Dysregulated processes in NAFLD (BB103X)

Hong Yang

10/02/2022

2.1 Load necessary packages

- Rstudio
- load packages (need each time you run your code when you come back from)

```
library(DESeq2)
library(pheatmap)
library(tidyverse)
library(xlsx)
library(readxl)
library(gplots)
library(ggbiplot)
library(piano)
library(venn)
library(clusterProfiler)
library(GEOquery)
library(openxlsx)
library(GEOquery)
library(ggrepel)
```

2.2 Load data you've prepared

-metadata: characteristics of samples

-count: count values of genes fro each sample

-Those 2 data tables need to be merged at end of the pro-processing step

```
metadata = read.xlsx('./data/DGSE135251_metadata.xlsx')
sample2count = read.xlsx('./data/DGSE135251_sample2count_str.xlsx')
```

2.3 What the data looks like? (1)

```
## Metadata
metadata[1:6,1:4]
```

```
##                title geo_accession                status submission_date
## 1  Liver patient 97    GSM3998167 Public on Dec 03 2020     Aug 01 2019
## 2  Liver patient 98    GSM3998168 Public on Dec 03 2020     Aug 01 2019
## 3 Liver patient 101    GSM3998169 Public on Dec 03 2020     Aug 01 2019
## 4 Liver patient 102    GSM3998170 Public on Dec 03 2020     Aug 01 2019
## 5 Liver patient 105    GSM3998171 Public on Dec 03 2020     Aug 01 2019
## 6 Liver patient 106    GSM3998172 Public on Dec 03 2020     Aug 01 2019
```

- This table contains all the clinical information of samples in this study

```
sample2count[1:8,1:6]
```

```
##                        ID GSM3998167 GSM3998168 GSM3998169 GSM3998170 GSM3998171
## 1 __alignment_not_unique    3064384    2537331    3007853    3645473    3832511
## 2            __ambiguous     992499     855096    1062611    1225874    1187577
## 3           __no_feature    1544139    1238869    1050137    1345656    1491388
## 4          __not_aligned          0          0          0          0          0
## 5        __too_low_aQual          0          0          0          0          0
## 6        ENSG00000000005          0         14          0          0          0
## 7        ENSG00000000419        605        525        709        671        869
## 8        ENSG00000000457        315        330        329        418        348
```

- This table contains the count of each gene in each sample.

2.4 What the data looks like? (2)

- Check all the columns that metadata table contains

```
colnames(metadata)
```

```
##  [1] "title"                   "geo_accession"
##  [3] "status"                  "submission_date"
##  [5] "last_update_date"        "type"
##  [7] "channel_count"           "source_name_ch1"
##  [9] "organism_ch1"            "characteristics_ch1"
## [11] "characteristics_ch1.1"   "characteristics_ch1.2"
## [13] "characteristics_ch1.3"   "characteristics_ch1.4"
## [15] "molecule_ch1"            "extract_protocol_ch1"
## [17] "extract_protocol_ch1.1"  "taxid_ch1"
## [19] "description"             "data_processing"
## [21] "data_processing.1"       "data_processing.2"
## [23] "data_processing.3"       "data_processing.4"
## [25] "platform_id"             "contact_name"
## [27] "contact_email"           "contact_institute"
## [29] "contact_address"         "contact_city"
## [31] "contact_zip/postal_code" "contact_country"
## [33] "data_row_count"          "instrument_model"
## [35] "library_selection"       "library_source"
## [37] "library_strategy"        "relation"
## [39] "relation.1"              "supplementary_file_1"
## [41] "disease:ch1"             "fibrosis.stage:ch1"
## [43] "group.in.paper:ch1"      "nas.score:ch1"
## [45] "Stage:ch1"
```

- Summary of samples in each group

```
table(metadata$`group.in.paper:ch1`)
```

```
##
##    control       NAFL NASH_F0-F1    NASH_F2    NASH_F3    NASH_F4
##         10         51         34         53         54         14
```

- Summary of samples in each fibrosis stage

```
table(metadata$`fibrosis.stage:ch1`)
```

