LAB 4 – GENE EXPRESSION DATA ANALYSES

Objectives

By the end of this practical, you will know:

- How to select significant genes from a set of high throughput data
- How to cluster genes/samples with similar expression profiles
- How to organize genes according to gene ontology categories and biological pathway

Problem Scenario

Liver hepatocellular carcinoma (LIHC) is the main manifestation of primary liver cancer, with low survival rate and poor prognosis. Medical decision-making process of LIHC is so complex that new biomarkers for diagnosis and prognosis have yet to be explored. In this exercise, we aim to identify the genes involved in the pathophysiology of LIHC and make sense of them.

A table of RNA-seq read counts for 374 tumor and 50 normal tissue samples were retrieved from the Cancer Genome Atlas (TCGA) and are available on Learningmall (LIHC_counts_lab4.csv), with each row representing a single gene and each column representing a single sample.

1. Selecting Significant Genes

A common objective in gene expression profiling experiments is to identify those genes that exhibit differential expression (an increased or decreased steady state abundance) under a particular treatment relative to the expression levels of those genes in the control samples. In the simplest case, this has often involved the identification of genes exhibiting a relative fold change greater than some arbitrary cutoff, for example, say 2-fold or greater expression difference in the treatment gene's expression level relative to its level in the control. We can't really generate any sort of p-value for such a calculation. A more statistically sound approach is to use one of the several packages available, such as *limma* or *edgeR*, to carry out such analyses. We'll use both *limma* and *edgeR* here, which use a negative binomial distribution along with the appropriate statistical tests.

[1] Prepare R packages

```
chooseCRANmirror()

install.packages('stringr')
install.packages('ggplotify')
install.packages('patchwork')
install.packages('cowplot')

install.packages("BiocManager")
BiocManager::install(c("edgeR", "limma")
# Install edgeR and limma packages for Bioconductor
```

[2] Set work path, read data and data statistics

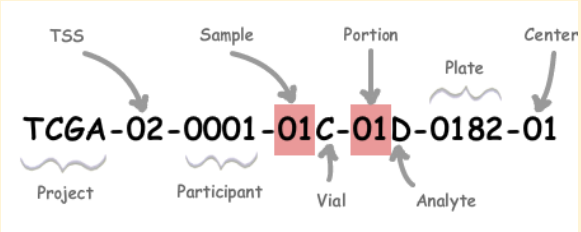
```
> path<-'/Users/xin.liu/Desktop/BIO211/Lab 4'
> setwd(paste(path))
# Change the work path accordingly
# For example, path<-‘C://BIO211/LAB4’

> rm(list=ls())
> a=read.csv('LIHC_counts_lab4.csv')
> dim(a)
[1] 60488      425

> row.names(a)<-a[,1]
> a<-a[,-1]
> group_list<-ifelse(as.numeric(substr(colnames(a),14,15))<10,'tumor','normal')
# Separate samples into two groups
# Check the hints on TCGA Barcode for how to differentiate tumor/normal samples

> group_list<-factor(group_list,levels=c("normal","tumor"))
> table(group_list)
group_list
normal    tumor
      50      374

> View(a)
# Check the data matrix
```

Data: a

	row.names	TCGA.DD.A3A4.11A.11R.A22L.07	TCGA.UB.A7MB.01A.11R.A33R.07	TCGA.RG.A7D4.01A.12R.A33R.07	TCGA.DD.AACD.01A.11R.A41C.07	TCGA.DD.AAEG.01A.11R.A39D.07	TCGA.ED.A82E.01A.11R.A352.07
1	ENSG00000000003.13	2936	3733	6332	6637	2931	12247
2	ENSG00000000005.5	5	13	1	1	4	1
3	ENSG00000000419.11	1373	1891	1544	476	2458	2149
4	ENSG00000000457.12	636	1182	468	184	1568	1895
5	ENSG00000000460.15	129	796	567	91	252	188
6	ENSG00000000938.11	393	121	236	116	98	274
7	ENSG00000000971.14	44018	27845	746	33089	197813	2165
8	ENSG00000001036.12	3771	3125	4744	1423	2541	4097
9	ENSG00000001084.9	6373	6813	9364	2318	4722	1426
10	ENSG00000001167.13	702	1192	1474	350	791	2677
11	ENSG00000001460.16	138	192	167	64	272	239
12	ENSG00000001461.15	443	693	435	135	824	2263
13	ENSG00000001497.15	2189	2748	3120	3100	2060	2281
14	ENSG00000001561.6	381	1156	802	120	1112	503
15	ENSG00000001617.10	481	969	1029	448	617	1284
16	ENSG00000001626.13	158	209	1	1	63	53851
17	ENSG00000001629.8	714	1288	890	335	945	2502
18	ENSG00000001630.14	153	1288	215	76	436	429
19	ENSG00000001631.13	701	970	811	193	852	1326
20	ENSG00000002016.15	199	327	236	39	291	448
21	ENSG00000002079.11	0	15	3	1	3	5
22	ENSG00000002330.12	1646	2851	764	888	1973	4156
23	ENSG00000002549.11	9054	4790	7392	7028	5315	6953
24	ENSG00000002586.16	9419	12449	7969	5072	4244	5229
25	ENSG00000002587.8	56	14	27	29	121	598
26	ENSG00000002726.18	138	0	83	0	5	15
27	ENSG00000002745.11	0	51	3	0	1	1
28	ENSG00000002746.13	3	1	6	14	3	42
29	ENSG00000002822.14	850	1250	1120	579	1412	1229
30	ENSG00000002834.16	9092	10171	14323	3485	22651	16047
31	ENSG00000002919.13	957	911	901	474	759	656

Tiny Hints on TCGA Barcode

The TCGA barcode is the primary identifier of biospecimen data within the TCGA project. A TCGA barcode is composed of a collection of identifiers. Each specifically identifies a TCGA data element. Refer to the following figure for an illustration of how metadata identifiers comprise a barcode. A typical example is shown below.



Here, the ***Sample*** label is the identifier for sample type. Specifically, *tumor* types range from 01-09, *normal* types from 10-19 and *control* samples from 20-29. In the example above, the ***Sample*** label 01 means it is a solid tumor sample.

