# create data\_dir directory and subdirectories for the project:

%cd /content/drive/Shareddrives/BIO485/

%mkdir data\_dir

%cd data\_dir

%mkdir indexes

%mkdir samples

%mkdir genes

%mkdir genome

(create mergelist.txt file used later when merging transcripts from all samples with Stringtie:

ERR5684197.gtf

ERR5684198.gtf

ERR5684199.gtf

ERR5684200.gtf

ERR5684201.gtf

ERR5684202.gtf)

**\*\*\* PYTHON \*\*\***

# if use Google Colab, need to mount drive:

from google.colab import drive

drive.mount('/content/drive')

%cd /content/drive/Shareddrives/BIO485/data\_dir/

B.1 Install all the necessary packages

# First, we installed the conda lab into the Python collab to obtain other package into the environment:

!pip install -q condacolab

# Checking the version of the condacolab is optional since it would automatically install the latest version of it, but it is important to make sure the condacolab can run in the environment:

import condacolab

condacolab.install()

!conda --version

!which conda

# Next, install needed packages for the study of the comparative analysis of RNA-Seq Differential Gene Expression (DGE):

!conda install -c bioconda hisat2

!conda install -c bioconda stringtie

!conda install -c bioconda samtools

# Install SRA toolkit for SRA data download

%%bash

wget <https://ftp-trace.ncbi.nlm.nih.gov/sra/sdk/2.9.6-1/sratoolkit.2.9.6-1-ubuntu64.tar.gz>

tar xvfz sratoolkit.2.9.6-1-ubuntu64.tar.gz

B.2 SRA Data Collection:

Download 2 sets of data from SRA (species = Homo sapiens, 3 replicates each set) using SRA Toolkit

=> into *samples* directory:

* 1. Carboplatin-sensitive Breast Cancer (wild type)
  2. Carboplatin-resistant Breast Cancer

%cd /content/drive/Shareddrives/BIO485/data\_dir/samples

# Carboplatin-resistant TNBC cells:

./sratoolkit.2.9.6-1-ubuntu64/bin/fastq-dump ERR5684197

./sratoolkit.2.9.6-1-ubuntu64/bin/fastq-dump ERR5684198

./sratoolkit.2.9.6-1-ubuntu64/bin/fastq-dump ERR5684199

# Carboplatin-sensitive (WT) TNBC cells:

./sratoolkit.2.9.6-1-ubuntu64/bin/fastq-dump ERR5684200

./sratoolkit.2.9.6-1-ubuntu64/bin/fastq-dump ERR5684201

./sratoolkit.2.9.6-1-ubuntu64/bin/fastq-dump ERR5684202

B.3 Perform quality control and clean SRA/FastQ data with FastQC:

# Run fastqc for quality control of all 6 data sets --> all good scores (no low quality bp + no adapters)

!FastQC/fastqc ../samples/ERR5684197.fastq

!FastQC/fastqc ../samples/ERR5684198.fastq

!FastQC/fastqc ../samples/ERR5684199.fastq

!FastQC/fastqc ../samples/ERR5684200.fastq

!FastQC/fastqc ../samples/ERR5684201.fastq

!FastQC/fastqc ../samples/ERR5684202.fastq

B.4 Align RNA-seq reads with the human genome reference:

1. Download *Homo sapiens* genome transcripts (as reference) - the prebuilt HISAT2 indexes for the human genome from the HISAT website (into *indexes* directory):

!wget <https://genome-idx.s3.amazonaws.com/hisat/grch38_tran.tar.gz>

# redirect and unzip the reference genome transcripts

%mv grch38\_tran.tar.gz indexes/

!tar -xvzf indexes/grch38\_tran.tar.gz

1. Download FastA (DNA Seq) into *genes* directory and GTF (RefSeq Human Gene Annotations) of the reference human genome into *genome* directory:

%%bash

# Download the reference genome

cd /content/drive/Shareddrives/BIO485/data\_dir/genome/

wget ftp://[ftp.ensembl.org/pub/release-84/fasta/homo\_sapiens/dna/Homo\_sapiens.GRCh38.dna.primary\_assembly.fa.gz](http://ftp.ensembl.org/pub/release-84/fasta/homo_sapiens/dna/Homo_sapiens.GRCh38.dna.primary_assembly.fa.gz)

gzip -d Homo\_sapiens.GRCh38.dna.primary\_assembly.fa.gz

mv Homo\_sapiens.GRCh38.dna.primary\_assembly.fa genome.fa

cd /content/drive/Shareddrives/BIO485/data\_dir/genes/

wget ftp://[ftp.ensembl.org/pub/release-84/gtf/homo\_sapiens/Homo\_sapiens.GRCh38.84.gtf.gz](http://ftp.ensembl.org/pub/release-84/gtf/homo_sapiens/Homo_sapiens.GRCh38.84.gtf.gz)

