Codes: A Journey to Dive into Biological Information

Budhaditya Basu

2023-02-26

Table of contents

Preface

For last four years I have been actively involved in all bioinformatics projects in our lab. In the beginning of my career I have never imagined myself to become a bioinformatician. I was and am a trained molecular biologist. Cell culture, cloning, animal handling and gene expression studies are my domain. When I joined my lab as a graduate student I saw that my seniors and PI were utterly frustrated with the bioinformatics service provided by the well known NGS analytics industries in India. Industries never paid attention to the biological questions being addressed by the project. There was a real need of a person who had the necessary computational expertise to handle the NGS data as well as understands the biological questions thoroughly. So I invested time during the lab hours to understand the biological questions and learned new molecular techniques. When I came to hostel and mostly during weekends I invested full time to learn about NGS analysis. Weekends have shaped me what I am today. Obviously that came at a cost of personal life. I owe a lot to my closer ones who always had a complaint that I never gave them my time.

Here, in this book I have documented my journey to stitch the code blocks and several resources which I think may help others who are on the same path. I believe that this text will enable the learners to transition from learning to do their own biological data analysis.

Introduction

Biological data analysis

This is a handbook of data analysis in the field of Biology. The main aim of this text is to introduce the concepts of the followings.

1. Introduction to unix
2. Introduction to R for Biologists
3. Data wrangling and data visualization
4. RNA-Seq analysis
5. Single-cell RNA-Seq analysis

By virtue of its requirement there is also a need to introduce

5. Managing Conda environment
6. Git and Github
7. Shiny web application
8. Containerized Bioinformatics

In due course, I will refer to the relevant resources for further reading. There are enough books which provide with detailed introduction and theory behind every topic. My purpose is not to repeat the same but to give you a full flavor of hands-on bioinformatics with lots of unix commands and R code blocks and resulting data visualization. I will try to make the codes as simple as possible and add meaning of the code block as and when required. Still I urge you that in any section or any code block that you cannot execute or becoming difficult to understand please contact me. I am always available to help.

Part I

RNA Sequencing Analysis

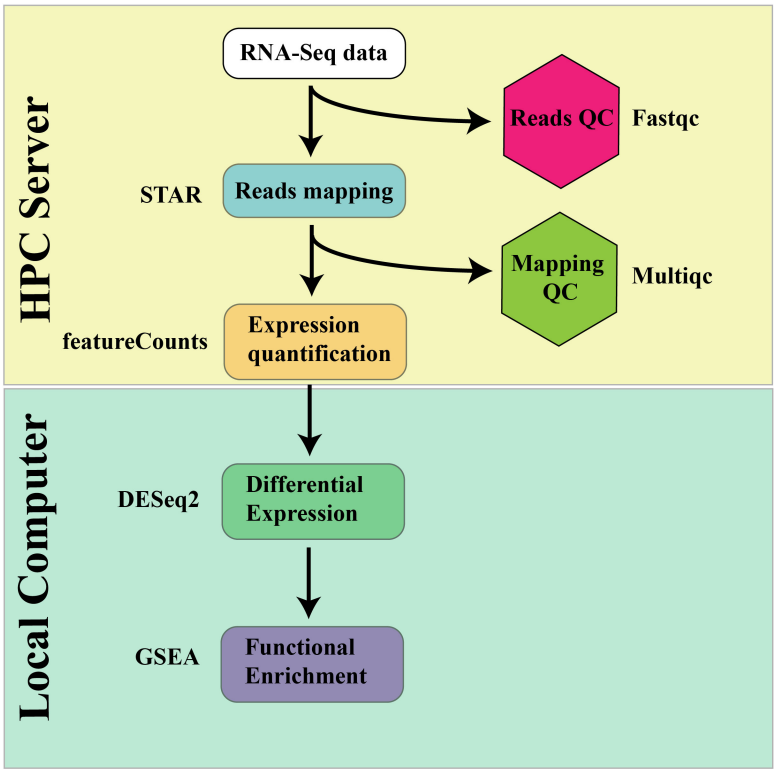


Figure 1: In this section of the book, you'll learn the entire workflow of RNA-Seq data analysis. 1. It starts with quality checking of the fastq files using fastqc. 2. The sequencing data is then aligned to the reference genome using STAR. The mapping QC is performed to check percentage of alignment to the reference genome. 3. Expression count is then calculated using featureCounts. 4. DESeq2 is used for calculating differntial gene expression (DGE) analysis. 5. Functional enrichment of the DGE is performed using GSEA.

1 Conda environment

1.1 Introduction

- conda is a package manager where users can easily find and install thousands of packages.
- It is also an environment manager which can create a separate environment within a few seconds which runs on different dependencies.
- It is compatible with multiple OS.
- Miniconda (minimal version of conda) can be downloaded as a script file (.sh) from the given link and installed using the following command.
- Download Miniconda (file size ~ 2.36 Mb)

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
```

- Install Miniconda

```
bash Miniconda3-latest-Linux-x86_64.sh
```

```
turiya@DESKTOP-L2O3RLM:~$ bash Miniconda3-latest-Linux-x86_64.sh

Welcome to Miniconda3 py310_23.1.0-1

In order to continue the installation process, please review the license
agreement.
Please, press ENTER to continue
>>>
```

- Press ENTER