For more information about TCGA barcode, please refer to:

https://docs.gdc.cancer.gov/Encyclopedia/pages/TCGA_Barcode/

[3] Select significant genes exhibiting differential expression

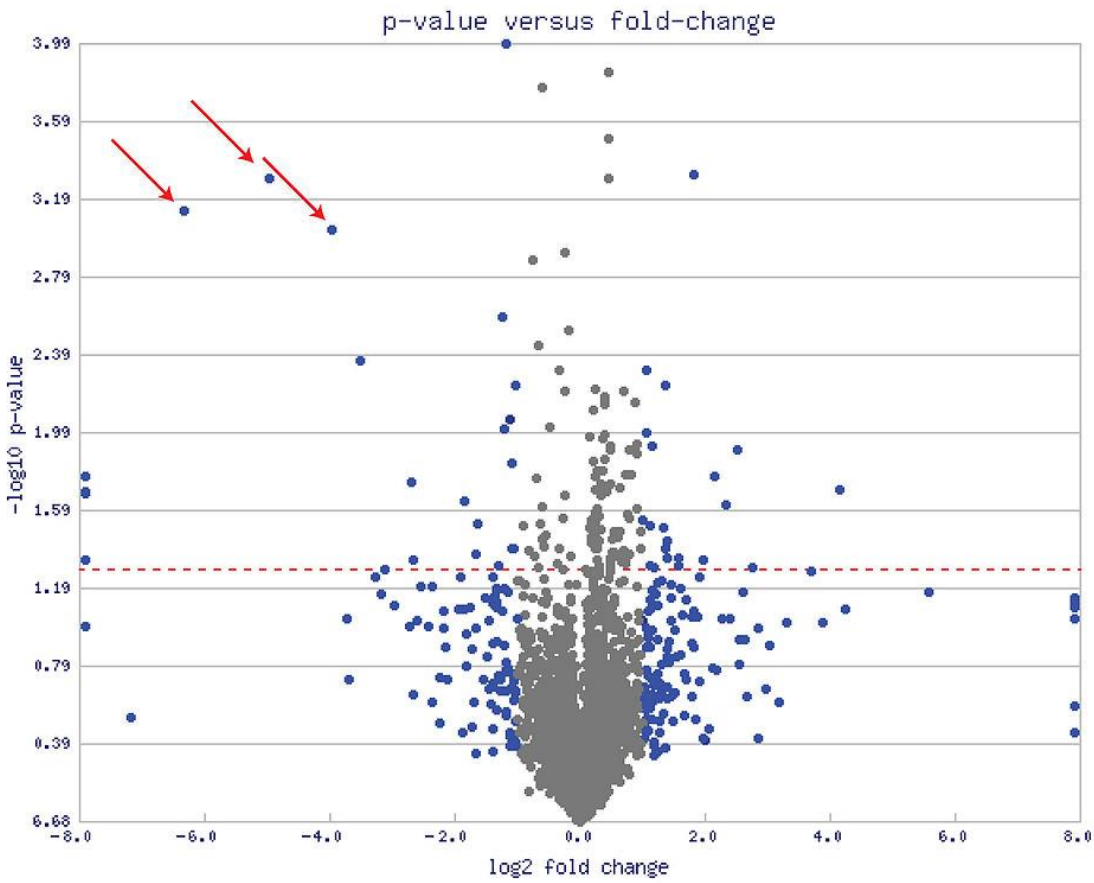
```
## Method 1: edgeR
> library(edgeR)
> dge<-DGEList(counts=a, group=group_list)
> dge$samples$lib.size<-colSums(dge$counts)
> design<-model.matrix(~0+group_list)
> rownames(design)<-colnames(dge)
> colnames(design)<-levels(group_list)
> dge<-estimateGLMCommonDisp(dge,design)
> dge<-estimateGLMTrendedDisp(dge,design)
> dge<-estimateGLMTagwiseDisp(dge,design)
> fit<-glmFit(dge,design)
> fit2<-glmLRT(fit,contrast=c(-1,1))
> DEG2=topTags(fit2, n=nrow(a))
> DEG2=as.data.frame(DEG2)
> logFC_cutoff2<-with(DEG2, mean(abs(logFC))+2*sd(abs(logFC)))
> DEG2$change=as.factor(
+       ifelse(DEG2$PValue<0.05 & abs(DEG2$logFC) > logFC_cutoff2,
+             ifelse(DEG2$logFC > logFC_cutoff2,"UP","DOWN"),"NOT")
+ )
> head(DEG2)
                        logFC    logCPM       LR       PValue          FDR change
ENSG00000130300.7    3.503338  5.990635 401.1044 3.166156e-89 1.915144e-84     UP
ENSG00000136011.13  -4.257759  1.393196 381.5993 5.581837e-85 1.688171e-80   DOWN
ENSG00000187730.7    5.128670  1.307227 367.9976 5.107250e-82 1.029758e-77     UP
ENSG00000147113.15   2.842924  2.652984 345.6887 3.681279e-77 5.566829e-73    NOT
ENSG00000131097.5    3.734560 -0.471801 320.7663 9.862482e-72 1.193124e-67     UP
ENSG00000158882.11   2.380159  4.453545 319.6380 1.736840e-71 1.750966e-67    NOT
> table(DEG2$change)

 DOWN   NOT    UP
   30 57794  2664


## Method 2: limma-voom
> library(limma)
> design<-model.matrix(~0+group_list)
> colnames(design)=levels(group_list)
> rownames(design)=colnames(a)
> dge<-DGEList(counts=a)
> dge<-calcNormFactors(dge)
> logCPM<-cpm(dge,log=TRUE,prior.count=3)
> v<-voom(dge,design,normalize="quantile")
> fit<-lmFit(v,design)
> constra=paste(rev(levels(group_list)),collapse="-")
> cont.matrix<-makeContrasts(contrasts=constra,levels=design)
> fit3=contrasts.fit(fit,cont.matrix)
> fit3=eBayes(fit3)
> DEG3=topTable(fit3,coef=constra,n=Inf)
> DEG3=na.omit(DEG3)
> logFC_cutoff3<-with(DEG3,mean(abs(logFC)+2*sd(abs(logFC))))
> DEG3$change=as.factor(
+       ifelse(DEG3$P.Value<0.05 & abs(DEG3$logFC) > logFC_cutoff3,
+             ifelse(DEG3$logFC > logFC_cutoff3,"UP","DOWN"),"NOT")
+ )
```

```
> head(DEG3)
                        logFC   AveExpr         t       P.Value     adj.P.Val        B change
ENSG00000182566.11 -8.239692 -1.820630 -41.22539 3.463472e-150 2.094985e-145 331.4897   DOWN
ENSG00000104938.15 -9.135127 -3.924408 -40.46522 1.848373e-147 5.590218e-143 324.7178   DOWN
ENSG00000165682.13 -8.258450 -3.711571 -36.40327 2.299505e-132 4.636416e-128 290.3370   DOWN
ENSG00000279204.1   4.095732 -3.033125  35.65719 1.702995e-129 2.575269e-125 284.6265     UP
ENSG00000136011.13 -6.267465 -1.296913 -34.65907 1.310918e-125 1.585896e-121 275.3950   DOWN
ENSG00000263761.2  -8.837758 -4.275124 -34.50255 5.394090e-125 5.437962e-121 273.6863   DOWN
> table(DEG3$change)
# Count the number of up- and down-regulated genes

 DOWN     NOT      UP
 1693 57644    1151

## Save your results
> edgeR_DEG<-DEG2
> limma_voom_DEG<-DEG3
> save(edgeR_DEG,limma_voom_DEG,group_list,file='DEG.Rdata')
```

2. Data Visualization by Volcano Plot

In statistics, a volcano plot is a type of scatter-plot that is used to quickly identify changes in large data sets composed of replicate data. It plots significance versus fold-change on the y and x axes, respectively. These plots are increasingly common in omics experiments such as genomics, proteomics, and metabolomics where one often has a list of many thousands of replicate data points between two conditions and one wishes to quickly identify the most meaningful changes. A volcano plot combines a measure of statistical significance from a statistical test (e.g., a P-value from an ANOVA model) with the magnitude of the change, enabling quick visual identification of those data-points (genes, etc.) that display large magnitude changes that are also statistically significant.

