**Final Project**

**Name: Nuerye Ainiwan**

**Part-I**

Creating summarized Experiment object. The alignment step is already done for you and are located as /courses/example-data/bam/SRR\*\_all.bam. Fill in the following table given below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **command** | **what is the name of the function in this command?** | **list argument variables of the function and their type** | **what is the name and type of output variable (if data is returned)?** | **What does the function do? Brief** |
| sampleTable <- read.csv("/courses/example-data/csv/ sample\_table.reduced.csv",row.names=1) | read.csv | file [“characte r”] (),row.na mes [numeric] | sampleTable  [data.frame] | Loads csv file and creates a data frame from it |
| fileNames <- file.path("/courses/example-data/bam/ ", paste0(sampleTable$Run, "\_all.bam")) | file.path | 1.("/courses/example-ata/bam/ ") [‘character],  2. paste0(sampleTable$Run, "\_all.bam")[‘character] | fileNames  [character] | making a file path to the file we want to perform |
| library("Rsamtools") | NA | NA | NA | NA |
| bamFiles <- BamFileList(fileNames, yieldSize=2000000) | BamFileList | 1.fileNames,[ "character"],  2.yieldSize [  [1] "standardGeneric"  attr(,"package")  [1] "methods"] | bamFiles  [ [1]"BamFileList"  attr(,"package")  [1] "Rsamtools"] | Providing a way of managing BamFile |
| library("GenomicFeatures") | NA | NA | NA | NA |
| txdb <- makeTxDbFromGFF("/courses/example-data/gtf/Homo\_sapiens.GRCh37.75.gtf", format="gtf",circ\_seqs=character()) | makeTxDbFromGFF | 1.("/courses/example-data/gtf/Homo\_sapiens.GRCh37.75.gtf")[‘character],  2. format[function] | Txdb  [[1] "TxDb"  attr(,"package")  [1]"GenomicFeatures"] | build a txdb from GTF file |
| ebg <- exonsBy(txdb, by="gene") | exonsBy | 1. txdb [[1] "TxDb"  attr(,"package")  [1]"GenomicFeatures"]  2.by[ [1] "standardGeneric"  attr(,"package")  [1] "methods"] | Ebg  [[1]"GRangesList"  attr(,"package")  [1]"GenomicRanges"] | exons grouped in a GRangesList by gene |
| library("GenomicAlignments") | NA | NA | NA | NA |
| library("BiocParallel") | NA | NA | NA | NA |
| register(SerialParam()) | register | (SerialParam())  [1] "SerialParam"  attr(,"package")  [1] "BiocParallel" | NA | to add to or query a registry of back-endsq |
| se <- summarizeOverlaps(features=ebg, reads=bamFiles, mode="Union", singleEnd=FALSE, ignore.strand=TRUE, fragments=TRUE ) | summarizeOverlaps | 1.features  [1] "standardGeneric"  attr(,"package")  [1] "methods"  2. mode[function] | se  [1]"RangedSummarizedExperiment"  attr(,"package")  [1]"SummarizedExperiment" | Assign a value to a name.(Summarizing information across ranges and experiments) |
| colData(se) <- DataFrame(sampleTable) | DataFrame | sampleTable["data.frame"] | colData(se)  [1] "DataFrame"  attr(,"package")  [1] "S4Vectors" | Assign a value to a name. (assign the sampleTable as the colData of the summarized experiment, by converting it into a DataFrame and using the assignment function) |
| se$dex <- relevel(se$dex, "untrt") | relevel | 1. se$dex["factor"] 2. "untrt"[ "character"] | se$dex  ["factor] | Assign a value to a name. (The levels of a factor are re-ordered) |