```
Do you accept the license terms? [yes|no]
[no] >>>
Please answer 'yes' or 'no':'
>>>
```

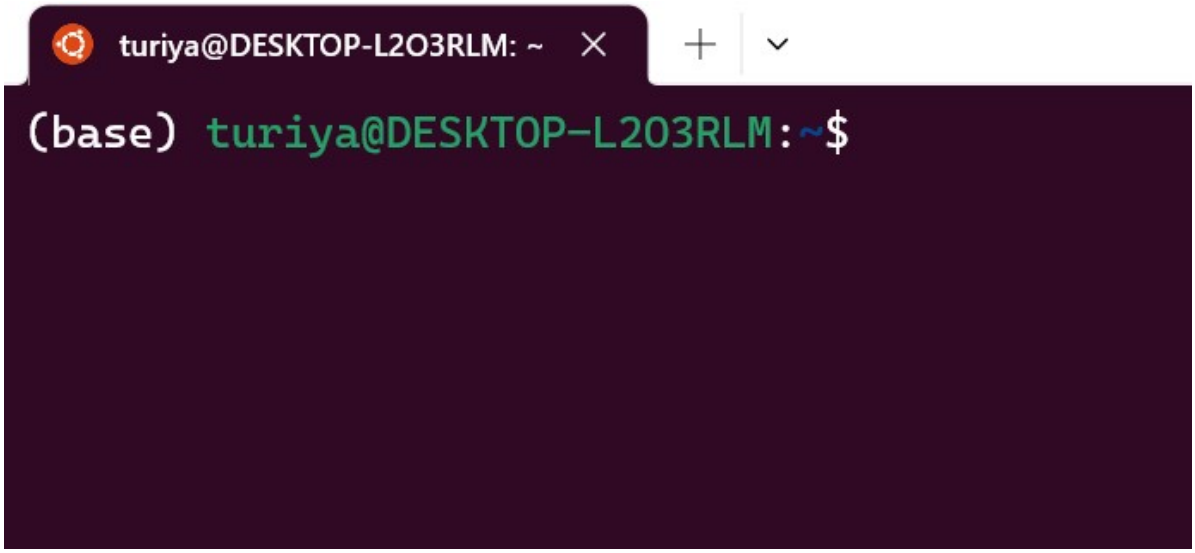
- Type yes

```
Miniconda3 will now be installed into this location:
/home/turiya/miniconda3

  - Press ENTER to confirm the location
  - Press CTRL-C to abort the installation
  - Or specify a different location below

[/home/turiya/miniconda3] >>>
```

- Press ENTER

```
Downloading and Extracting Packages

Preparing transaction: done
Executing transaction: done
installation finished.
Do you wish the installer to initialize Miniconda3
by running conda init? [yes|no]
[no] >>>
```

- This means every time you open your terminal, it will initialize conda by default.
- Type yes. If you wish to reverse it later, you can do it.
- The Miniconda installation process is complete. Close the terminal and open again.



- (base) at the command prompt indicate conda base environment is active.
- For basic conda commands, you can follow this link. Conda commands
- This installation process would create a directory called miniconda3 which would have all the installed packages as well as the environment dependencies.

> **Warning**
>
> Please do not delete 'miniconda3' directory. Deleting this directory would remove conda environment.

1.2 Create a conda environment with required packages

- Here I want to create a conda environment with all packages required for RNA-Seq analysis. The details of the packages will be discussed in the next section.
- ENV Name - rnaSeq
- Packages: fastqc, trim-galore, star, subread, multiqc
- Packages to be installed from bioconda channel.

```
conda create --name rnaSeq -c bioconda fastqc trim-galore star subread multiqc
```

- The process may take several minutes depending on the network speed.

```
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate rnaSeq
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) turiya@DESKTOP-L2O3RLM:~$
```

```
#List conda environments
conda env list
```

```
(base) turiya@DESKTOP-L2O3RLM:~$ conda env list
# conda environments:
#
base                  *  /home/turiya/miniconda3
rnaSeq                   /home/turiya/miniconda3/envs/rnaSeq

(base) turiya@DESKTOP-L2O3RLM:~$
```

- To activate our desired environment, run the following command.

```
conda activate rnaSeq
```

```
(base) turiya@DESKTOP-L2O3RLM:~$ conda activate rnaSeq
(rnaSeq) turiya@DESKTOP-L2O3RLM:~$
```

- Check whether our desired packages are installed or not. For example check trim-galore.

```
conda list | grep trim-galore
```

```
(rnaSeq) turiya@DESKTOP-L2O3RLM:~$ conda list | grep trim-galore
trim-galore               0.6.10                hdfd78af_0    bioconda
(rnaSeq) turiya@DESKTOP-L2O3RLM:~$
```

- In the next section, we shall discuss about using remote HPC server.

2 Connect to a remote unix computer

2.1 Getting Started

In this section we shall learn how to connect to a unix machine with `ssh` and run few basic commands. Later I shall introduce with the concept of bash scripting.

- Fire up your terminal and execute the following code.

```
ssh userid@ipaddress
```

- Here userid can be a short name which was used to create an account on the HPC server. Example: rohit
- IP address is a numerical identifier of a system. Example: 213.253.210.118
- Please ask your network administrator for these details before proceeding.
- Next it would ask for password

```
userid@ipaddress's password:
```

- After I entered my password it displayed like this.

```
(base) basu@darwin:~$
```

> Tip
>
> If you are new to unix computer, then you are recommended to be familiar with the basic commands of unix. Check this link for Basic unix commands If you are not sure what a command means/does? Type it here and get a very easy to understand explanation. explainshell.com

3 Package Installation

3.1 Install FastQC

- The first step of RNA_Seq data analysis is to check the quality of the sequencing data in .fastq/.fq file format.
- FastQC is the commonly used tool to perform quality control.

```
conda install -c bioconda fastqc
```

- To check whether installation is complete or not, just type in the terminal the following.