A volcano plot (an example is shown below) is constructed by plotting the negative log of the P-value on the y axis (usually base 10). This results in data points with low P-values (highly significant) appearing toward the top of the plot. The x axis is the log of the fold change between the two conditions. The log of the fold change is used so that changes in both directions appear equidistant from the center. Plotting points in this way results in two regions of interest in the plot: those points that are found toward the top of the plot that are far to either the left- or right-hand sides. These represent values that display large magnitude fold changes (hence being left or right of center) as well as high statistical significance (hence being toward the top).



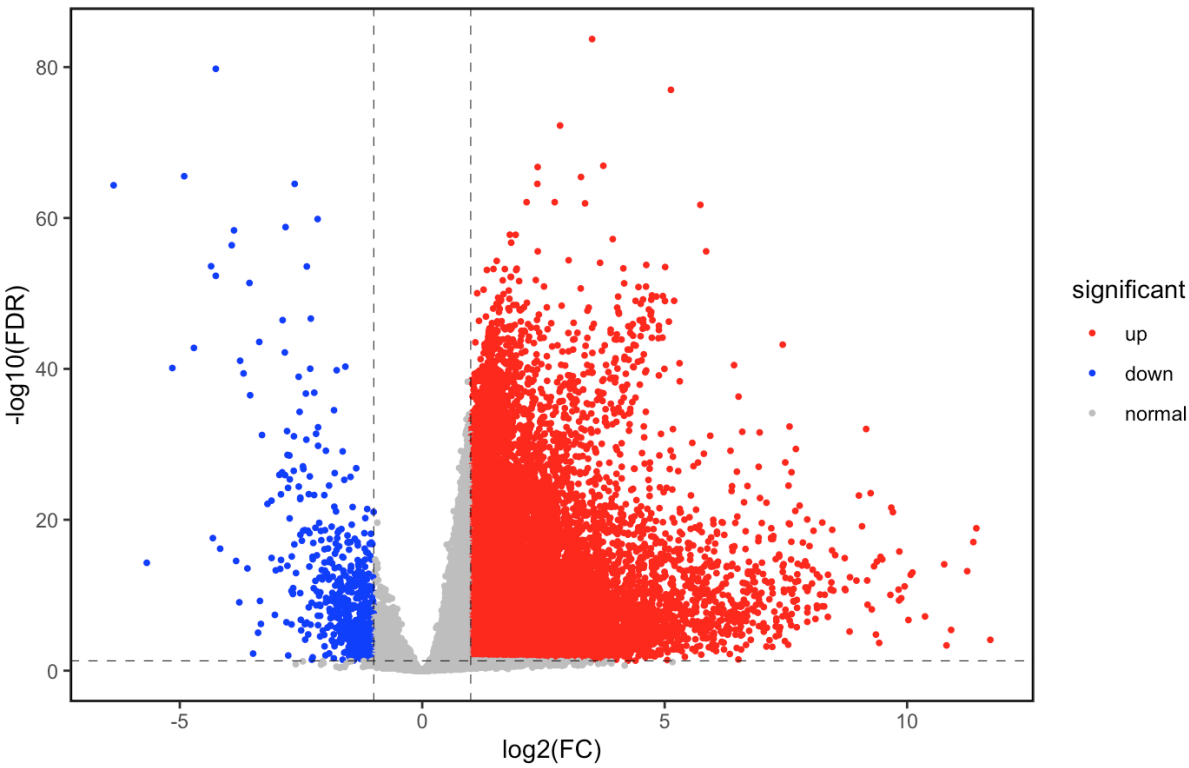
This is a volcano plot presenting a set of metabolomic data. The red arrows indicate points-of-interest that display both large magnitude fold-changes (x axis) and high statistical significance (-log10 of p value, y axis). The dashed red line shows where p = 0.05 with points above the line having p < 0.05 and points below the line having p > 0.05. This plot is colored such that those points having a fold-change less than 2 (log2 = 1) are shown in gray.

[1] Data preparation

```
> library(stringr)
> rownames(DEG2)<-str_sub(rownames(DEG2),start=1,end=15)
> DEG2$ENSEMBL<-rownames(DEG2)
> diff_gene<-DEG2[DEG2$change!='NOT',]
> write.table(diff_gene,
+ file='/Users/xin.liu/Desktop/BIO211/Lab 4/LIHC_diff_gene.txt',
+ sep='\t',quote=F)
# Change the path accordingly
```

[2] Construct a volcano plot

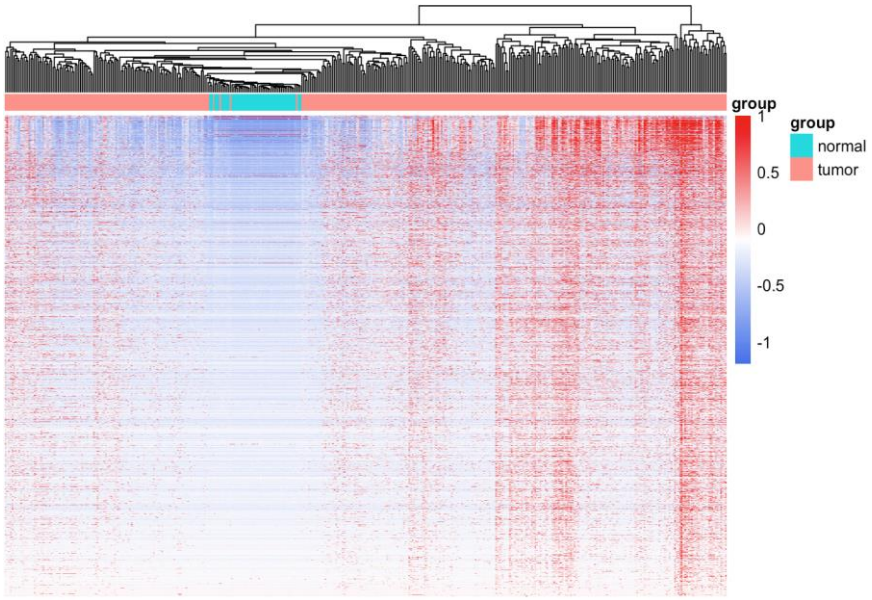
```
> install.packages('ggplot2')
> library(cowplot)
> library(patchwork)
> library(ggplotify)
> library(ggplot2)

> loc_up<-intersect(which(DEG2$FDR<0.05),which(DEG2$logFC>=1))
> loc_down<-intersect(which(DEG2$FDR<0.05),which(DEG2$logFC<(-1)))
> significant<-rep('normal',times=nrow(DEG2))
> significant[loc_up]<-'up'
> significant[loc_down]<-'down'
> significant<-factor(significant,levels=c('up','down','normal'))
> p<-qplot(x=DEG2$logFC,y=-log10(DEG2$FDR),xlab='log2(FC)',ylab='-log10(FDR)',
+ size=I(0.7),colour=significant)
> p<-p+scale_color_manual(values=c('up'='red','normal'='grey','down'='blue'))

## Add cut-off lines & export image
> xline=c(-log2(2),log2(2))
> p<-p+geom_vline(xintercept = xline,lty=2,size=I(0.2),color='grey11')

> yline=-log(0.05,10)
> p<-p+geom_hline(yintercept = yline,lty=2,size=I(0.2),color='grey11')

> p<-p+theme_bw()+theme(panel.background = element_rect(colour = 'black',
+ size=1,fill='white'),panel.grid=element_blank())
> pdf('deg2_volcano.pdf')
> print(p)
> dev.off()
```



