RNASeq Analysis DESeq2 Pipeline Notebook

Bishal Paudel

01/24/2018

This is a pipeline for differential analysis of RNASeq data from SKMEL5 sublines using DESeq2 statistical package. Three sublines: SC01 (*regressing*), SC07 (*stationary*) and SC10 (*expanding*) were analyzed for gene expression differences. In addition, time course changes in 8uM PLX4720 were also performed for each subline. Time points are: 0, 3d, 8d. The differential analysis will be performed based on the contrasts defined below. General steps for the analysis are:

1. Read counts table:

- Could be read directly as a csv/txt file.
- Alignment and read counts could be done within R environment to create read counts table.

1. Define working directory, load the required libraries.

```
require(knitr)
```

```
## Loading required package: knitr
```

```
opts_knit$set(root.dir = '~/RNA_Seq_Samples/Synergy_paper/data')
```

2. Get read counts table. Read the raw counts file processed by featureCounts. The fastq files were aligned with HiSat2, and the read counts were obtained using featureCounts of Rsubread packages.

```
d <- read.csv("featureCounts_matrix_all.csv", header=T, sep=",")
# get the size of the object d, the first column is Geneid, and the rest are raw counts.
dim(d)
```

```
## [1] 60675    28
```

```
# Get the countdata from the csv.This is the table with the fragment counts.
countdata <- d[,-1]
baseline <- c(1,2,3,10,11,12,19,20,21)
treat3d  <- c(4,5,6,13,14,15,22,23,24)
treat8d  <- c(7,8,9,16,17,18,25,26,27)
# define the groups by subclones
sc01 <- c(baseline[1:3], treat3d[1:3], treat8d[1:3])
sc07 <- c(baseline[4:6], treat3d[4:6], treat8d[4:6])
sc10 <- c(baseline[7:9], treat3d[7:9], treat8d[7:9])
# Get the countdata specific to conditions:
input <- c(baseline)
countdata <- countdata[,c(input)]
rownames(countdata) <- d[,"Geneid"]
head(countdata)
```

```
##                 X3345.BP.33 X3345.BP.34 X3345.BP.35 X3345.BP.42
## ENSG00000223972           0           0           0           0
## ENSG00000227232          57          82          83          73
## ENSG00000278267          18          15          21          28
## ENSG00000243485           0           0           0           1
## ENSG00000274890           0           0           0           0
## ENSG00000237613           0           0           0           0
```

```
##                 X3345.BP.43 X3345.BP.44 X3345.BP.51 X3345.BP.52
## ENSG00000223972           0           0           0           0
## ENSG00000227232          71          62          70          31
## ENSG00000278267          30          23           9           7
## ENSG00000243485           0           0           0           0
## ENSG00000274890           0           0           0           0
## ENSG00000237613           0           0           0           0
##                 X3345.BP.53
## ENSG00000223972           0
## ENSG00000227232          48
## ENSG00000278267           9
## ENSG00000243485           0
## ENSG00000274890           0
## ENSG00000237613           0
```

```
ncol(countdata)
```

```
## [1] 9
```

2. Convert counts table to DESeq2 object.

Convert counts table to object for DESeq2 or any other analysis pipeline. This step will require to prepare data object in a form that is suitable for analysis in DESeq2 pipeline: we will need the following to proceed:

- countdata: a table with the read/fragment counts.
- coldata: a table with information about the samples.

Using the matrix of counts and the sample information table, we need to construct the DESeqDataSet object, for which we will use DESeqDataSetFromMatrix. . . ..

1. Define the samples and treatment conditions.

```
condition <- c("0", "3d", "8d")
treatment <- rep(condition, each=3) # Three biological replicates
unique(treatment)
```

```
## [1] "0"  "3d" "8d"
```

```
cell <- c("SC01", "SC07","SC10") #sublines used for the analysis
cellName <- c(cell)
treatment <- c(unique(treatment)[1])

if(length(cellName)==3) cellName <- rep(cellName, each=length(cellName)) else cellName <- rep(cellName,
coldata <- data.frame(cell=rep(cellName), treatment=rep(treatment, each=9))
cellName
```

```
## [1] "SC01" "SC01" "SC01" "SC07" "SC07" "SC07" "SC10" "SC10" "SC10"
```

```
coldata
```

```
##   cell treatment
## 1 SC01         0
## 2 SC01         0
## 3 SC01         0
## 4 SC07         0
## 5 SC07         0
## 6 SC07         0
```

```
## 7 SC10         0
## 8 SC10         0
## 9 SC10         0
```

```
treatment
```

```
## [1] "0"
```

```
group = factor(paste(coldata$cell, coldata$treatment, sep="."))
coldata$group = group
coldata
```

```
##   cell treatment  group
## 1 SC01         0 SC01.0
## 2 SC01         0 SC01.0
## 3 SC01         0 SC01.0
## 4 SC07         0 SC07.0
## 5 SC07         0 SC07.0
## 6 SC07         0 SC07.0
## 7 SC10         0 SC10.0
## 8 SC10         0 SC10.0
## 9 SC10         0 SC10.0
```