```
fastqc -h
```

3.2 Install Trim galore

- Trim Galore! is a wrapper script to automate quality and adapter trimming as well as quality control.
- If adapter content is found then adapter should be removed using Trim Galore.

```
conda install -c bioconda trim-galore
```

- To check the successful installation

```
trim_galore -help
```

3.3 Install STAR aligner

- The alignment process involves performing the read alignment using one of several splice-aware alignment tools such as STAR or HISAT2.
- The choice of aligner is often a personal preference and also dependent on the computational resources.

```
conda install -c bioconda star

STAR -h
```

3.4 Install subread

- Once alignment is complete, the next step would be to count of aligned reads to each gene.
- `featureCounts` is a part of the subread toolkit which is used for this purpose.

```
conda install -c bioconda subread

featureCounts -h
```

> **Note**
>
> Initial QC to be done using FastQC, followed by trimming with TrimGalore!. Reads will be aligned using STAR and overlaps to be counted with featureCounts.

3.5 Install MultiQC

- MultiQC searches a given directory for analysis logs and compiles a HTML report. It's a general use tool, perfect for summarising the output from numerous bioinformatics tools.

```
conda install -c bioconda multiqc

multiqc -h
```

4 Quality Control of The Sequencing Reads

4.1 Activate conda environment

```
conda activate rnaSeq
```

4.2 Run FastQC

- Raw sequencing data is stored in any one of the formats listed i.e., .fastq/.fastq.gz/.fq.gz
- Quality checking of the raw data is the first step to be performed before proceeding with further analysis.
- Take a look at the example of a good illumina data.
- Take a look at the example of a bad illumina data.
- Now, let's jump to analyse our own data.

```
#Make a directory for the fastq files

mkdir fastq/

#Move all the fastq files into that recently created directory

mv *.fastq.gz fastq/
```

- Create a directory where fastqc output will be stored

```
mkdir fastqc_out/
```

- RUN FastQC

```
fastqc -o fastqc_out/ -t 8 fastq/*.fastq.gz

# -t (--threads) specifies the number of threads for parallel computing
```

- Check the fastqc output .html files stored in fastqc_out directory. Transfer the .html files to local computer using either WinSCP or FileZilla.
- If adapter content is found or sequence quality is suboptimal then those sequence need to be trimmed using trim_galore.
- Create a directory where trimmed fastq files will be stored.

```
mkdir trim_galore/
```

- If you want to remove the adapter content, run Trim Galore.

4.3 Run Trim Galore!

- Before that we want to check our bash script. Execute the following command and check the output

```
for filename in fastq/*_R1.fastq.gz
do
base=$(basename $filename _R1.fastq.gz)
echo "Running alignment for ${base} using STAR"
done
```

- If everything looks okay then we have create a file, trim-galore.sh

```
vim trim-galore.sh
```

- Insert the following script by pressing 'I' in your keyboard (meaning insert). Paste the following script. Press Esc and then type :wq to save. Later make the file executable.

```
 chmod +x trim-galore.sh

 bash trim-galore.sh
```

```
#!/bin/bash/
for filename in fastq/*_R1.fastq.gz
do
base=$(basename $filename _R1.fastq.gz)

trim_galore --illumina --paired -j 8 \
--fastqc -o trim_galore/ fastq/${base}_R1.fastq.gz\
fastq/${base}_R2.fastq.gz
done
```

- –ilumina : Adapter sequence to be trimmed of illumina universal adapter. –paired: Trimming for paired end files. -j :Number of cores to be used for trimming. –fastqc : Once trimming is complete, run fastqc in default mode. -o :Output directory

> **Note**
>
> The output of the above trimming command will have varied read length. For some sequencing data analysis (i.e., CIRI pipeline) it is not advisable to have varied read length. For differential gene expression analysis it is not an issue.

- In any case, if you require to remove adapter and have uniform read length then we can use `--hardtrim5` option in trim_galore.

```
trim_galore -j 8 --hardtrim5 100 -o trim_galore/ fastq/*.fastq.gz

# --hardtrim5 <int> will retain <int> bases from the 5' end of the sequence.
```

- After running trim-galore, it would create .html files which contain QC data and the trimmed fastq files as .fq.gz
- If you want to clean and categorize your data then move the trimmed fastq files to a new directory.

> **Tip**
>
> ```
> cd trim_galore/
> mkdir fastq/
> mv *.fq.gz fastq/
> ```

- Did you notice that all trimmed fastq files have names with a string 'trimmed' or 'val_1'? Next, I want to remove the string 'trimmed' from all file names.
- Here is my solution. But before looking into this you can try your own ways.

> **Tip**
>
> ```
> cd fastq/
> # Now remove the string '_trimmed' from all file names
>
> for file in *;do mv "${file}" "${file/_trimmed/}";done
> ```

5 Alignment to The Reference Genome

5.1 Download reference genome and annotation file

- Human reference genome

```
mkdir hg38/

cd hg38/

wget http://hgdownload.soe.ucsc.edu/\
goldenPath/hg38/bigZips/hg38.fa.gz

wget https://hgdownload.soe.ucsc.edu/\
goldenPath/hg38/bigZips/genes/hg38.refGene.gtf.gz
```

- Mouse Reference Genome

```
mkdir mm39/

cd mm39/

wget https://hgdownload.soe.ucsc.edu/\
goldenPath/mm39/bigZips/mm39.fa.gz

wget https://hgdownload.soe.ucsc.edu/\
goldenPath/mm39/bigZips/genes/refGene.gtf.gz
```

5.2 Generating Indexes

- We must first generate an index of the genome we want to align to, so that there tools can efficently map over millions of sequences

```
cd hg38/

STAR --runMode genomeGenerate \
--genomeDir star_index \
--genomeFastaFiles hg38.fa \
--sjdbGTFfile hg38.refGene.gtf \
--runThreadN 40
```

5.3 Alignment using STAR

- Check the following trial script whether it is working or not by executing it on the terminal.
- Go back one directory upward of trim_galore.