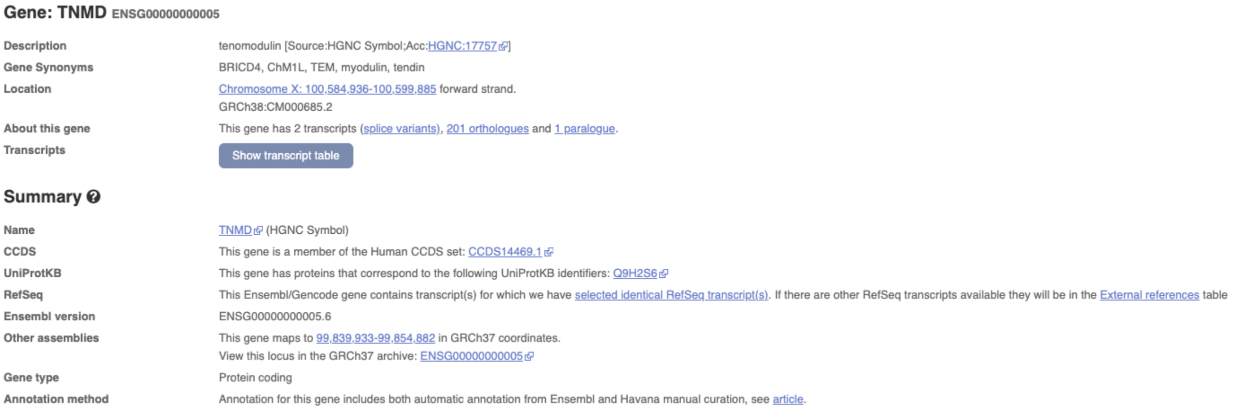
```
##
```

```
##  0  1  2  3  4
## 46 48 54 54 14
```

- Check other columns using the above code (just change the column names behind '$')

2.5 Annotate count table by ID conversion (*)

- It is not obvious when we look at the ID of gene, for example, ENSG00000000005. So we need to convert this ID to gene name.
- ENSG00000000005 is identifier of gene TNMN in database Ensembl
-



Gene: TNMD ENSG00000000005

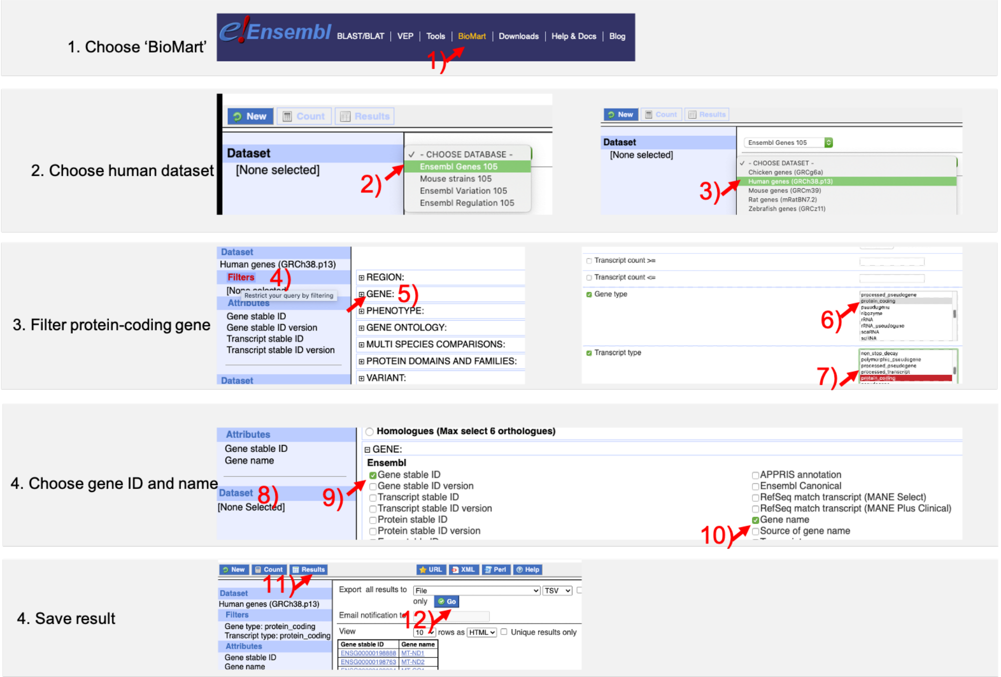
Description	tenomodulin [Source:HGNC Symbol;Acc:HGNC:17757]
Gene Synonyms	BRICD4, ChM1L, TEM, myodulin, tendin
Location	Chromosome X: 100,584,936-100,599,885 forward strand. GRCh38:CM000685.2
About this gene	This gene has 2 transcripts (splice variants), 201 orthologues and 1 paralogue.
Transcripts	Show transcript table

Summary

Name	TNMD (HGNC Symbol)
CCDS	This gene is a member of the Human CCDS set: CCDS14469.1
UniProtKB	This gene has proteins that correspond to the following UniProtKB identifiers: Q9H2S6
RefSeq	This Ensembl/Gencode gene contains transcript(s) for which we have selected identical RefSeq transcript(s). If there are other RefSeq transcripts available they will be in the External references table
Ensembl version	ENSG00000000005.6
Other assemblies	This gene maps to 99,839,933-99,854,882 in GRCh37 coordinates. View this locus in the GRCh37 archive: ENSG00000000005
Gene type	Protein coding
Annotation method	Annotation for this gene includes both automatic annotation from Ensembl and Havana manual curation, see article.