gzip -d Homo\_sapiens.GRCh38.84.gtf.gz

mv Homo\_sapiens.GRCh38.84.gtf genome.gtf

1. Alignment using Hisat2:

%cd /content/drive/Shareddrives/BIO485/data\_dir/

# Alignment the RNAseq reads (obtained directly from SRA) with Homo sapiens reference genome using HISAT2

!hisat2 -p 8 --dta -x indexes/grch38\_tran/genome\_tran -U samples/ERR5684197.fastq ERR5684197.sam

!hisat2 -p 8 --dta -x indexes/grch38\_tran/genome\_tran -U samples/ERR5684198.fastq ERR5684198.sam

!hisat2 -p 8 --dta -x indexes/grch38\_tran/genome\_tran -U samples/ERR5684199.fastq ERR5684199.sam

!hisat2 -p 8 --dta -x indexes/grch38\_tran/genome\_tran -U samples/ERR5684200.fastq ERR5684200.sam

!hisat2 -p 8 --dta -x indexes/grch38\_tran/genome\_tran -U samples/ERR5684201.fastq ERR5684201.sam

!hisat2 -p 8 --dta -x indexes/grch38\_tran/genome\_tran -U samples/ERR5684202.fastq ERR5684202.sam

B.5 Sort and convert SAM to BAM then assemble transcripts using StringTie:

1. Sort and convert SAM to BAM:

!samtools sort -@ 8 -o outputs/ERR5684197.bam outputs/ERR5684197.sam

!samtools sort -@ 8 -o outputs/ERR5684198.bam outputs/ERR5684198.sam

!samtools sort -@ 8 -o outputs/ERR5684199.bam outputs/ERR5684199.sam

!samtools sort -@ 8 -o outputs/ERR5684200.bam outputs/ERR5684200.sam

!samtools sort -@ 8 -o outputs/ERR5684201.bam outputs/ERR5684201.sam

!samtools sort -@ 8 -o outputs/ERR5684202.bam outputs/ERR5684202.sam

1. Transcript assembly using StringTie:

!stringtie -p 8 -G genes/genome.gtf -o outputs/ERR5684197.gtf outputs/ERR5684197.bam

!stringtie -p 8 -G genes/genome.gtf -o outputs/ERR5684198.gtf outputs/ERR5684198.bam

!stringtie -p 8 -G genes/genome.gtf -o outputs/ERR5684199.gtf outputs/ERR5684199.bam

!stringtie -p 8 -G genes/genome.gtf -o outputs/ERR5684200.gtf outputs/ERR5684200.bam

!stringtie -p 8 -G genes/genome.gtf -o outputs/ERR5684201.gtf outputs/ERR5684201.bam

!stringtie -p 8 -G genes/genome.gtf -o outputs/ERR5684202.gtf outputs/ERR5684202.bam

1. Merge transcripts from all samples

%cd /content/drive/Shareddrives/BIO485/data\_dir/outputs/

!stringtie --merge -p 8 -G ../genes/genome.gtf -o stringtie\_merged.gtf ../mergelist.txt

B.6 Quantify and compare gene/transcript expression levels using Stringtie:

# Quantification using StringTie

!stringtie -e -B -p 8 -G stringtie\_merged.gtf -o ballgown/ERR5684197/ERR5684197.gtf ERR5684197.bam

!stringtie -e -B -p 8 -G stringtie\_merged.gtf -o ballgown/ERR5684198/ERR5684198.gtf ERR5684198.bam

!stringtie -e -B -p 8 -G stringtie\_merged.gtf -o ballgown/ERR5684199/ERR5684199.gtf ERR5684199.bam

!stringtie -e -B -p 8 -G stringtie\_merged.gtf -o ballgown/ERR5684200/ERR5684200.gtf ERR5684200.bam

!stringtie -e -B -p 8 -G stringtie\_merged.gtf -o ballgown/ERR5684201/ERR5684201.gtf ERR5684201.bam

!stringtie -e -B -p 8 -G stringtie\_merged.gtf -o ballgown/ERR5684202/ERR5684202.gtf ERR5684202.bam

B.7 Filtering out repetitive gene IDs that have N/A gene names (after getting tsv files of DE gene list from R Ballgown):

%cd /content/drive/Shareddrives/BIO485/data\_dir/outputs/

geneID\_name = {} # Create a dictionary for named Gene IDs and Names geneID\_na = {} # Create a dictionary for unnamed Gene IDs and Names