```
for filename in trim_galore/fastq/*_R1_val_1.fq.gz
do
base=$(basename $filename _R1_val_1.fq.gz)
echo "Running alignment for ${base} with STAR"
done
```

```
 vim star.sh
```

- Insert the following script by pressing 'I' in your keyboard (meaning insert). Paste the following script. Press Esc and then type :wq to save. Later make the file executable.

```
 chmod +x star.sh
 bash star.sh
```

```
#!/bin/bash/
echo "Starting Alignment"
for filename in trim_galore/fastq/*_R1_val_1.fq.gz
do
base=$(basename $filename _R1_val_1.fq.gz)
echo "Running alignment for ${base} with STAR"
STAR --runMode alignReads --outSAMtype BAM SortedByCoordinate \
--readFilesCommand zcat\
--genomeDir ~/hg38/star_index \
--outFileNamePrefix output/${base} \
--quantMode GeneCounts \
--readFilesIn trim_galore/fastq/${base}_R1_val_1.fq.gz\
```

```
trim_galore/fastq/${base}_R2_val_2.fq.gz \
--runThreadN 20
done
```

- This process will generate a directory 'output'.
- 'output' folder will contain files with the naming pattern: "sortedByCoord.out.bam".
- These files would be used in the next step of featureCount.

5.4 Run featureCount

```
cd output/

mkdir aligned_bam/

mv *.bam aligned_bam/

mkdir featureCount/

featureCounts -p -T 4 -a ~/hg38/hg38.refGene.gtf \
-o featureCount/final_counts_all.txt -g 'gene_name'\
aligned_bam/*.out.bam
```

- -p This is only applicable for paired-end reads. -T specifies the number (n) of threads to be used. -a is the genome annotation file (example_genome_annotation.gtf). -o specifies the name of the output file, which includes the read counts (example_featureCounts_output.txt).

5.5 Run multiqc

```
multiqc output/ --outdir multiqc_all
```

- It creates a html file compiling all the alignment data.
- Transfer featurecount and multiqc files to the local computer for further analysis.
- In the next section we shall discuss about differential gene expression analysis in RStudio.

6 Differential gene expression analysis

6.1 Analysis using DESeq2

- Here, we are going to perform differential gene expression analysis using DESeq2.
- Before this exercise, you are recommended to have basic R programming knowledge and data visualization skill. For that you can refer to my workshop material.

```
#==============================================================
# Install packages
#==============================================================
# Install bioconductor packages.
bioconductor_packages <- c(
  'DESeq2', 'clusterProfiler',
  'biomaRt', 'org.Hs.eg.db',
  'org.Mm.eg.db', 'enrichplot'
)

if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
BiocManager::install(bioconductor_packages)

# Install CRAN packages.
cran_packages <- c(
  'tidyverse', 'pheatmap',
  'msigdbr', 'RColorBrewer',
  'ggrepel'
)
install.packages(cran_packages)
#==============================================================
# Load the packages
#==============================================================
library(DESeq2)
library(ggplot2)
library(pheatmap)
```

```
library(RColorBrewer)

#=====================================================================
# Import data
#=====================================================================
# Import gene counts table generated from featureCounts
# - skip first row (general command info)
# - make row names the gene identifiers
countdata <- read.table("final_counts_all.txt",
                        header = TRUE, skip = 1,
                        row.names = 1)
head(countdata)
# Remove .bam from column identifiers
colnames(countdata) <- gsub("Aligned.sortedByCoord.out.bam",
                            "",
                            colnames(countdata),
                            fixed = T)
ncol(countdata)
#Take only expression values
countdata <- countdata[ ,c(6:11)]
head(countdata)
#=====================================================================
# Convert to matrix
#=====================================================================
countdata <- as.matrix(countdata)
head(countdata)
# Assign condition (first three are control,
# second three contain the Knock-Out)
condition <- factor(c(rep("Control", 3),
                      rep("Knock-Out", 3)))
head(countdata)
#=====================================================================
# Prepare DESeqDataSet
#=====================================================================
# Create a coldata frame and instantiate the DESeqDataSet.
#See ?DESeqDataSetFromMatrix
(coldata <- data.frame(row.names=colnames(countdata),
                       condition))
head(coldata)
dds <- DESeqDataSetFromMatrix(countData=countdata,
                              colData=coldata,
                              design=~condition)
```

```
dds$condition
# Determining the directionality of comparison.
dds$condition <- relevel(dds$condition, ref = "Control")
#==================================================================
# Run the DESeq2 pipeline
#==================================================================
dds <- DESeq(dds)
dds
# Sample level QC by PCA and hierarchical clustering methods
# Transform normalized counts using the rlog transformation
# Transform counts for data visualization
rld <- rlog(dds, blind=TRUE)
#==================================================================
# Principal components analysis (PCA)
#==================================================================
# Plot PCA
# Save the correlation plot
jpeg(filename = "PCA_plot.jpg",
     height = 4,width = 6,units = "in",res = 600)
plotPCA(rld, intgroup="condition")+ theme_light()
dev.off()
#==================================================================
#Hierarchical Clustering
#==================================================================
rld_mat <- assay(rld)
# Extract the rlog matrix from the object
rld_mat <- assay(rld)
# Compute pairwise correlation values
rld_cor <- cor(rld_mat)
# Plot heatmap
pheatmap(rld_cor)
heat.colors <- brewer.pal(6, "Greens")
# Save the correlation plot
jpeg(filename = "Correlation_plot.jpg",
     height = 4,width = 6,units = "in",res = 600)
pheatmap(rld_cor, color = heat.colors,fontsize = 10,
         fontsize_row = 10, height=20)
dev.off()
#==================================================================
# Get differential expression results
#==================================================================
results <- results(dds, pAdjustMethod = "fdr", alpha = 0.05)
```