2. construct the DESeqDataSet object from the matrix of counts and the sample information table.

Described above are: countdata- raw counts, coldata: sample information table.

```
library(DESeq2)
```

```
## Warning: package 'DESeq2' was built under R version 3.3.1
```

```
## Loading required package: S4Vectors
```

```
## Warning: package 'S4Vectors' was built under R version 3.3.1
```

```
## Loading required package: stats4
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, cbind, colnames,
##     do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, lengths, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
```

```
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff,
##     sort, table, tapply, union, unique, unsplit
##
## Attaching package: 'S4Vectors'
## The following objects are masked from 'package:base':
##
##     colMeans, colSums, expand.grid, rowMeans, rowSums
## Loading required package: IRanges
## Loading required package: GenomicRanges
## Warning: package 'GenomicRanges' was built under R version 3.3.1
## Loading required package: GenomeInfoDb
## Warning: package 'GenomeInfoDb' was built under R version 3.3.1
## Loading required package: SummarizedExperiment
## Loading required package: Biobase
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
dds <- DESeqDataSetFromMatrix(countData = countdata,
                              colData = coldata,
                              design = ~ group)
dds
```

```
## class: DESeqDataSet
## dim: 60675 9
## metadata(1): version
## assays(1): counts
## rownames(60675): ENSG00000223972 ENSG00000227232 ...
##   ENSG00000276666 ENSG00000277917
## rowData names(0):
## colnames(9): X3345.BP.33 X3345.BP.34 ... X3345.BP.52 X3345.BP.53
## colData names(3): cell treatment group
```

```
nrow(dds); ncol(dds)
```

```
## [1] 60675
## [1] 9
```

3. Exploratory analysis and visualization.

There are two separate steps in the workflow; the one which involves data transformations in order to visualize sample relationships and the second step involves statistical testing methods which requires the original raw counts.

1. Pre-filtering and normalization.

Pre-filtering and normalization is required to remove lowly expressed genes.

```
dds <- dds[rowSums(counts(dds)) > 9, ] # remove rows with minimum of 1 reads per sample
nrow(dds)
```

```
## [1] 25131
```

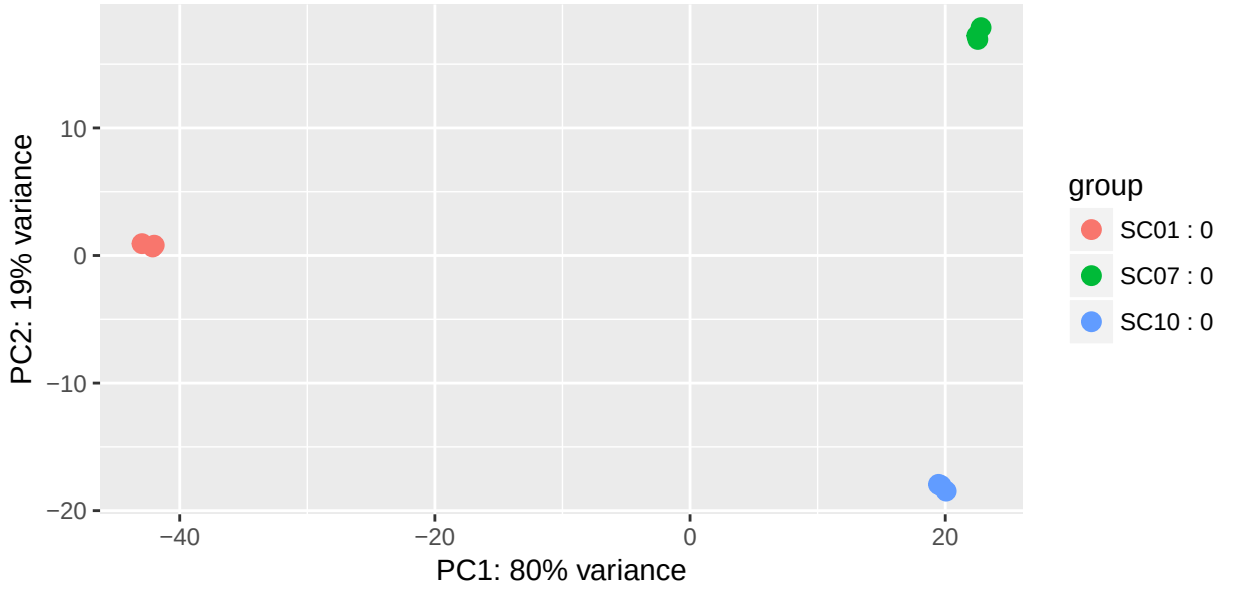
2. Visualize sample-to-sample distances.