2.6 (*) Download a table for annotation

- This table would contain 2 columns, 1st column is ID, 2nd column is gene name
- Using BioMart in Ensembl
- https://www.ensembl.org/index.html



1. Choose 'BioMart'

2. Choose human dataset

3. Filter protein-coding gene

4. Choose gene ID and name

4. Save result

- save the result under 'data' folder and name as 'ID2genename.database.txt'

2.7 Load annotation table into R

```
ID2gene = read.table('./data/ID2genename.database.txt', sep = '\t', header = TRUE) %>% distinct()
head(ID2gene)
```

```
##    Gene.stable.ID Gene.name
## 1 ENSG00000198888    MT-ND1
## 2 ENSG00000198763    MT-ND2
## 3 ENSG00000198804    MT-CO1
## 4 ENSG00000198712    MT-CO2
## 5 ENSG00000228253   MT-ATP8
## 6 ENSG00000198899   MT-ATP6
```

2.8 Merge annotation table into count table

```
count_reindex = merge(sample2count, ID2gene, by.x = 'ID', by.y = 'Gene.stable.ID') %>%
  select(Gene.name, everything())

### Check if we mapped data correctly
(count_reindex %>% filter(ID == 'ENSG00000198888'))[,1:6]
```

```
##   Gene.name              ID GSM3998167 GSM3998168 GSM3998169 GSM3998170
## 1    MT-ND1 ENSG00000198888     143159     111171     164564     159035
```

2.9 Processing the case with one gene related to multiple ID

- for example: gene 'ABHD16A'

```
(count_reindex %>% filter(Gene.name == 'ABHD16A'))[,1:6]
```

```
##   Gene.name              ID GSM3998167 GSM3998168 GSM3998169 GSM3998170
## 1   ABHD16A ENSG00000204427         12         12         10          7
## 2   ABHD16A ENSG00000206403          0          0          0          0
## 3   ABHD16A ENSG00000224552          0          0          0          0
## 4   ABHD16A ENSG00000230475          0          0          0          0
## 5   ABHD16A ENSG00000231488          0          0          0          0
## 6   ABHD16A ENSG00000235676          0          0          0          0
## 7   ABHD16A ENSG00000236063          0          0          0          0
```

Solution: take one row that have high expression value across samples

First of all, find all the rows need to be processed.

```
count_multipleIDs = count_reindex %>% select(Gene.name) %>% group_by(Gene.name) %>% tally() %>% filter(
head(count_multipleIDs)
```

```
## # A tibble: 6 x 2
##   Gene.name     n
##   <chr>     <int>
## 1 ""          267
## 2 "AADACL2"     2
## 3 "AATF"        2
## 4 "ABCB11"      2
## 5 "ABCC1"       2
## 6 "ABCC6"       2
```

n: represents the number of rows (IDs) for one specific gene.

The total number of genes with multiple IDs

```
dim(count_multipleIDs)
```

```
## [1] 692   2
```

- The output indicates there are 692 genes with multiple IDs or null in ‘Gene.name’ columns.

Second, calculate the mean value of each ID and decide which row will take for downstream analysis based on above statistics

Multiple steps for this process

a. filter rows that have the gene names with multiple IDs

b. filter out rows that contain nothing in column ‘Gene.name’

c. convert count value to numeric since it is character now, which cann’t be used for calculation

d. calculate the mean count value across samples and add a new columns named as ‘mean_row’

e. put the column ‘mean_row’ at the begining of this subset table

```
count_multipleIDs_subset = count_reindex %>%
  filter(Gene.name %in% count_multipleIDs$Gene.name) %>%
  filter(Gene.name != '') %>% mutate(across(starts_with('GSM'), ~as.numeric(.))) %>%
  mutate(mean_row = rowMeans(across(where(is.numeric)))) %>% select(mean_row, everything()) %>%
  arrange(., desc(Gene.name))

###
count_multipleIDs_subset[1:6,1:6]
```

```
##       mean_row Gene.name              ID GSM3998167 GSM3998168 GSM3998169
## 1 376.45833333    ZNHIT3 ENSG00000273611        416        358        506
## 2   0.00000000    ZNHIT3 ENSG00000278574          0          0          0
## 3  11.85185185     ZNF85 ENSG00000105750          4          9         23
## 4   0.00000000     ZNF85 ENSG00000278091          0          0          0
## 5   0.06018519    ZNF729 ENSG00000196350          0          0          0
## 6   0.00000000    ZNF729 ENSG00000279552          0          0          0
```