```
head(results)
summary(results)
# Generate MA plot
jpeg("MA_plot.jpg", units="in",
     width=7, height=5, res = 600)
plotMA(results)
dev.off()
#================================================================
# Convert Gene Symbol to multiple IDs
#================================================================
# Human genome database (Select the correct one)
library(org.Hs.eg.db)

# Add gene full name
results$description <- mapIds(x = org.Hs.eg.db,
                              keys = row.names(results),
                              column = "GENENAME",
                              keytype = "SYMBOL",
                              multiVals = "first")

# Add ENTREZ ID
results$entrez <- mapIds(x = org.Hs.eg.db,
                         keys = row.names(results),
                         column = "ENTREZID",
                         keytype = "SYMBOL",
                         multiVals = "first")

# Add ENSEMBL
results$ensembl <- mapIds(x = org.Hs.eg.db,
                          keys = row.names(results),
                          column = "ENSEMBL",
                          keytype = "SYMBOL",
                          multiVals = "first")

head(results)
# Order by adjusted p-value
res <- results[order(results$padj), ]

# Merge with normalized count data
resdata <- merge(as.data.frame(counts(dds, normalized=TRUE)),
                 as.data.frame(res),
                 by="row.names", sort=FALSE)
```

```r
head(resdata)
names(resdata)[1] <- "Gene"
head(resdata)

#To remove rows containing NA and write as csv file
library(tidyverse)
resdata1 <- resdata %>% drop_na()
write.csv(resdata1, file="diff_KO_vs_Control.csv",
          row.names = F)

# Subset Upregulated and Downregulated genes
upreg <- resdata1 %>%
  dplyr::filter(log2FoldChange > 0 & padj < 0.05)

downreg <- resdata1 %>%
  dplyr::filter(log2FoldChange < 0 & padj < 0.05)

write.csv(upreg, file = "KO_upregulated_genes.csv",
          row.names = F)
write.csv(downreg, file = "KO_downregulated_genes.csv",
          row.names = F)

# Gather Log-fold change and FDR-corrected pvalues from DESeq2 results
# - Change pvalues to -log10 (1.3 = 0.05)
data <- data.frame(gene = row.names(res),
                   pval = -log10(res$padj),
                   lfc = res$log2FoldChange)

# Remove any rows that have NA as an entry
data <- na.omit(data)

# Color the points which are up or down log2(FC=1.5)= 0.58,
# -log10(P-adj=0.05)=1.3
## If fold-change > 0.58 and pvalue > 1.3 (Increased significant)
## If fold-change < 0.58 and pvalue > 1.3 (Decreased significant)
data <- mutate(data,
color = case_when(data$lfc > 0 & data$pval > 1.3 ~ "Increased",
                  data$lfc < 0 & data$pval > 1.3 ~ "Decreased",
                  data$pval < 1.3 ~ "nonsignificant"))


summary(data)
```

```r
head(data)
# Make a basic ggplot2 object with x-y values
vol <- ggplot(data, aes(x = lfc, y = pval, color = color))


# Add ggplot2 layers
p <- vol+
  geom_point(size = 0.5, alpha = 0.4, na.rm = T) +
  scale_color_manual(name = "Directionality",
                     values = c(Increased = "deepskyblue4",
                                Decreased = "deepskyblue2",
                                nonsignificant = "gray80")) +
  theme_classic() + # change overall theme
  theme(legend.position = "none") + # change the legend
  xlab(expression(log[2]("KO" / "Control"))) +
  # Change X-Axis label
  ylab(expression(-log[10]("adjusted p-value"))) +
  # Change Y-Axis label
  scale_y_continuous(trans = "log1p")+
  # Scale yaxis due to large p-values
  geom_hline(yintercept = 1.3,
             colour = "red",
             linetype="dashed")
  # Add p-adj value cutoff horizontal line
  #geom_vline(aes(xintercept=0.58),
              colour="gray60",
              linetype="dashed")+
  #geom_vline(aes(xintercept=-0.58),
              colour="gray60",
              linetype="dashed")+
  #xlim(-2.5, 2.5)
#Base vocano plot
p

library(ggrepel)
#If few selected genes need to be annotated in the volcano plot
p1 <- p+ geom_text_repel(data = data %>%
                           filter(gene %in% c("HES5", "RBPJ",
                                              "MKI67", "AURKB")),
                         aes(label = gene, x = lfc, y = pval),
                         box.padding = unit(.7, "lines"),
                         hjust= 0.30,
```

```
                          segment.color = 'black',
                          colour = 'black')
#View the volcano  plot
p1

#If want to plot top 10 differentially expressed genes
p2 <- p+ geom_text_repel(data=head(data, 10), aes(label=gene),
                          box.padding = unit(.5, "lines"),
                          hjust= 0.30,
                          segment.color = 'black',
                          max.overlaps = Inf,
                          colour = 'black')
p2