# Read the DE gene file to start the filtering

with open('CR\_vs\_CS\_gene\_results\_sig.tsv') as tsv:

for line in tsv:

line = line.rstrip()

if line:

row = line.split()

if row[5] == ".": # the gene has no name

geneID\_na[row[0]] = row[5]

else:

geneID\_name[row[0]] = row[5]

noName = list(set(geneID\_na) - set(geneID\_name)) # get a list of genes that actually do not have names

# start writing the output file with named genes and truly unnamed genes

outTxt = ""

with open('CR\_vs\_CS\_gene\_results\_sig.tsv') as tsv:

for line in tsv:

line = line.rstrip()

if line:

row = line.split()

if row[5] != "." or row[0] in noName:

outTxt += line + "\n"

with open('CRCS\_sig\_genes\_named.tsv', "w") as outF:

outF.write(outTxt)

**\*\*\* R \*\*\***

# if need to load and use R in Google Colab with python runtime

%load\_ext rpy2.ipython

B.1 Install and import all the necessary packages

%%R

if (!requireNamespace("BiocManager", quietly=TRUE))

install.packages("BiocManager")

BiocManager::install("ballgown")

BiocManager::install("EBSeq")

install.packages('pheatmap')

install.packages('RColorBrewer')

install.packages('reshape2')

%%R

library(ballgown)

library(EBSeq)

library(genefilter)

library(dplyr)

library(devtools)

devtools::install\_github('alyssafrazee/RSkittleBrewer')

install.packages('ggrepel')

library(ggrepel)

library(RSkittleBrewer)

library(ggplot2)

library(gplots)

library(GenomicRanges)

library("pheatmap")

library("RColorBrewer")

library("reshape2")

B.2 Create a Ballgown object and save it for future use

%%R

# set working directory

setwd("/content/drive/Shareddrives/BIO485/data\_dir/outputs")

%%R

# create phenotype dataframe needed for the ballgown object

ids <- c("ERR5684197","ERR5684198","ERR5684199","ERR5684200","ERR5684201","ERR5684202")

therapy\_response <- c('resistant','resistant','resistant','control','control','control')

pheno <- data.frame(ids,therapy\_response)

%%R

bg\_TNBC = ballgown(dataDir = "./ballgown", samplePattern = "ERR5684", pData = pheno) # create a Ballgown object: 217405 transcripts & 6 samples

%%R

# Save the ballgown object to a file for later use

save(bg\_TNBC, file='bg.rda')

B.3 Quality Control between Replicates

%%R

# Load the ballgown object to start quality control

load('bg.rda')

%%R

# Examine the variance in FPKM across 6 datasets

bg\_filt = subset (bg\_TNBC,"rowVars(texpr(bg\_TNBC)) > 1", genomesubset=TRUE) # Filter out low-abundance genes

gene\_expression = as.data.frame(gexpr(bg\_filt))

colnames(gene\_expression) <- c("ERR5684197","ERR5684198","ERR5684199","ERR5684200","ERR5684201","ERR5684202")

data\_colors=c("tomato1","tomato2","tomato3","wheat1","wheat2","wheat3")

data\_columns=c(1:6)

short\_names=c("res\_1","res\_2","res\_3","sen\_1","sen\_2","sen\_3") # res is short for resistant, while sen is short for sensitive boxplot(log2(gene\_expression[,data\_columns]+1), col=data\_colors, names=short\_names, las=2, ylab="log2(FPKM)", main="Distribution of FPKMs for all 6 samples")

%%R

# Examine the variance in FPKM between replicates within resistant group

x = gene\_expression[,"ERR5684197"]

y = gene\_expression[,"ERR5684198"]

z = gene\_expression[,"ERR5684199"]

plot(x=log2(x+1), y=log2(y+1), pch=16, col="blue", cex=0.25, xlab="FPKM (ERR5684197)", ylab="FPKM (ERR5684198)", main="Comparison of expression values for a pair of replicates within resistant strain")

abline(a=0,b=1)

rs=cor(x,y)^2

legend("topleft", paste("R squared = ", round(rs, digits=3), sep=""), lwd=1, col="black")

plot(x=log2(x+1), y=log2(z+1), pch=16, col="blue", cex=0.25, xlab="FPKM (ERR5684197)", ylab="FPKM (ERR5684199)", main="Comparison of expression values for a pair of replicates within resistant strain")

abline(a=0,b=1)

rs=cor(x,y)^2

legend("topleft", paste("R squared = ", round(rs, digits=3), sep=""), lwd=1, col="black")

plot(x=log2(z+1), y=log2(y+1), pch=16, col="blue", cex=0.25, xlab="FPKM (ERR5684199)", ylab="FPKM (ERR5684198)", main="Comparison of expression values for a pair of replicates within resistant strain")

abline(a=0,b=1)

rs=cor(x,y)^2

legend("topleft", paste("R squared = ", round(rs, digits=3), sep=""), lwd=1, col="black")