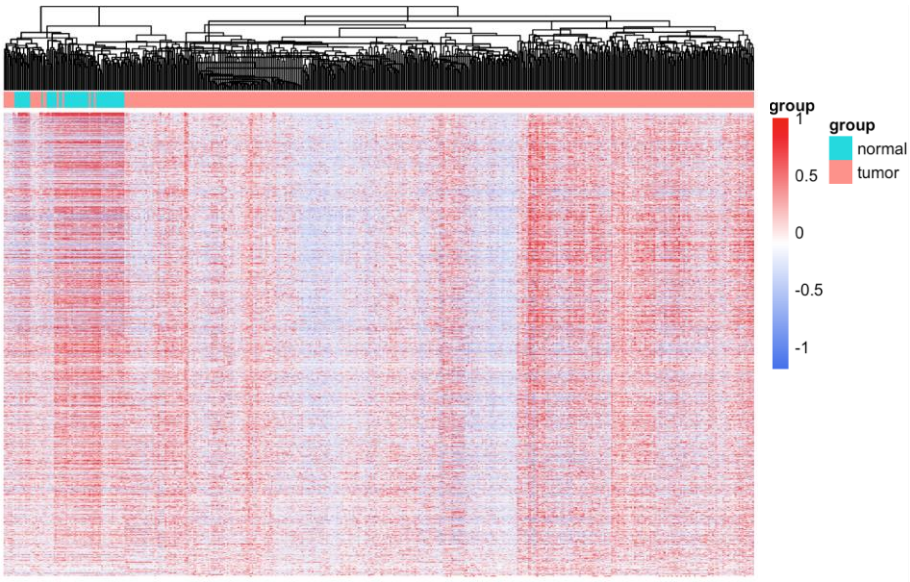
3. Data Organization by Hierarchical Clustering

There are a variety of algorithms for performing clustering analysis as it pertains to gene expression data. One of the most common methods is hierarchical clustering, which we will examine briefly in this section. The *heatmap()* method actually applies a default hierarchical clustering to expression data whenever it draws a heatmap, using Euclidean distance and complete linkage. This is not the ideal scenario for cluster analysis in BioConductor, as several sophisticated methods exist (e.g: *agnes()*, *diana()*, *hopach()*), but we will stick to the heatmap implementation for sake of example and simplicity. Let's examine the clustering features of heatmap by looking at gene-wise and sample-wise clustering.

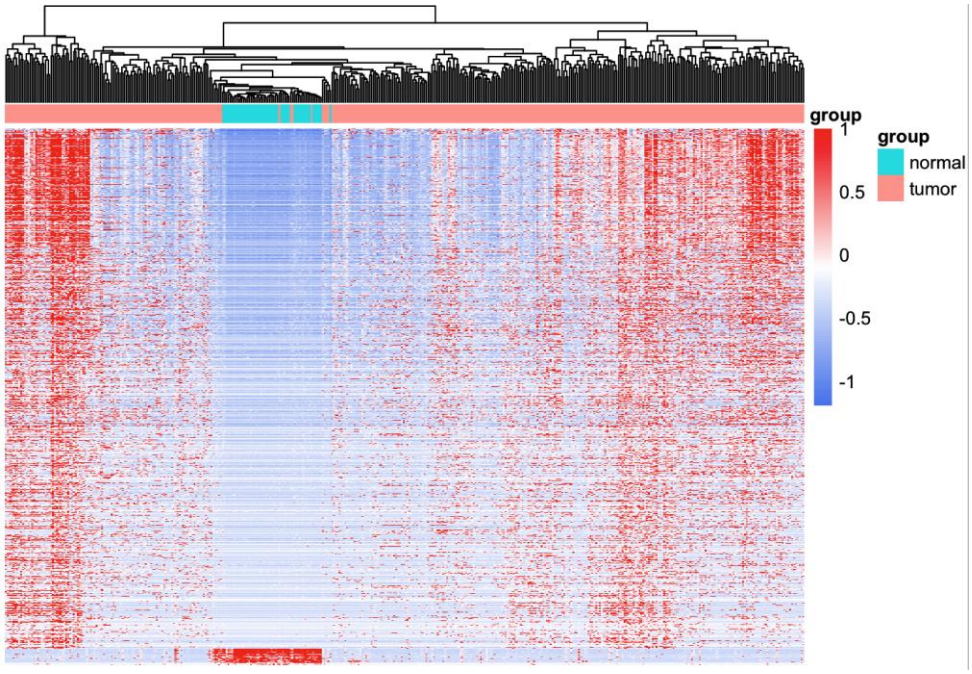
```
> install.packages('pheatmap')

> cg1=rownames(edgeR_DEG)[edgeR_DEG$change!='NOT']
> cg2=rownames(limma_voom_DEG)[limma_voom_DEG$change!='NOT']
> library(pheatmap)
> library(RColorBrewer)
> color<-colorRampPalette(c('#436EEE','white','#EE0000'))(100)

## Hierarchical clustering on edgeR produced significant genes
> mat1=a[cg1,]
> n1=t(scale(t(mat1)))
> n1[n1>1]=1
> n1[n1<-1]=-1
> ac=data.frame(group=group_list)
> rownames(ac)=colnames(mat1)
> ht1<-pheatmap(n1,show_rownames = F,show_colnames = F,
+         cluster_rows = F,cluster_cols = T,
+         annotation_col = ac, color = color)
> print(ht1)
> dev.off()
```



```
## Hierarchical clustering on limma-voom produced significant genes
> mat2=a[cg2,]
> n2=t(scale(t(mat2)))
> n2[n2>1]=1
> n2[n2<-1]=-1
> ac=data.frame(group=group_list)
> rownames(ac)=colnames(mat2)
> ht2=pheatmap(n2,show_rownames = F,show_colnames = F,
+        cluster_rows = F, cluster_cols = T,
+        annotation_col = ac, color = color)
> print(ht2)
> dev.off()
```



