
## hsa04218     2305/4605/9133/890/983/51806/1111/891/776/3708    10
```

KEGG pathway gene set enrichment analysis

gseKEGG() is for Gene Set Enrichment Analysis of KEGG.

Arguments:

`geneList`	order ranked geneList
`organism`	supported organism listed in 'http://www.genome.jp/kegg/catalog/org_list.html'
`keyType`	one of "kegg", 'ncbi-geneid', 'ncib-proteinid' and 'uniprot'
`exponent`	weight of each step
`minGSSize`	minimal size of each geneSet for analyzing
`maxGSSize`	maximal size of genes annotated for testing
`eps`	This parameter sets the boundary for calculating the p value.

`pvalueCutoff`	pvalue Cutoff
`pAdjustMethod`	pvalue adjustment method
`verbose`	print message or not
`use_internal_data`	logical, use KEGG.db or latest online KEGG data
`seed`	logical
`by`	one of 'fgsea' or 'DOSE'
`...`	other parameter

Example:

```
kk2 <- gseKEGG(geneList     = geneList,
               organism     = 'hsa',
               minGSSize    = 120,
               pvalueCutoff = 0.05,
               verbose      = FALSE)
head(kk2,3)
```

```
##                ID                                Description setSize
## hsa05169 hsa05169            Epstein-Barr virus infection     193
## hsa05166 hsa05166 Human T-cell leukemia virus 1 infection     202
## hsa04613 hsa04613 Neutrophil extracellular trap formation     130
##          enrichmentScore       NES       pvalue     p.adjust      qvalues rank
## hsa05169       0.4335010  1.930356 1.485802e-07 1.144068e-05 7.820013e-06 2820
## hsa05166       0.3893613  1.744456 7.017747e-06 2.479262e-04 1.694643e-04 1955
## hsa04613       0.4496569  1.902675 9.659464e-06 2.479262e-04 1.694643e-04 2575
##                            leading_edge
## hsa05169 tags=39%, list=23%, signal=31%
## hsa05166 tags=26%, list=16%, signal=22%
## hsa04613 tags=37%, list=21%, signal=30%
##
core_enrichment
## hsa05169
3627/890/6890/9636/898/9134/6502/6772/3126/3112/4609/917/5709/1869/3654/919/915/40
67/4938/864/4940/5713/5336/11047/3066/54205/1871/578/1019/637/916/3383/4939/10213/
23586/4793/5603/7979/7128/6891/930/5714/3452/6850/5702/4794/7124/3569/7097/5708/22
08/8772/3119/5704/7186/5971/3135/1380/958/5610/4792/10018/8819/3134/10379/9641/114
7/5718/6300/3109/811/5606/2923/3108/5707/1432
## hsa05166
991/9133/890/4085/7850/1111/9232/8061/701/9700/898/4316/9134/3932/3559/3126/3112/4
609/3561/917/1869/1029/915/114/2005/5902/55697/1871/1031/2224/292/1019/3689/916/33
83/11200/706/3600/6513/3601/468/5604/7124/1030/3569/4049/4055/10393/3119/5901/5971
/1959/3135
## hsa04613
820/366/51311/64581/3015/85236/55506/8970/8357/1535/2359/5336/4688/92815/3066/8336
/292/1991/3689/8345/5603/4689/5880/10105/1184/6404/3018/6850/5604/3014/7097/1378/8
290/1536/834/5605/1183/728/2215/8335/5594/9734/3674/5578/5582/7417/8331/6300
```

Visualize enriched KEGG pathways

The enrichplot package implements several methods to visualize enriched terms. Most of them are general methods that can be used on GO, KEGG, MSigDb, and other gene set annotations. Here, we introduce the clusterProfiler::browseKEGG() and pathview::pathview() functions to help users explore enriched KEGG pathways with genes of interest.

To view the KEGG pathway, users can use the browseKEGG() function, which will open a web browser and highlight enriched genes.