We could use Principal Component Analysis (PCA) to visualize relationships between samples.

```
rld <- rlog(dds, blind = FALSE)
head(assay(rld), 3)
```

```
##                 X3345.BP.33 X3345.BP.34 X3345.BP.35 X3345.BP.42
## ENSG00000227232    5.832054   6.2588817    6.223661    6.125971
## ENSG00000278267    4.080866   4.0650069    4.182232    4.333334
## ENSG00000241860    1.060146   0.9758038    1.066954    1.097917
##                 X3345.BP.43 X3345.BP.44 X3345.BP.51 X3345.BP.52
## ENSG00000227232    5.961342   5.9692961    5.998354    5.464597
## ENSG00000278267    4.289035   4.2236795    3.863081    3.830795
## ENSG00000241860    1.000285   0.9746176    1.031689    1.123221
##                 X3345.BP.53
## ENSG00000227232    5.760028
## ENSG00000278267    3.886562
## ENSG00000241860    1.005788
```

```
plotPCA(rld, intgroup = c("cell", "treatment"), ntop=1000)
```



4. Differential Expression Analysis.

Always make sure to use the unnormalized raw counts for this. We will use DESeq function to perform differential analysis between samples; Unless specified, the analysis is between the last group and the first group. Different comparison can be done using 'contrast' argument. Steps involved underneath:

1. estimation of size factors (controls for differences in sequencing depth of the samples)
2. estimation of dispersion values for each gene,

3. fitting a generalized linear model

1. Running the differential expression pipeline.

```
design(dds) = ~ group
dds <- DESeq(dds, test = "LRT", reduced = ~ 1)
```

```
## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
```

```
dds
```

```
## class: DESeqDataSet
## dim: 25131 9
## metadata(1): version
## assays(3): counts mu cooks
## rownames(25131): ENSG00000227232 ENSG00000278267 ...
##   ENSG00000277856 ENSG00000271254
## rowData names(22): baseMean baseVar ... deviance maxCooks
## colnames(9): X3345.BP.33 X3345.BP.34 ... X3345.BP.52 X3345.BP.53
## colData names(4): cell treatment group sizeFactor
```