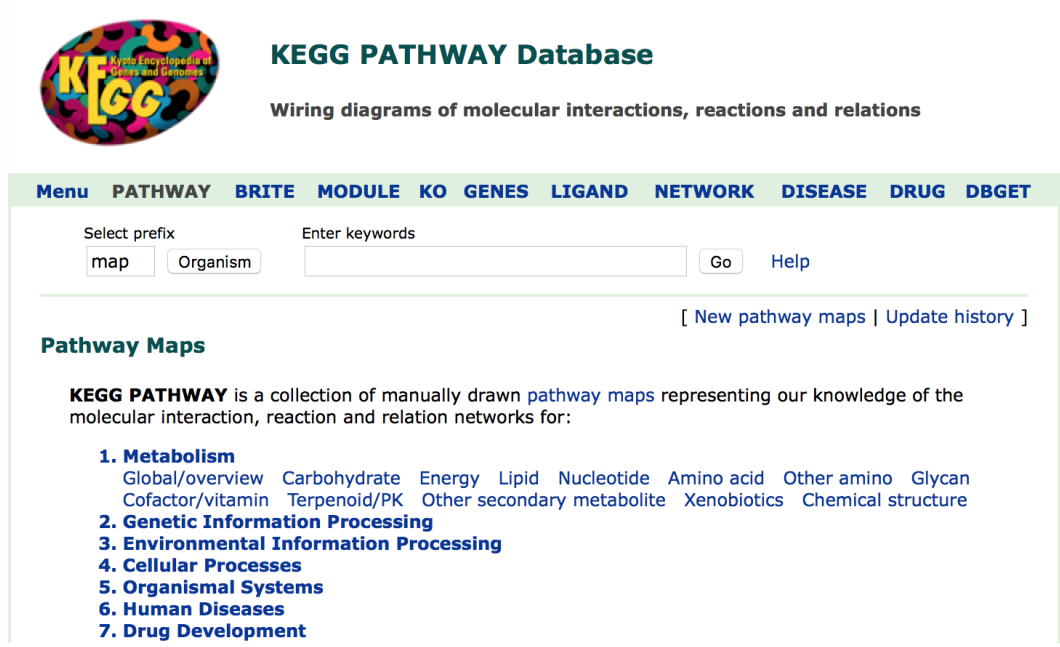

```
## Hierarchical clustering on edgeR & limma-voom overlapped significant genes
> UP=function(df){
+ rownames(df)[df$change=='UP']
+ }
> DOWN=function(df){
+ rownames(df)[df$change=='DOWN']
+ }
> up=intersect(UP(edgeR_DEG),UP(limma_voom_DEG))
> down=intersect(DOWN(edgeR_DEG),DOWN(limma_voom_DEG))
> mat_total=a[c(up,down),]
> n4=t(scale(t(mat_total)))
> n4[n4>1]=1
> n4[n4<-1]=-1
> ac=data.frame(group=group_list)
> rownames(ac)=colnames(mat_total)
> ht_combine<-pheatmap(n4,show_rownames = F,show_colnames = F,
+ cluster_rows = F,cluster_cols = T,annotation_col = ac, color = color)
> print(ht_combine)
> dev.off()
```



4. GO, KEGG pathway enrichment analysis

KEGG: https://www.kegg.jp/kegg/pathway.html

KEGG is a database resource for understanding high-level functions and utilities of the biological system, such as the cell, the organism and the ecosystem, from molecular-level information, especially large-scale molecular datasets generated by genome sequencing and other high-throughput experimental technologies.

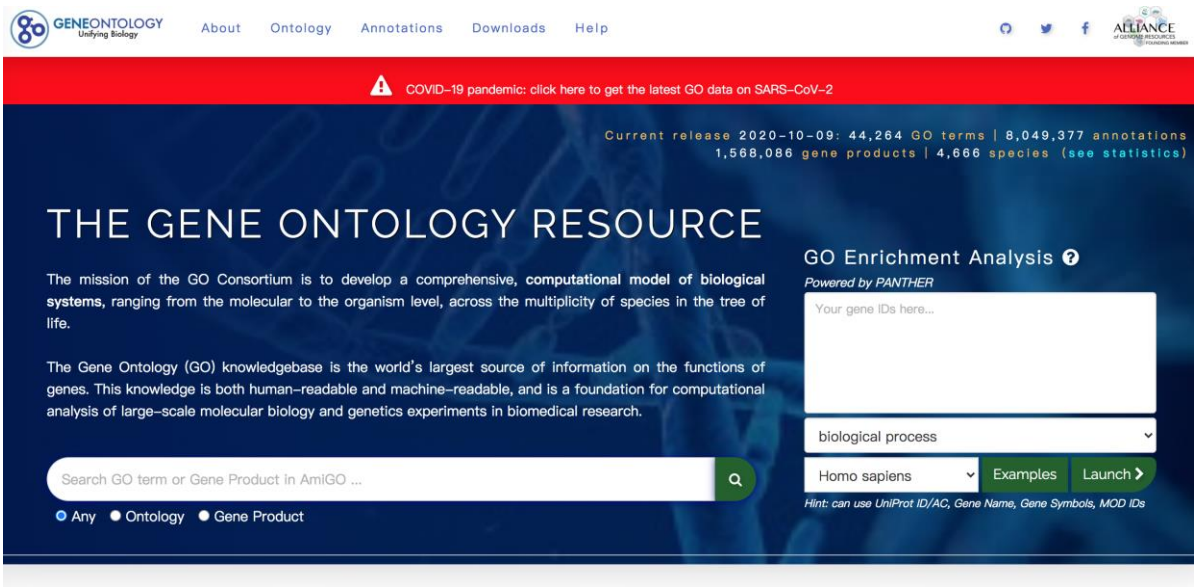


GO: http://geneontology.org/

The **Gene Ontology** (**GO**) is a major bioinformatics initiative to unify the representation of gene and gene product attributes across all species. More specifically, the project aims to: 1) maintain and develop its controlled vocabulary of gene and gene product attributes; 2) annotate genes and gene products, and assimilate and disseminate annotation data; and 3) provide tools for easy access to all aspects of the data provided by the project, and to enable functional interpretation of experimental data using the GO, for example via enrichment analysis.

"Ontologies" consist of representations of things that are detectable or directly observable, and the relationships between those things. There is no universal standard terminology in biology and related domains, and term usages may be specific to a species, research area or even a particular research group. This makes communication and sharing of data more difficult. The Gene Ontology project provides an ontology of defined terms representing gene product properties. The ontology covers three domains:

- **cellular component**, the parts of a cell or its extracellular environment;
- **molecular function**, the elemental activities of a gene product at the molecular level, such as binding or catalysis;
- **biological process**, operations or sets of molecular events with a defined beginning and end, pertinent to the functioning of integrated living units: cells, tissues, organs, and organisms.



Enrichment

Enrichment module gives you the answer of which pathways, diseases, and GO terms is statistically significant associated with the genes/proteins you just input.

Gene List Enrichment

It accepts same input formats as Annotation module, and the results of Annotation module as input is also allowed. It is based on the first generation gene set enrichment method, a gene-level statistic called Overrepresentation Analysis(ORA), a simple and frequently used test based on the hypergeometric distribution. Many tools have applied this methods, such as DAVID. However, we support other distributions like binominal test, chi-square test, frequency list and 3 FDR correction methods, like Benjamini and Hochberg (1995), Benjamini and Yekutieli (2001), and QVALUE.