# Save the volcano plot
ggsave(
  "volcano_diff_KO_vs_Control.jpg",
  p1,
  width = 6.00,
  height = 5.00,
  dpi = 600
)
```

6.2 Pathway Enrichment Analysis

- **Pathway enrichment analysis** is a statistical method by which we can predict what biological pathways are enriched in a given gene list.
- There are two statistical test that can be performed.

1. Statistical over-representation test
2. Statistical enrichment test

- To know more about these tests you may refer to Nature Protocol

6.2.1 Statistical Over-representation Analysis

```
library(clusterProfiler)
library(org.Hs.eg.db)
library(enrichplot)
```

```r
library(tidyverse)
library(msigdbr)

# Import the data
res <- read.csv(file = "diff_KO_vs_Control.csv",
                header = T, row.names = 1)
data <- data.frame(gene = row.names(res),
                   pval = -log10(res$padj),
                   lfc = res$log2FoldChange)
#================================================================
# GO over-representation analysis (ORA) using enrichGO
#================================================================
#Filter the genes which are upregulated in KO
geneList <- data %>% dplyr::filter(lfc >0 & pval > 1.3)

ego <- enrichGO(gene          = geneList$gene,
                OrgDb         = org.Hs.eg.db, # or Org.Hs.eg.db
                ont           = "ALL",
                #one of "BP", "MF", "CC" or "ALL"
                pAdjustMethod = "fdr",
                #one of "bonferroni", "BH", "BY", "fdr", "none"
                pvalueCutoff  = 0.01,
                qvalueCutoff  = 0.05,
                keyType = "SYMBOL",
                #"ENSEMBL", "ENTREZID", "SYMBOL"
                readable      = TRUE)
write.csv(ego@result,
          file = "GO_upregulated_clusterprofiler.csv",
          row.names = F)
# Plot
jpeg(filename = "Upreg_enrichment.jpg",
     width = 8, height = 6, units = "in",
     res = 600)
dotplot(ego)+
labs(title = "Functional enrichment of upregulated genes")
dev.off()

# Few other plots
barplot(ego)
upsetplot(ego)

#Filter the genes which are downregulated in KO
```

```r
geneList_down <- data %>% dplyr::filter(lfc <0 & pval > 1.3)

ego_down <- enrichGO(gene          = geneList_down$gene,
                     OrgDb         = org.Hs.eg.db,
                     # or Org.Mm.eg.db
                     ont           = "ALL",
                     #one of "BP", "MF", "CC" or "ALL"
                     pAdjustMethod = "fdr",
                     #"bonferroni", "BH", "BY", "fdr", "none"
                     pvalueCutoff  = 0.01,
                     qvalueCutoff  = 0.05,
                     keyType = "SYMBOL",
                     #"ENSEMBL", "ENTREZID", "SYMBOL"
                     readable      = TRUE)

head(ego_down@result)
write.csv(ego_down@result,
          file = "GO_downregulated_clusterprofiler.csv",
          row.names = F)
##Plot
jpeg(filename = "Downreg_enrichment.jpg",
     width = 8, height = 6, units = "in",
     res = 600)
dotplot(ego_down)+
labs(title = "Functional enrichment of downregulated genes")
dev.off()

barplot(ego_down)
upsetplot(ego_down)

#=================================================================
# over-representation analysis (ORA) using MSigDb gene sets
#=================================================================
m_ont <- msigdbr(species = "Homo sapiens", category = "C5") %>%
  select(gs_name, gene_symbol)
head(m_ont)

# Upregulated genes functional enrichment analysis
em <- enricher(geneList$gene, TERM2GENE=m_ont)
head(em)

barplot(em)
```

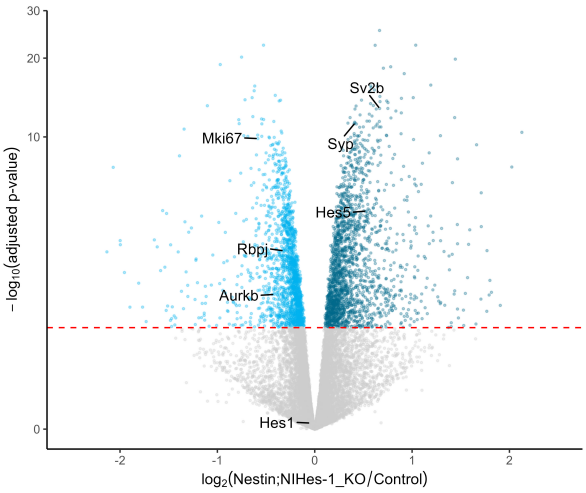
```
dotplot(em)
upsetplot(em)

# Downregulated genes functional enrichment analysis

ed <- enricher(geneList_down$gene, TERM2GENE=m_ont)
barplot(ed)
dotplot(ed)
upsetplot(ed)
```

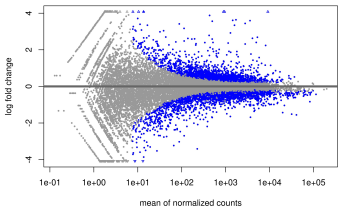
6.2.2 Statistical Enrichment Analysis

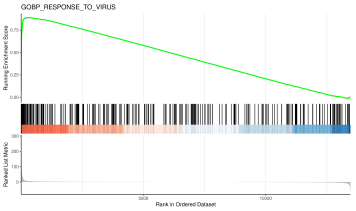
- The most used tool for statistical enrichment test is GSEA.
- To know more about GSEA you may refer to Nature Protocol
- However, here we shall perform GSEA in R which is very easy and fast.

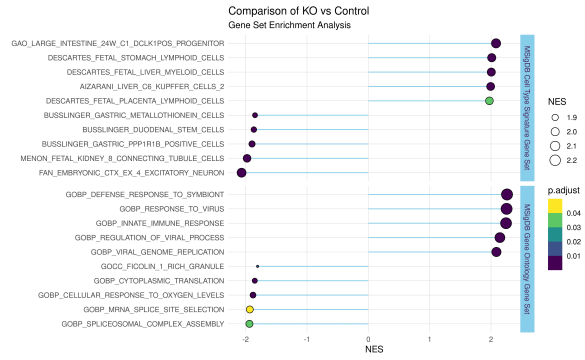
```
library(clusterProfiler)
library(enrichplot)
library(tidyverse)
library(msigdbr)