Third, subset the dataset based on above calculation

```
count_multipleIDs_subset_selected = count_multipleIDs_subset %>%
  group_by(Gene.name) %>% slice_max(mean_row) %>%
  filter(!(mean_row == 0)) %>% arrange(., desc(Gene.name))

count_multipleIDs_subset_selected[1:6,1:6]
```

```
## # A tibble: 6 x 6
## # Groups:   Gene.name [6]
##   mean_row Gene.name ID              GSM3998167 GSM3998168 GSM3998169
##      <dbl> <chr>     <chr>                <dbl>      <dbl>      <dbl>
## 1 376.     ZNHIT3    ENSG00000273611        416        358        506
## 2  11.9    ZNF85     ENSG00000105750          4          9         23
## 3   0.0602 ZNF729    ENSG00000196350          0          0          0
## 4  54.9    ZNF707    ENSG00000181135         62          52          47
## 5   4.63   ZNF676    ENSG00000196109          2           2           1
## 6   2.41   ZNF66     ENSG00000160229          1           2           3
```

Finally, combine rows with multiple ID after filtering and rows with unique IDs

```
count_uniqueIDs = count_reindex %>% filter(!(Gene.name %in% count_multipleIDs$Gene.name))
count_multipleIDs_subset_selected = count_multipleIDs_subset_selected %>% select(-mean_row)

sample2count_processed = rbind(count_uniqueIDs, count_multipleIDs_subset_selected)
dim(sample2count_processed)
```

```
## [1] 19081   218
```

In summary, the proceed table contains the count data of 19081 protein-coding gene.

```
sample2count_processed[1:6, 1:6]
```

```
##   Gene.name              ID GSM3998167 GSM3998168 GSM3998169 GSM3998170
## 1      TNMD ENSG00000000005          0         14          0          0
## 2      DPM1 ENSG00000000419        605        525        709        671
## 3     SCYL3 ENSG00000000457        315        330        329        418
## 4  C1orf112 ENSG00000000460         92         89        115         97
## 5       FGR ENSG00000000938         96         91        292        115
## 6       CFH ENSG00000000971      41898      38693      37744      42796
```

2.10 The results would be expected to be count matrix

- Gene name would be setted to be row names

```
sample2count_processed = as.matrix(sample2count_processed %>%
  column_to_rownames(., var = 'Gene.name') %>% select(-ID))

mode(sample2count_processed) = 'numeric'
sample2count_processed[1:6,1:6]
```

```
##          GSM3998167 GSM3998168 GSM3998169 GSM3998170 GSM3998171 GSM3998172
## TNMD              0         14          0          0          0          7
## DPM1            605        525        709        671        869        500
## SCYL3           315        330        329        418        348        347
## C1orf112         92         89        115         97        108         65
## FGR              96         91        292        115        144        154
## CFH           41898      38693      37744      42796      44933      62515
```

2.11 Differential analysis

- DeSeq package will be used in this step
- Read the website carefully to study the methods implemented in this package
- https://bioconductor.org/packages/release/bioc/vignettes/DESeq2/inst/doc/DESeq2.html

```
mkdir Figures
```

2.12 Generate DESeqDataSet (called dds)

- read document to study DESeqDataSet

```
rownames(metadata) = metadata$geo_accession
sample2count_processed = sample2count_processed[,rownames(metadata)]

####Perform the Differential Expression Analysis
conds=as.factor(metadata$`group.in.paper:ch1`)
coldata = data.frame(row.names=rownames(metadata),conds)
dds = DESeqDataSetFromMatrix(countData=round(as.matrix(sample2count_processed)),
                             colData=coldata,
                             design=~conds)
```

```
## converting counts to integer mode
```

```
##   Note: levels of factors in the design contain characters other than
##   letters, numbers, '_' and '.'. It is recommended (but not required) to use
##   only letters, numbers, and delimiters '_' or '.', as these are safe characters
##   for column names in R. [This is a message, not a warning or an error]
```