WebGestalt (www.webgestalt.org)

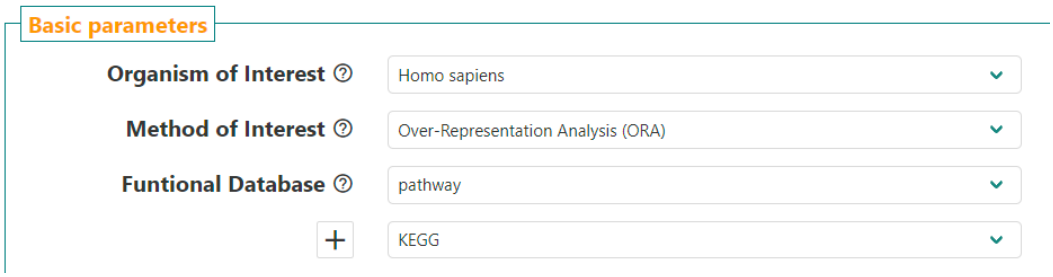
Introduction:

WebGestalt (WEB-based Gene SeT AnaLysis Toolkit) is a functional enrichment analysis web tool, which has on average 26,000 unique users from 144 countries and territories per year according to Google Analytics. The WebGestalt 2005, WebGestalt 2013 and WebGestalt 2017 papers have been cited in more than 2,500 scientific papers according to Google Scholar.

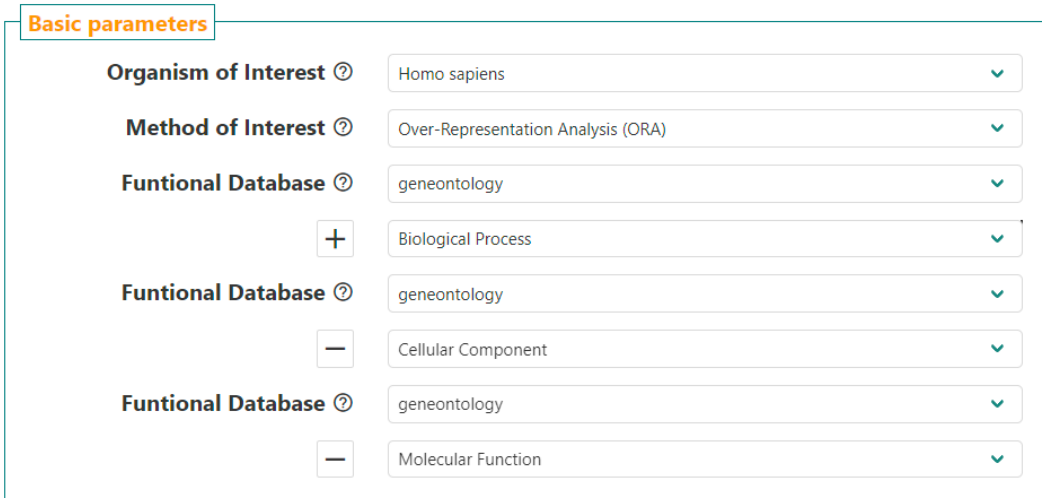
Step 1: Basic parameters setting

1. **Organism of Interest**: 'Homo Sapiens'
2. **Method of Interest**: 'ORA'
3. **Functional Database**: (KEGG pathway enrichment or GO enrichment)

   KEGG: 'pathway' ~ 'KEGG'

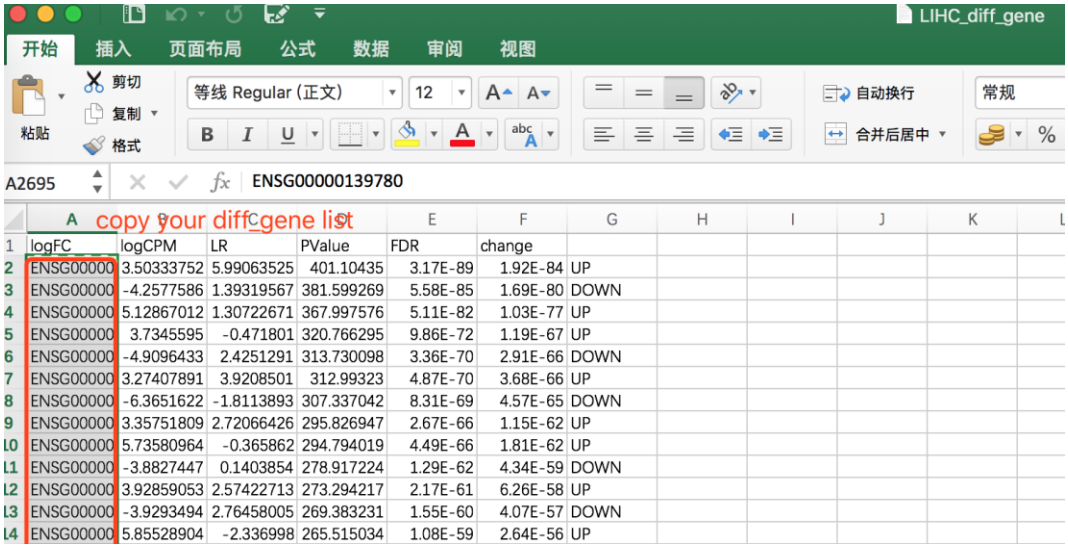


GO: ‘geneontology’ ~ ‘Biological Process’/’Cellular Component’/’Molecular Function’

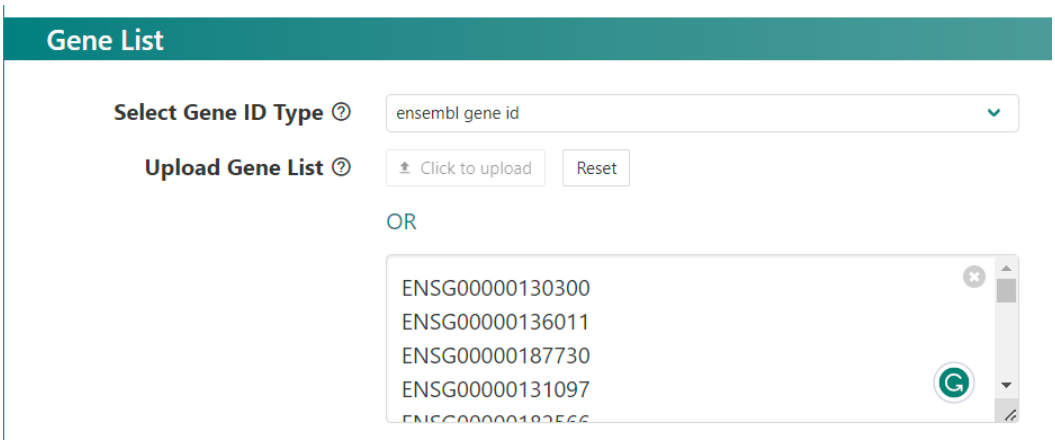


Step 2: Input your gene list and select reference gene list

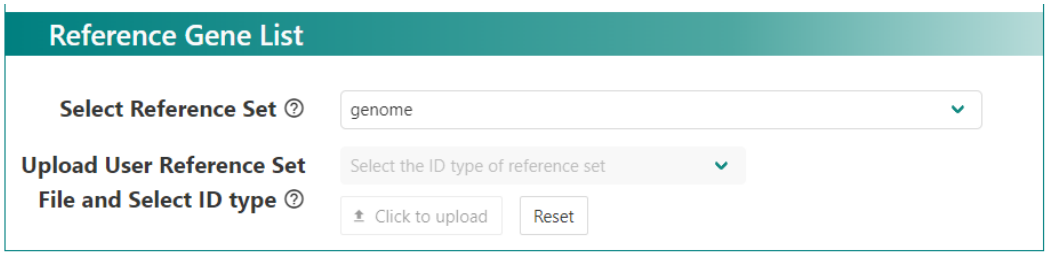
1. **Select Gene ID Type**: ‘ensemble gene id’
2. **Copy and paste your DEG gene list**