Usage:

browseKEGG(x, pathID)

Arguments:

`x`	**an instance of enrichResult or gseaResult**
`pathID`	pathway ID

Example:

```
browseKEGG(kk, 'hsa04110')
```

Output:

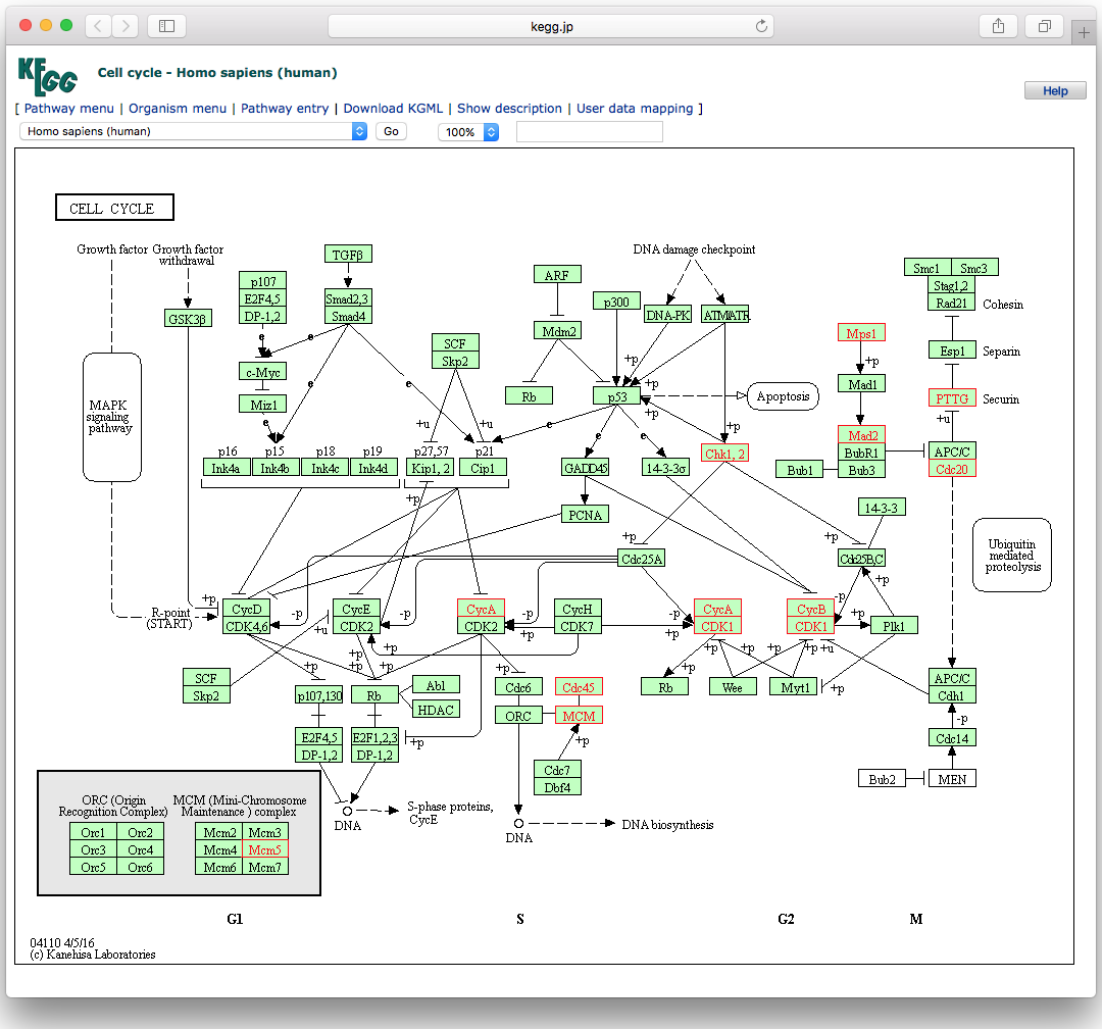

Figure 6. Explore selected KEGG pathway. Differentially expressed genes that are enriched in the selected pathway will be highlighted.

Users can also use the pathview() function from the pathview (Luo and Brouwer 2013) to visualize enriched KEGG pathways identified by the clusterProfiler package (Yu et al. 2012).

Pathview() is a tool set for pathway-based data integration and visualization. It maps and renders user data on relevant pathway graphs. All users need is to supply their gene or compound data and specify the target pathway. Pathview() automatically downloads the pathway graph data, parses the data file, maps user data to the pathway, and render pathway graph with the mapped data. Pathview() generates both native KEGG view and Graphviz views for pathways. keggview.native and keggview.graph are the two viewer functions, and pathview is the main function providing a unified interface to downloader, parser, mapper and viewer functions.

The following example illustrates how to visualize the “hsa04110” pathway, which was enriched in our previous analysis.