2. Building the results table.

By default, results will extract the estimated log2 fold changes and p values for the last variable in the design formula. If there are more than 2 levels for this variable, results will extract the results table for a comparison of the last level over the first level.

```
# Esimate the differences between groups by: # a) Lowering the FDR (padj) or (b) raise the log2 fold ch
res <- results(dds, alpha = 0.001)
res
```

```
## log2 fold change (MLE): group SC10.0 vs SC01.0
## LRT p-value: '~ group' vs '~ 1'
## DataFrame with 25131 rows and 6 columns
##                    baseMean log2FoldChange     lfcSE       stat
##                   <numeric>      <numeric> <numeric>  <numeric>
## ENSG00000227232  63.639082     -0.6318222 0.2745233  5.5602855
## ENSG00000278267  17.541919     -1.1470364 0.4812803 13.4585855
## ENSG00000241860   2.092274      0.4149772 1.3615897  0.2434660
## ENSG00000279928   3.344199      0.8042648 1.1924565  0.8190057
## ENSG00000279457 257.023046     -0.4915496 0.1705423  8.2964089
## ...                    ...            ...       ...        ...
## ENSG00000278384 131.788608     -0.8478057 0.1788315 22.8360167
## ENSG00000278066   3.423414     -1.3905271 1.1632465  2.3697919
## ENSG00000276345  25.405664     -0.2247674 0.3988365  0.6390537
## ENSG00000277856   1.572396      1.4440233 2.1197838  2.8081118
## ENSG00000271254 256.450479     -0.3238889 0.1272563  6.7669332
##                      pvalue         padj
```

```
##                    <numeric>    <numeric>
## ENSG00000227232  0.062029653  0.100155526
## ENSG00000278267  0.001195378  0.002544371
## ENSG00000241860  0.885384735  0.920254490
## ENSG00000279928  0.663980261  0.743123380
## ENSG00000279457  0.015792748  0.028791384
## ...                      ...          ...
## ENSG00000278384 1.099568e-05 2.881496e-05
## ENSG00000278066 3.057780e-01 4.005900e-01
## ENSG00000276345 7.264927e-01 7.948669e-01
## ENSG00000277856 2.455988e-01 3.332614e-01
## ENSG00000271254 3.392963e-02 5.808060e-02
```

```
summary(res)
```

```
##
## out of 25131 with nonzero total read count
## adjusted p-value < 0.001
## LFC > 0 (up)     : 5618, 22%
## LFC < 0 (down)   : 5004, 20%
## outliers [1]     : 0, 0%
## low counts [2]   : 1462, 5.8%
## (mean count < 2)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
sdg = subset(res, res$padj <= 0.001)
sdg = data.frame(sdg)
sdg$value = -log10(sdg$padj * sdg$log2FoldChange)
```

```
## Warning: NaNs produced
```

```
sdg = data.frame(sdg)
sdg = sdg[(order(sdg$value, decreasing = T)),]
sdg = sdg[1:1000, ]
dim(sdg)
```

```
## [1] 1000    7
```

```
head(sdg)
```

```
##                   baseMean log2FoldChange      lfcSE     stat pvalue padj
## ENSG00000117115   859.5700      1.8230954 0.10286286 2080.726      0    0
## ENSG00000163873   227.1553     -8.2037137 0.55677218 1858.881      0    0
## ENSG00000143127  1653.7939     -1.7949424 0.07962728 3068.395      0    0
## ENSG00000163565 10289.4005     -0.5173227 0.04808421 1595.963      0    0
## ENSG00000162706   990.7020     -5.0639667 0.10928615 5969.131      0    0
## ENSG00000143153   776.9644      3.5064250 0.09865793 1497.709      0    0
##                  value
## ENSG00000117115    Inf
## ENSG00000163873    Inf
## ENSG00000143127    Inf
## ENSG00000163565    Inf
## ENSG00000162706    Inf
## ENSG00000143153    Inf
```