	logFC	logCPM	LR	PValue	FDR	change
2	ENSG00000	3.50333752	5.99063525	401.10435	3.17E-89	1.92E-84	UP
3	ENSG00000	-4.2577586	1.39319567	381.599269	5.58E-85	1.69E-80	DOWN
4	ENSG00000	5.12867012	1.30722671	367.997576	5.11E-82	1.03E-77	UP
5	ENSG00000	3.7345595	-0.471801	320.766295	9.86E-72	1.19E-67	UP
6	ENSG00000	-4.9096433	2.4251291	313.730098	3.36E-70	2.91E-66	DOWN
7	ENSG00000	3.27407891	3.9208501	312.99323	4.87E-70	3.68E-66	UP
8	ENSG00000	-6.3651622	-1.8113893	307.337042	8.31E-69	4.57E-65	DOWN
9	ENSG00000	3.35751809	2.72066426	295.826947	2.67E-66	1.15E-62	UP
10	ENSG00000	5.73580964	-0.365862	294.794019	4.49E-66	1.81E-62	UP
11	ENSG00000	-3.8827447	0.1403854	278.917224	1.29E-62	4.34E-59	DOWN
12	ENSG00000	3.92859053	2.57422713	273.294217	2.17E-61	6.26E-58	UP
13	ENSG00000	-3.9293494	2.76458005	269.383231	1.55E-60	4.07E-57	DOWN
14	ENSG00000	5.85528904	-2.336998	265.515034	1.08E-59	2.64E-56	UP

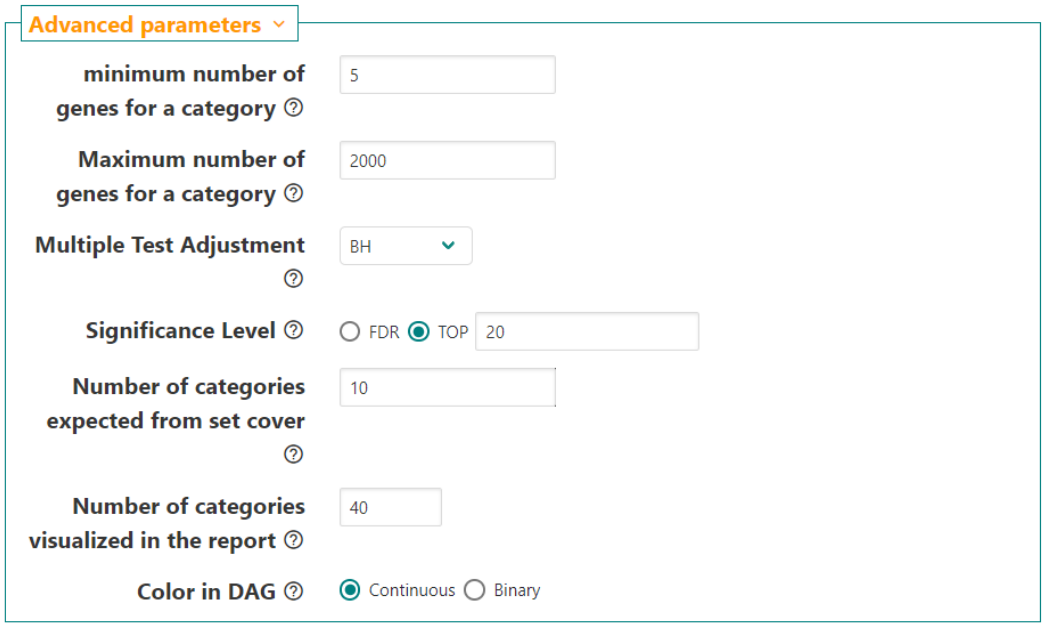


3. Input the background (reference set) for comparing with the input as Step 1. Background should be all the genes in your experiment, while gene list you have input is a set of genes you are interested in. If you want to use your own background, you should upload background genes first. Otherwise, you can **Select Reference Set** from database 'genome'.