```
library("pathview")
hsa04110 <- pathview(gene.data  = geneList,
                     pathway.id = "hsa04110",
                     species    = "hsa",
                     limit      = list(gene=max(abs(geneList)), cpd=1))
```

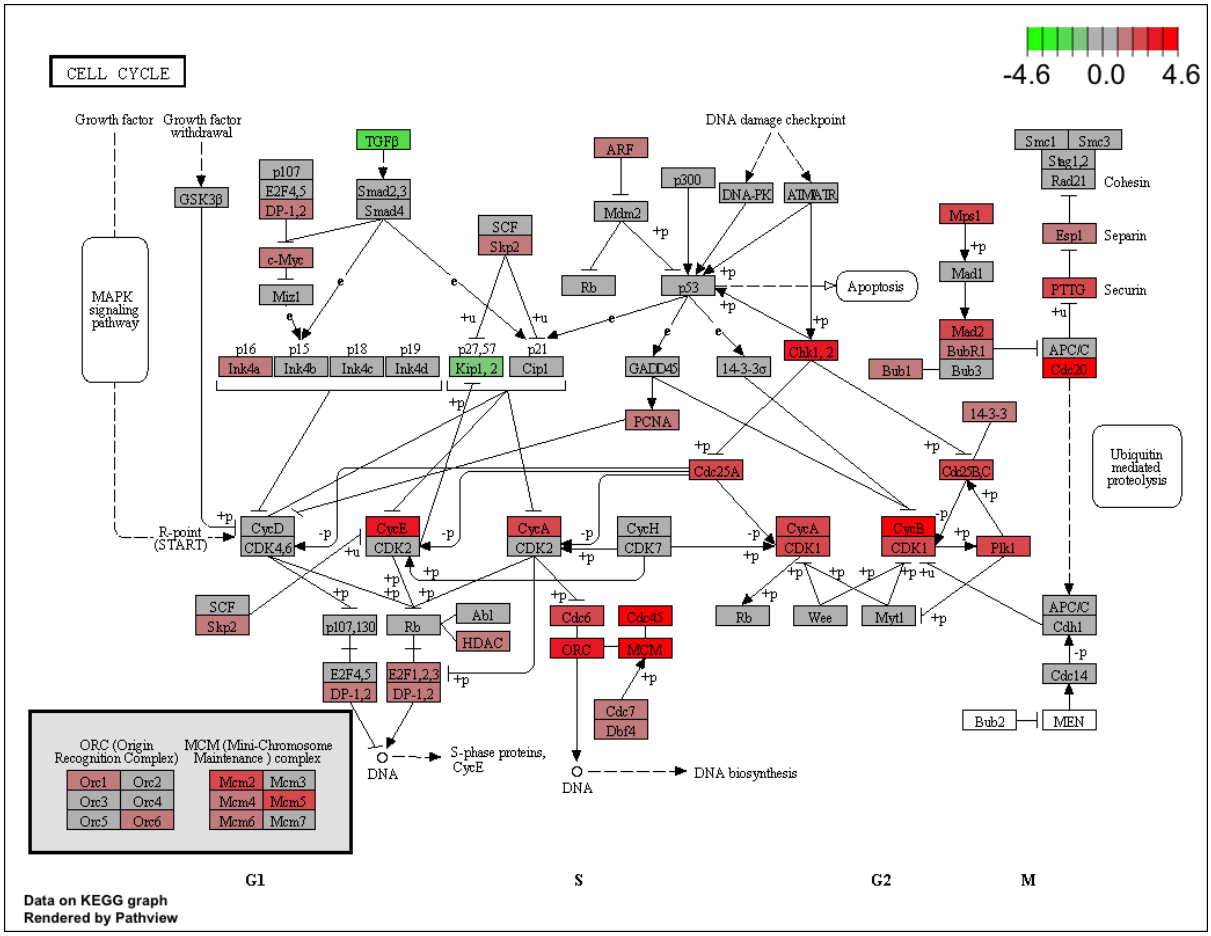

Figure 7. Visualize selected KEGG pathway by pathview(). Gene expression values can be mapped to gradient color scale.

For more setting please refer to pathview() function instruction:

https://rdrr.io/bioc/pathview/man/pathview.html

The clusterProfiler also provides many types of analysis, such as Disease enrichment(DO) analysis, MeSH enrichment analysis and all types of visualization methods. You can find the instruction in:

https://yulab-smu.top/biomedical-knowledge-mining-book/index.html

-End-