5. Annotate the geneID and obtain the gene symbols.

Annotating the geneId

```
library("AnnotationDbi")
```

```
## Warning: package 'AnnotationDbi' was built under R version 3.3.1
```

```
library("org.Hs.eg.db")
```

```
##
```

```
columns(org.Hs.eg.db)
```

```
##  [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"
##  [5] "ENSEMBLTRANS" "ENTREZID"     "ENZYME"       "EVIDENCE"
##  [9] "EVIDENCEALL"  "GENENAME"     "GO"           "GOALL"
## [13] "IPI"          "MAP"          "OMIM"         "ONTOLOGY"
## [17] "ONTOLOGYALL"  "PATH"         "PFAM"         "PMID"
## [21] "PROSITE"      "REFSEQ"       "SYMBOL"       "UCSCKG"
## [25] "UNIGENE"      "UNIPROT"
```

```
sdg$symbol <- mapIds(org.Hs.eg.db,
                     keys=row.names(sdg),
                     column="SYMBOL",
                     keytype="ENSEMBL",
                     multiVals="first")
```

```
## 'select()' returned 1:many mapping between keys and columns
```

```
sdg = na.omit(sdg)
head(sdg, 20)
```

```
##                   baseMean log2FoldChange      lfcSE     stat pvalue padj
## ENSG00000117115   859.5700      1.8230954 0.10286286 2080.726      0    0
## ENSG00000163873   227.1553     -8.2037137 0.55677218 1858.881      0    0
## ENSG00000143127  1653.7939     -1.7949424 0.07962728 3068.395      0    0
## ENSG00000163565 10289.4005     -0.5173227 0.04808421 1595.963      0    0
## ENSG00000162706   990.7020     -5.0639667 0.10928615 5969.131      0    0
## ENSG00000143153   776.9644      3.5064250 0.09865793 1497.709      0    0
## ENSG00000163395   256.0031     -5.8288130 0.25407238 1683.333      0    0
## ENSG00000188783  3219.1991     -3.6271047 0.05785690 6500.216      0    0
## ENSG00000130508   764.8821     -8.3779098 0.32278740 5589.519      0    0
## ENSG00000115758  3102.2228      2.1406530 0.05476644 1669.782      0    0
## ENSG00000152689  1318.1740      3.2081701 0.07688759 1874.274      0    0
## ENSG00000115380   872.1015     -8.5650705 0.32208564 6112.682      0    0
## ENSG00000014641 21539.7878      1.9298195 0.04378791 2138.848      0    0
## ENSG00000115318  1744.0737     -5.2765914 0.09412165 7203.576      0    0
## ENSG00000159399  4451.2766     -1.3698930 0.04605979 1833.532      0    0
## ENSG00000115423  4930.1723      2.4235376 0.06072895 1514.734      0    0
## ENSG00000153208  8015.1277      1.7926268 0.04338645 1982.314      0    0
## ENSG00000081479   775.1537     -2.6905948 0.09309084 1929.362      0    0
## ENSG00000114948  3064.1532      1.2705760 0.04658376 1958.881      0    0
## ENSG00000138413 16036.8866      1.1161429 0.04051251 1769.224      0    0
##                  value  symbol
## ENSG00000117115    Inf   PADI2
## ENSG00000163873    Inf   GRIK3
## ENSG00000143127    Inf  ITGA10
```

```
## ENSG00000163565   Inf   IFI16
## ENSG00000162706   Inf   CADM3
## ENSG00000143153   Inf  ATP1B1
## ENSG00000163395   Inf   IGFN1
## ENSG00000188783   Inf   PRELP
## ENSG00000130508   Inf    PXDN
## ENSG00000115758   Inf    ODC1
## ENSG00000152689   Inf RASGRP3
## ENSG00000115380   Inf  EFEMP1
## ENSG00000014641   Inf    MDH1
## ENSG00000115318   Inf   LOXL3
## ENSG00000159399   Inf     HK2
## ENSG00000115423   Inf   DNAH6
## ENSG00000153208   Inf   MERTK
## ENSG00000081479   Inf    LRP2
## ENSG00000114948   Inf  ADAM23
## ENSG00000138413   Inf    IDH1
```

6. Export the report for the differentially expressed genes from the analysis as html file.

Export html report for the differentially expressed genes from the analysis.

```
library("ReportingTools")
```

```
## No methods found in "RSQLite" for requests: dbGetQuery
```

```
##
```

```
## Warning: replacing previous import 'ggplot2::Position' by
## 'BiocGenerics::Position' when loading 'ggbio'
```

```
## No methods found in "RSQLite" for requests: dbGetQuery
```

```
##
```

```
htmlRep <- HTMLReport(shortName=paste0("December2017 DESeq2 ANOVA"), title=paste0("Differentially expres
publish(sdg, htmlRep)
url <- finish(htmlRep)
browseURL(url)
```

1. Write the data into a data frame as csv.

Write data into a data frame.

```
write.csv(sdg, file = paste0("./data/results/results_DESeq2_", "ANOVA",".csv"))
```

9. Heatmap of differentially expressed genes:

From the set of differentially expressed genes, we will show heatmap of the normalized expression across the samples.

```
dataF <- data.frame(assay(rld))
mat1 <- dataF[(rownames(dataF) %in% rownames(sdg)),]
mat1 <- data.matrix(mat1)
rownames(mat1) <- sdg$symbol
mat1 <- mat1 - rowMeans(mat1)
```

```
colnames(mat1) <- paste0(cellName)
require(gplots)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:IRanges':
##
##     space
```

```
## The following object is masked from 'package:S4Vectors':
##
##     space
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
my_palette <- colorRampPalette(c("green","red"))(n = 120)
sidecols = c("blue", "yellow", "black")[cellName]
heatmap.2(mat1, trace="none", col=my_palette, ColSideColors = sidecols, labRow = rownames(mat1),
          mar=c(10,8), scale="row", main="", key = T)
```