%%R

# Examine the variance in FPKM between replicates within resistant group

a = gene\_expression[,"ERR5684200"]

b = gene\_expression[,"ERR5684201"]

c = gene\_expression[,"ERR5684202"]

plot(x=log2(a+1), y=log2(b+1), pch=16, col="green", cex=0.25, xlab="FPKM (ERR5684200)", ylab="FPKM (ERR5684201)", main="Comparison of expression values for a pair of replicates within sensitive strain")

abline(a=0,b=1)

rs=cor(x,y)^2

legend("topleft", paste("R squared = ", round(rs, digits=3), sep=""), lwd=1, col="black")

plot(x=log2(a+1), y=log2(c+1), pch=16, col="green", cex=0.25, xlab="FPKM (ERR5684200)", ylab="FPKM (ERR5684202)", main="Comparison of expression values for a pair of replicates within sensitive strain")

abline(a=0,b=1)

rs=cor(x,y)^2

legend("topleft", paste("R squared = ", round(rs, digits=3), sep=""), lwd=1, col="black")

plot(x=log2(c+1), y=log2(b+1), pch=16, col="green", cex=0.25, xlab="FPKM (ERR5684202)", ylab="FPKM (ERR5684201)", main="Comparison of expression values for a pair of replicates within sensitive strain")

abline(a=0,b=1)

rs=cor(x,y)^2

legend("topleft", paste("R squared = ", round(rs, digits=3), sep=""), lwd=1, col="black")

%%R

# Heatmap of gene expression pairwise comparison for all 6 datasets

gene\_expression[,"sum"]=apply(gene\_expression[,data\_columns], 1, sum)

sig\_gene = which(gene\_expression[,"sum"] > 5)

r\_val = cor(gene\_expression[sig\_gene,data\_columns], use="pairwise.complete.obs", method="pearson")

melted\_cormat <- melt(r\_val)

ggplot(data = melted\_cormat, aes(x=Var1, y=Var2, fill=value)) +

geom\_tile()

B.4 Identify Differentially Expressed Genes and Transcripts with Ballgown

%%R

# Load the ballgown object to start identifying DE genes and transcripts

load('bg.rda')

%%R

# Load all attributes including gene and transcript names

bg\_table = texpr(bg\_TNBC, 'all')

bg\_gene\_names = unique(bg\_table[, 9:10])

bg\_transcript\_names = unique(bg\_table[, c(1,6,10)])

%%R

# Filter low-abundance genes. Here we remove all transcripts with a variance across the samples of less than one

bg\_filt = subset (bg\_TNBC,"rowVars(texpr(bg\_TNBC)) > 1", genomesubset=TRUE) # After filter: 35227 transcripts and 6 samples

%%R

# Perform DE analysis now using the filtered data

results\_transcripts = stattest(bg\_filt, feature="transcript", covariate="therapy\_response", getFC=TRUE, meas="FPKM")

results\_genes = stattest(bg\_filt, feature="gene", covariate="therapy\_response", getFC=TRUE, meas="FPKM")

results\_genes = merge(results\_genes, bg\_gene\_names, by.x=c("id"), by.y=c("gene\_id"))

results\_transcripts = merge(results\_transcripts, bg\_transcript\_names, by.x=c("id"), by.y=c("t\_id"))

%%R

# Output the filtered list of genes and transcripts and save to tab delimited files

write.table(results\_transcripts, "CR\_vs\_CS\_transcript\_results\_filtered.tsv", sep="\t", quote=FALSE, row.names = FALSE)

write.table(results\_genes, "CR\_vs\_CS\_gene\_results\_filtered.tsv", sep="\t", quote=FALSE, row.names = FALSE)

%%R

# Identify the significant DE genes with q-value < 0.05

sig\_transcripts = subset(results\_transcripts, results\_transcripts$qval<0.05 & (results\_transcripts$fc<=1/1.5 | results\_transcripts$fc>=1.5))

sig\_genes = subset(results\_genes, results\_genes$qval<0.05 & (results\_genes$fc<=1/1.5 | results\_genes$fc>=1.5))