2.13 Gene-level exploratory analysis

- PCA: principle component analyis, read here
- Here is an example for PCA analysis based on vst transformation method,

```
# PCA analysis
vsd = vst(dds, blind = FALSE)
```

```
##   Note: levels of factors in the design contain characters other than
##   letters, numbers, '_' and '.'. It is recommended (but not required) to use
##   only letters, numbers, and delimiters '_' or '.', as these are safe characters
##   for column names in R. [This is a message, not a warning or an error]
```

```
pcadata = plotPCA(vsd,intgroup=c("conds"), returnData = TRUE)
percentVar = round(100 * attr(pcadata, "percentVar"))

pcadata_p = ggplot(pcadata, aes(x = PC1, y = PC2, color = factor(conds))) +
  geom_point(size =3, aes(fill=factor(conds),shape=factor(conds))) +
  scale_shape_manual(values=c(16,18,17,15,14,13)) +
  geom_label_repel(aes(label = name),box.padding = 0.35, point.padding = 0.5, segment.color = 'grey50')
  xlab(paste0("PC1: ", percentVar[1], "% variance")) + ylab(paste0("PC2: ", percentVar[2], "% variance"
  theme(axis.text.x = element_blank(), axis.title = element_text(size = 16), legend.text = element_text

ggsave(pcadata_p, filename = "./Figures/00-pca_group.pdf", height = 3.5, width = 5)
```

```
## Warning: ggrepel: 213 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```

```
pcadata_p_withname = ggplot(pcadata, aes(x = PC1, y = PC2, color = factor(conds))) +
  geom_point(size =3, aes(fill=factor(conds),shape=factor(conds))) +
  scale_shape_manual(values=c(16,18,17,15,14,13)) +
  geom_label_repel(aes(label = name),box.padding = 0.35, point.padding = 0.5, segment.color = 'grey50')
  xlab(paste0("PC1: ", percentVar[1], "% variance")) + ylab(paste0("PC2: ", percentVar[2], "% variance"
  theme(axis.text.x = element_blank(), axis.title = element_text(size = 16), legend.text = element_text

ggsave(pcadata_p, filename = "./Figures/00-pca_group_withname.pdf", height = 6, width = 8)
```

```
## Warning: ggrepel: 190 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```

please try other method (such as log2, rlog) and generate the graphs for the analyses.

2.14 Perform differential analysis

- First of all, define the groups that you want to compare

```
table(metadata$`group.in.paper:ch1`)
```

```
##
##    control       NAFL NASH_F0-F1    NASH_F2    NASH_F3    NASH_F4
##         10         51         34         53         54         14
```

- for example, difference between control group and NAFL group

```
dds_re = DESeq(dds)
```

```
## estimating size factors

##   Note: levels of factors in the design contain characters other than
##   letters, numbers, '_' and '.'. It is recommended (but not required) to use
##   only letters, numbers, and delimiters '_' or '.', as these are safe characters
##   for column names in R. [This is a message, not a warning or an error]

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

##   Note: levels of factors in the design contain characters other than
##   letters, numbers, '_' and '.'. It is recommended (but not required) to use
##   only letters, numbers, and delimiters '_' or '.', as these are safe characters
##   for column names in R. [This is a message, not a warning or an error]

## final dispersion estimates

##   Note: levels of factors in the design contain characters other than
##   letters, numbers, '_' and '.'. It is recommended (but not required) to use
##   only letters, numbers, and delimiters '_' or '.', as these are safe characters
##   for column names in R. [This is a message, not a warning or an error]

## fitting model and testing

##   Note: levels of factors in the design contain characters other than
##   letters, numbers, '_' and '.'. It is recommended (but not required) to use
##   only letters, numbers, and delimiters '_' or '.', as these are safe characters
##   for column names in R. [This is a message, not a warning or an error]

## -- replacing outliers and refitting for 131 genes
## -- DESeq argument 'minReplicatesForReplace' = 7
## -- original counts are preserved in counts(dds)

## estimating dispersions

##   Note: levels of factors in the design contain characters other than
##   letters, numbers, '_' and '.'. It is recommended (but not required) to use
##   only letters, numbers, and delimiters '_' or '.', as these are safe characters
##   for column names in R. [This is a message, not a warning or an error]

## fitting model and testing

##   Note: levels of factors in the design contain characters other than
##   letters, numbers, '_' and '.'. It is recommended (but not required) to use
##   only letters, numbers, and delimiters '_' or '.', as these are safe characters
##   for column names in R. [This is a message, not a warning or an error]
```

```
cond1 = 'NAFL' #First Condition
cond2 = 'control' #Reference Condition
res=results(dds_re,contrast=c('conds',cond1,cond2))
res$Gene.name = res@rownames
write.xlsx(res,file='./data/DEseq_results_NAFLvscontrol.xlsx')
```

please complete all the comparisons that you want to look at the difference.