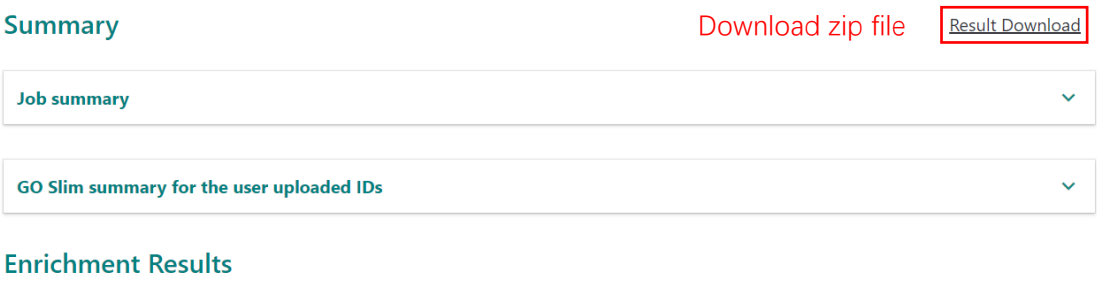


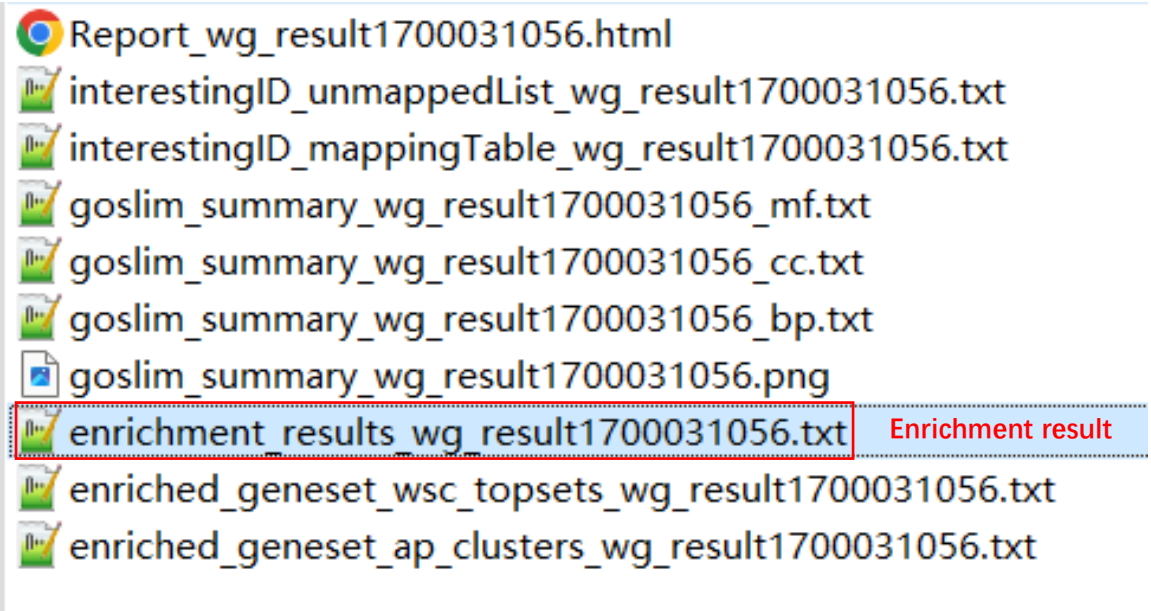
Step 3: Advanced parameters setting

1. **Significance Level:** 'Top 20' (you can try other thresholds)



Enrichment results output

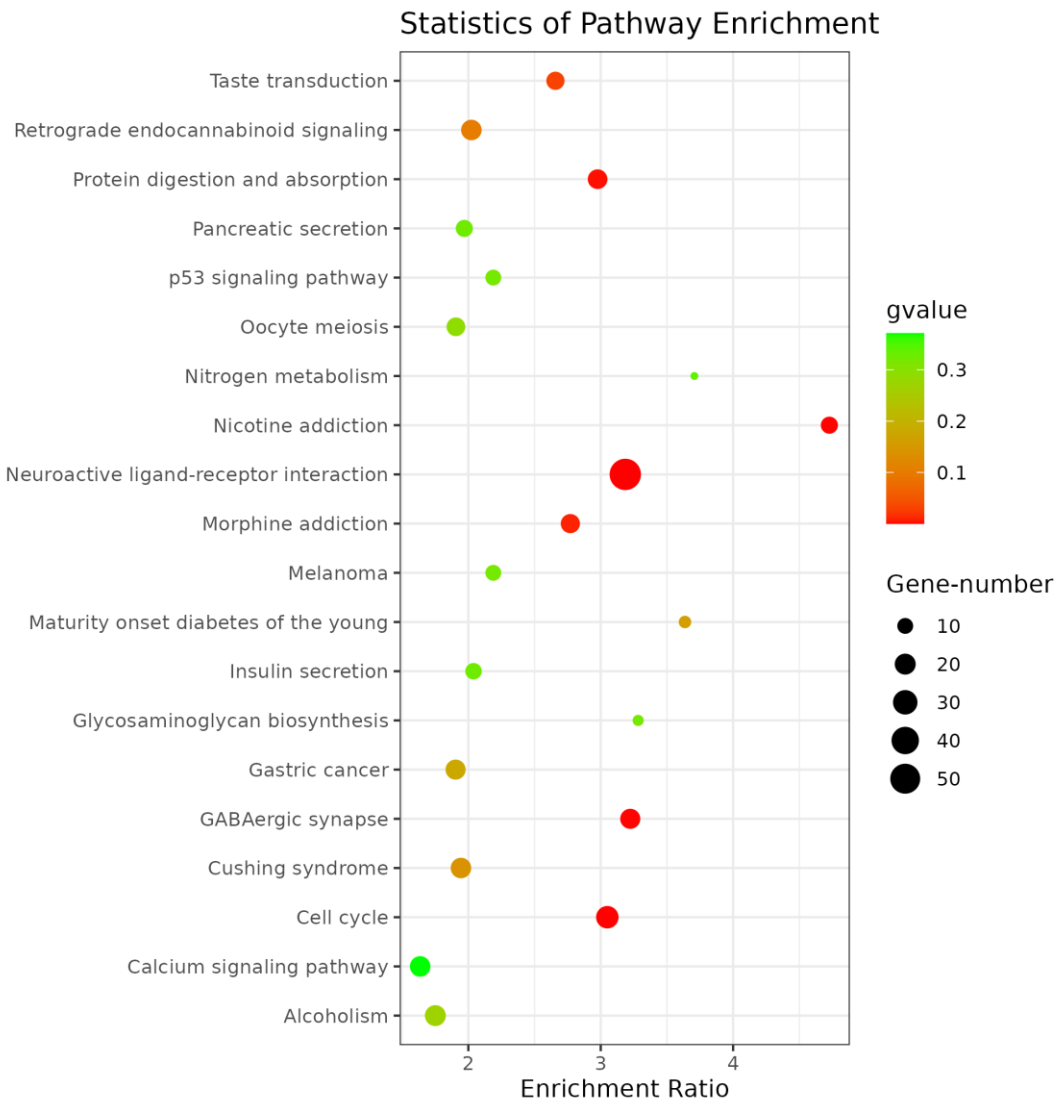



Report_wg_result1700031056.html
interestingID_unmappedList_wg_result1700031056.txt
interestingID_mappingTable_wg_result1700031056.txt
goslim_summary_wg_result1700031056_mf.txt
goslim_summary_wg_result1700031056_cc.txt
goslim_summary_wg_result1700031056_bp.txt
goslim_summary_wg_result1700031056.png
enrichment_results_wg_result1700031056.txt
Enrichment result
enriched_geneset_wsc_topsets_wg_result1700031056.txt
enriched_geneset_ap_clusters_wg_result1700031056.txt

Enrichment results visualization

[1] KEGG pathway enrichment results

```
# Read enrichment data
> pathway_enrich<-read.table("enrichment_results_wg_kegg.txt",header=T,sep="\t",stringsAsFactors=F)

# Draw a scatter plot
> library(ggplot2)
> colnames(pathway_enrich)
> p<-ggplot(pathway_enrich,aes(x=enrichmentRatio,y=description))
> p+geom_point()

# Map the size of the point to overlap and map the color to FDR
> p+geom_point(aes(size=overlap,color=FDR))+
    scale_color_gradient(low="red",high="green")+
    labs(title="Statistics of Pathway Enrichment",x="Enrichment Ratio",y="",color="gvalue",size="Gene-number")+
    theme_bw()
> ggsave('kegg.png',width=7,height=7)
> dev.off()
```



[2] GO enrichment results

```
# Read enrichment data
> go_enrich<-read.table("enrichment_results_wg_go.txt",header=T,sep="\t",stringsAsFactors=F)

# Draw a scatter plot
> library(ggplot2)
> colnames(go_enrich)
> p<-ggplot(go_enrich,aes(x=enrichmentRatio,y=description))
> p+geom_point()

# Map the size of the point to overlap and map the color to FDR
> p+geom_point(aes(size=overlap,color=FDR))+
    scale_color_gradient(low="red",high="green")+
    labs(title="Statistics of GO Enrichment",x="Enrichment Ratio",y="",color="gvalue",size="Gene-number")+
    theme_bw()
> ggsave('kegg.png',width=9,height=7)
> dev.off()
```