# Sort the results from most up-regulated to most down-regulated:

sig\_transcripts = arrange(sig\_transcripts, fc)

sig\_genes = arrange(sig\_genes, fc)

%%R

# Display statistical results

print(dim(results\_genes))

print(dim(results\_transcripts))

print(dim(sig\_genes))

print(dim(sig\_transcripts))

%%R

# Output the significant gene results to a pair of tab delimited files

write.table(sig\_transcripts, "CR\_vs\_CS\_transcript\_results\_sig.tsv", sep="\t", quote=FALSE, row.names = FALSE)

write.table(sig\_genes, "CR\_vs\_CS\_gene\_results\_sig.tsv", sep="\t", quote=FALSE, row.names = FALSE)

B.5 Visualize Differentially Expressed Genes and Transcripts

results\_genes[,"de"] = log2(results\_genes[,"fc"])

plot(-log10(qval) ~ de, data = results\_genes, pch = 21, main = "Volcano plot of differentially expressed genes", xlab = "log2 Fold Change", ylab = "-log10(P-Value)", bg = "darkgrey", xlim = c(-5.5, 5.5), ylim=c(1.2, 2.7), cex.lab = 1.3, cex.axis = 1.3)

points(-log10(qval) ~ de, data = results\_genes[results\_genes$qval<0.05,], bg = "orange", pch = 21)

points(-log10(qval) ~ de, data = results\_genes[results\_genes$qval<0.05&(df$de)>= 0.58496250072,], bg = "green", pch = 21)

points(-log10(qval) ~ de, data = results\_genes[results\_genes$qval<0.05&(df$de)<= -0.58496250072,], bg = "red", pch = 21)

legend("topright", legend = c("FDR > 0.05","FDR < 0.05 and 0.667 < FC < 1.5", "FDR < 0.05 and FC ≥ 1.5", "FDR < 0.05 and FC ≤ 0.667"), pt.bg = c("darkgrey", "orange", "green", "red"), pch = 21, cex = 1.0)

%%R

# plot mean expression level of all transcripts of top DE genes in Wnt

# myc gene

plotMeans('MSTRG.27098', bg\_filt, groupvar="therapy\_response",legend=FALSE)

# jun gene

plotMeans('MSTRG.1105', bg\_filt, groupvar="therapy\_response",legend=FALSE)

# ercc2 gene

plotMeans('MSTRG.14874', bg\_filt, groupvar="therapy\_response",legend=FALSE)

# pin1 gene

plotMeans('MSTRG.13974', bg\_filt, groupvar="therapy\_response",legend=FALSE)

%%R

# Heatmap of FPKM pairwise comparison for DE genes in the Beta-catenin/Wnt signaling pathway: jun, myc, sfrp1, tcf7, fzd7, sox9, ruvbl1, numb, ccnd1, smarca4, pin1, wnt6

wnt\_geneid = c('MSTRG.17160','MSTRG.5002','MSTRG.13974','MSTRG.14004','MSTRG.20128','MSTRG.26530','MSTRG.8404','MSTRG.27098','MSTRG.12785','MSTRG.22766','MSTRG.1105','MSTRG.16987')

sig\_gene = which(rownames(gene\_expression) %in% wnt\_genesid)

data\_columns = c(1:6)

r\_val = cor(t(gene\_expression[sig\_gene,data\_columns]), use="pairwise.complete.obs", method="pearson")

pheatmap(r\_val)

**\*\*\* GO Functional Analysis and Visualization \*\*\***

B.1 gProfiler:

Go to: <https://biit.cs.ut.ee/gprofiler/gost>

Click “Search”

* Enter the list of DE gene names under “Query”
* Choose “Organism:” to be Homo sapiens (Human), then press “Run query”
* Based on the resulted graph, choose to focus on significant (abundant) categories and significant (higher score) genes for literature research of their functional pathways

B.2 STRING:

* Go to: <https://string-db.org/>
* Click “Search”
  + Enter the list of DE gene names under “Protein Name:”
  + Choose “Organism:” to be Homo sapiens, then press “Search”
* For exploring, grouping genes into functional pathways, and narrowing down the list of gene targets for further analyses, choose the option “Clusters”
  + Choose “kmeans clustering”
  + Set “number of clusters” (e.g., 10)
  + Then press “Apply”
* For exploring groups of functional pathways, choose the option “Analysis”
  + Highlight relevant and significant pathways (higher count, lower false discovery rate)
  + Select genes that are in several of the chosen pathways (likely to be important genes)
  + Literature research for the functions, locations, and regulations of these genes
  + For example: Wnt/Beta catenin signaling pathway; Genes: *myc, ercc2, jun*, etc.