gene.list <- resdata1 %>%
  dplyr::mutate(Score = -log10(padj)* sign(log2FoldChange))%>%
  dplyr::select(symbol, Score)

# Make the rank file
ranks <- deframe(gene.list)
head(ranks)
# Set decreasing order
geneList = sort(ranks, decreasing = TRUE)
#================================================================
# Perform GSEA
#================================================================
em2 <- GSEA(geneList, TERM2GENE = m_ont)
head(em2)
gsea_result <- em2@result

# Save the GSEA result
write.csv(gsea_result,
          file = "GSEA_Ontology.csv",
```

```
          row.names = F)

# Save the GSEA Plot
jpeg(filename = "GSEA_plot.jpg",
     width = 10, height = 6, units = "in",
     res = 600)
gseaplot2(em2, geneSetID = 2, title = em2$Description[2])
dev.off()
```









6.2.3 Divergent Lollipop Chart

```
library(tidyverse)
library(msigdbr)
library(clusterProfiler)
library(enrichplot)
```

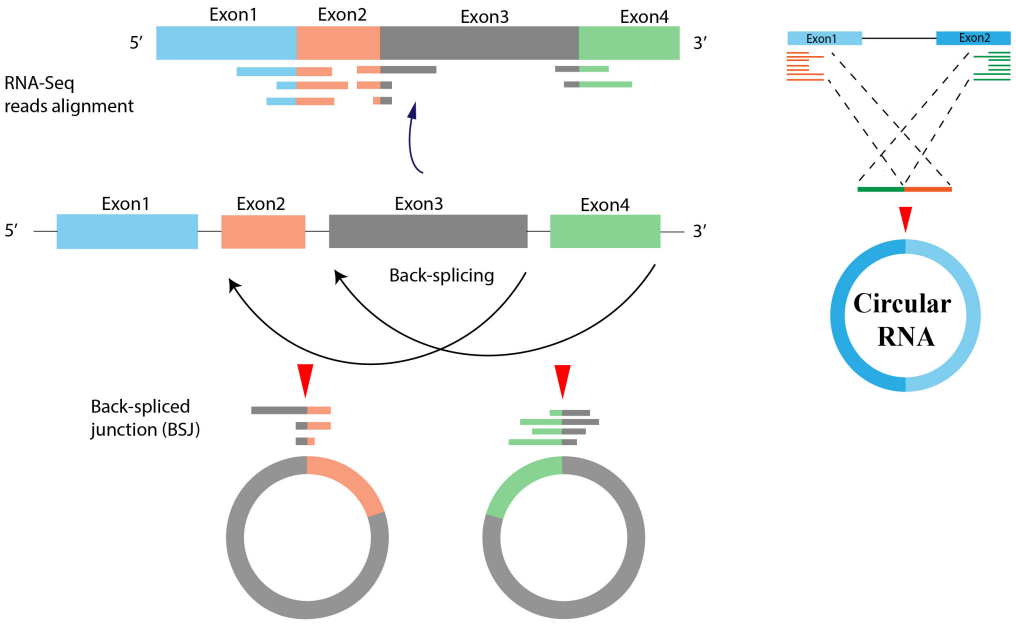
```
#================================================================
#Import the data
#================================================================
res <- read.csv(file = "diff_KO_vs_Control.csv",
                header = T)
head(res)
gene.list <- res %>%
  dplyr::mutate(Score = -log10(padj)* sign(log2FoldChange))%>%
  dplyr::select(symbol, Score)
#Make the rank file
ranks <- deframe(gene.list)
head(ranks)

#Download MSigDb ontology gene sets
#================================================================
m_ont <- msigdbr(species = "Homo sapiens",
                 category = "C5") %>%
  select(gs_name, gene_symbol)
head(m_ont)
#Download MSigDb Cell Type Signature gene sets
#================================================================
m_cell <- msigdbr(species = "Homo sapiens",
                  category = "C8") %>%
  select(gs_name, gene_symbol)
head(m_cell)
#decreasing order
geneList = sort(ranks, decreasing = TRUE)

#Perform GSEA using ONTOLOGY gene sets
#================================================================
em2 <- GSEA(geneList, TERM2GENE = m_ont)
head(em2@result)

#Perform GSEA using CELL TYPE gene sets
#================================================================
em3 <- GSEA(geneList, TERM2GENE = m_cell)
head(em3@result)

# Data wrangling
#================================================================
celltype_pos <- em3@result %>%
  mutate(gene_set = "MSigDB Cell Type Signature Gene Set")%>%
```

```
  top_n(n = 5, wt = NES)
celltype_neg <- em3@result %>%
  mutate(gene_set = "MSigDB Cell Type Signature Gene Set")%>%
  top_n(n = 5, wt = -NES)
ont_pos <- em2@result %>%
  mutate(gene_set = "MSigDB Gene Ontology Gene Set")%>%
  top_n(n = 5, wt = NES)
ont_neg <- em2@result %>%
  mutate(gene_set = "MSigDB Gene Ontology Gene Set")%>%
  top_n(n = 5, wt = -NES)

celltype_merge <- rbind(celltype_pos, celltype_neg)
GO_merge <- rbind(ont_pos, ont_neg)
my_data <- rbind(celltype_merge, GO_merge) |>
  mutate(ID = fct_reorder(ID, NES))
# Round up NES values upto 3 digits
my_data$NES <- round(my_data$NES, 3)
head(my_data)

#Divergent lollipop chart
#==================================================================
p <- ggplot(my_data,
            aes(x = ID,
                y = NES))+
  geom_segment(aes(y = 0,
                   x = ID,
                   xend = ID,
                   yend = NES),
               color = "skyblue")+
  geom_point(stat = "identity",
             aes(size = abs(NES),
                 fill = p.adjust),
             shape = 21)+
  coord_flip()+
  scale_fill_viridis_b()+
  theme_light()+
  theme(panel.border = element_blank(),
        panel.grid.minor.x = element_blank(),
        axis.ticks = element_blank(),
        strip.background = element_rect(fill = "skyblue"),
        strip.text = element_text(color = "#4a235a"))+
  labs(title = "Comparison of KO vs Control",
```

```
       subtitle = "Gene Set Enrichment Analysis",
       size = "NES",
       x = "")+
  facet_grid(gene_set ~.,space="free", scales="free")
p

#Save the file
#================================================================
jpeg(filename = "KO_GSEA_chart.jpg",
     height = 8, width = 14,
     units = "in", res = 600)
p
dev.off()
```

Part II

Circular RNA analysis



- In this section of the book, I shall introduce you to the concept of circular RNA and its identification from RNA-Seq data.
- Circular RNAs are product of back-splicing event where it forms a covalently closed circular structure of RNA.
- These class of RNA does not have 5' cap or 3' poly-A-tail.
- To identify the circular RNA from RNA-Seq data, several algorithms have been developed that mostly detect the back-splice junction (BSJ) region.
- Here, I shall introduce you with two identification tools i.e., CIRIquant and CIRCexplorer3.

7 Download GEO data

7.1 Download SRA files

- Download latest SRA-tool

```
wget --output-document sratoolkit.tar.gz\
http://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/current/sratoolkit.current-ubuntu64.tar.gz
```

- `sratoolkit.tar.gz` file will be downloaded.

```
tar -xvzf sratoolkit.tar.gz

export PATH=$PATH:/home/basu/sratoolkit.3.0.1-ubuntu64/bin
```

- Check whether the tool is working or not by executing the following command

```
prefetch -h
```

- Now let us download the following data from GEO. Transcriptional profiling of adult retinal ganglion cells during optic nerve regeneration [GSE142881: RGC injury dataset]
- Create a script file with vim editor `download.sh` and paste the following script into that.

```
vim download.sh
```

> **Note**
>
> vim download.sh → insert (press I in your keyboard) → esc → :wq → chmod +x download.sh

Bash Script

```
#!/bin/sh
# Downloading the SRA files using Prefetch command
for i in $(seq 10821165 10821205)
do
prefetch -v SRR$i --max-size 50G
done
```

```
chmod +x download.sh
```

- Execute the script by typing

```
nohup bash download.sh
```

7.2 Convert .sra file to .fastq files

```
mkdir fastq/
```

- Make another script `loop.sh` to convert .sra file to .fastq file

```
vim loop.sh
```

Bash script

```
#!/bin/sh
# convert using fasterq-dump command
for i in $(seq 10821165 10821205)
do
fasterq-dump --split-files --skip-technical -e 16 SRR$i --outdir fastq
done
```

```
chmod +x loop.sh
```

```
nohup bash loop.sh
```

7.3 Make gzip compression of the fastq files

```
cd fastq/
gzip *.fastq
```

> **Note**
>
> The next step would be checking the quality of the fastq files. See Chapter 4 for the details of fastqc commands and quality check. I would recommend to refer to that section before proceeding to quality control of fastq data.

- In the next section we are going to install packages and dependencies.

8 Install CIRIquant and CIRCexplorer3

8.1 CIRIquant

- Complete documentation can be found in CIRIquant.
- My purpose is to show some of the basic scripts to run CIRIquant repeatedly.
- Create an environment `quant` to install all dependencies and packages.

> ⚠ Warning
>
> ciriquant 1.1.2 has requirement pysam==0.15.2 ciriquant 1.1.2 has requirement PyYAML==5.1.1 ciriquant 1.1.2 has requirement python==2.7

```
conda create -n quant python=2.7 PyYAML=5.1.1 pysam=0.15.2
conda activate quant
conda install -c bioconda bwa
conda install -c bioconda hisat2
conda install -c bioconda stringtie
pip install ciriquant
```

- Make sure all the following tools are installed i.e., bwa, hisat2, samtools, stringtie
- And finally check

```
CIRIquant -h
```

8.2 CIRCexplorer3

- Create a Python 2 environment named py2, install Python 2.7
- Install circexplorer2 and required softwares (python version 2.7.1 encouraged)

```
conda create --name py2 python=2.7
conda activate py2
conda install circexplorer2 --channel bioconda
conda install -c bioconda bowtie==1.0.0
conda install -c bioconda samtools==0.1.18
conda install -c bioconda hisat2
conda install -c bioconda stringtie
conda install -c bioconda tophat==2.1.0
```

- Install CLEAR_QUANT

```
wget https://github.com/YangLab/CLEAR/archive/refs/heads/master.zip
cd CLEAR-master/
python setup.py install
```