# Part-II

Visualization and exploration. Fill in the following table given below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **command** | **what is the name of the function in this command?** | **list argumen t variables of the function and their type** | **what is the name and type of output variable (if data is returned)?** | **What does the function do? Brief** |
| library("DESeq2") | NA | NA | NA | NA |
| dds <- DESeqDataSet(se, design = ~ cell + dex) | DESeqDataSet | 1.se [[1]"RangedSummarizedExperiment"  attr(,"package")  [1]"SummarizedExperiment"]  2.design  [[1]"standardGeneric"  attr(,"package")  [1]"methods"] | dds  [[1] "DESeqDataSet"  attr(,"package")  [1] "DESeq2"] | used to store the input values, intermediate calculations and results of an analysis of differential expression. |
| nrow(dds) | nrow | dds  [[1] "DESeqDataSet"  attr(,"package")  [1]"DESeq2"] | nrow(dds)  ["integer"] | Return the number of rows in an array-like object. |
| dds <- dds[rowSums (counts(dds)) > 1, ] | dds | 1.rowSums  [[1] "standardGeneric"  attr(,"package")  [1] "methods"]  2. counts  [[1] "standardGeneric"  attr(,"package")  [1] "methods"] | Dds  [[1] "DESeqDataSet"  attr(,"package")  [1] "DESeq2"] | removing rows of the DESeqDataSet that have no counts |
| nrow(dds) | nrow | dds  [[1] "DESeqDataSet"  attr(,"package")  [1]"DESeq2"] | nrow(dds)  ["integer"] | Return the number of rows in an array-like object. |
| rld <- rlog(dds, blind=FALSE) | rlog | dds  [[1] "DESeqDataSet"  attr(,"package")  [1]"DESeq2"] | rld  [[1]"DESeqTransform"  attr(,"package")  [1] "DESeq2"] | transforms the count data to the log2 scale that minimizes differences between samples for rows with small counts, and normalizes with respect to library size. |
| head(assay(rld), 3) | head | 1.assay(rld)  [matrix]  2.3  ["numeric"] | head(assay(rld), 3)  [ "matrix"] | Returns the first 3 parts of a matrix.  . |
| sampleDists <- dist( t( assay(rld) ) ) | dist | t  [[1]"standardGeneric"attr(,"package")  [1] "methods" | sampleDists  ["dist"] | computes and returns the distance matrix |
| library("pheatmap") | NA | NA | NA | NA |
| library("RColorBrewer") | NA | NA | NA | NA |
| sampleDistMatrix <-as.matrix(sampleDists) | as.matrix | sampleDists  [dist] | sampleDistMatrix  [ "matrix"] | turn its argument into a matrix. |
| rownames(sampleDistMatrix) <- paste( rld$dex, rld$cell, sep="-" ) | paste | 1. rld$dex["factor"]  2. rld$cell["factor"] | rownames(sampleDistMatrix  ["character"] | concatenates vectors of strings or vector-like objects containing strings. |
| colnames(sampleDistMatrix) <- NULL | NA | NA | NA | NA |
| colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255) | colorRampPalette | rev["function"] | Colors  ["character"] | return functions that interpolate a set of given colors to create new color palettes and color ramps, functions that map the interval [0, 1] to colors. |
| pheatmap(sampleDistMatrix, clustering\_distance\_rows=sampleDist s, clustering\_distance\_cols=sampleDists  , col=colors) | pheatmap | 1.sampleDistMatrix  ["matrix"]  2.col["function"] | NA | A function to draw clustered heatmaps. |
| plotPCA(rld, intgroup = c("dex", "cell")) | plotPCA | rld  [[1] "DESeqTransform"  attr(,"package")  [1] "DESeq2"] | NA | This plot helps to check for batch effects and the like. |

# Part-III

Running the differential expression pipeline: building the results table. Fill in the following table given below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **command** | **what is the name of the function in this comma nd?** | **list argument variables of the function and their type** | **what is the name and type of output variable (if data is returned)?** | **What does the function do? Brief** |
| library("DESeq2") | NA | NA | NA | NA |
| dds <- DESeqDataSet(se, design = ~ cell + dex) | DESeqDataSet | 1.se  [[1]"RangedSummarizedExperiment"  attr(,"package")  [1]"SummarizedExperiment"]  2. design  [[1]"standardGeneric"  attr(,"package")  [1] "methods"] | dds  [[1]"DESeqDataSet"  attr(,"package")  [1] "DESeq2"] | used to store the input values, intermediate calculations and results of an analysis of differential expression. |
| dds <- DESeq(dds) | DESeq | dds  [[1]"DESeqDataSet"  attr(,"package")  [1] "DESeq2"] | dds  [[1] "DESeqDataSet"  attr(,"package")  [1] "DESeq2"] | 1.estimation of size factors  2.estimation of dispersion  3.Negative binomial GLM fitting and Wald statistics |
| res <- results(dds) | results | dds  [[1]"DESeqDataSet"  attr(,"package")  [1] "DESeq2"] | res  [[1]"DESeqResults"  attr(,"package")  [1] "DESeq2"] | extracts a result table from a DESeq analysis giving base means across samples, log2 fold changes, standard errors, test statistics, p-values and adjusted p-values and assign to res |
| mcols(res, use.names=TRUE) | mcols | 1.res  [[1] "DESeqResults"  attr(,"package")  [1] "DESeq2"] | [1] "DataFrame"  attr(,"package")  [1] "S4Vectors" | ‘mcols’ stands for \_metadata columns q and is represented as a DataTable object with a row for each element and a column for each metadata |
| summary(res) | summary | res  [[1] "DESeqResults"  attr(,"package")  [1] "DESeq2"] | the results from a DESeq analysis. | Print a summary of the results from a DESeq analysis. |
| res.05 <- results(dds, alpha=.05) | results | dds[[1] "DESeqDataSet"  attr(,"package")  [1] "DESeq2"] | res.05[[1] "DESeqResults"  attr(,"package")  [1] "DESeq2"] | extracts a result table from a DESeq analysis giving base means across samples, log2 fold changes, standard errors,test statistics, p-values and adjusted p-values and assign to res.05 |
| table(res.05$padj < .05) | table | res.05$padj  ["numeric"] | [table] | uses the cross-classifying factors to build a contingency table of the counts at each combination of factor levels. |
| resLFC1 <- results(dds, lfcThreshold=1) | results | dds  [[1]"DESeqDataSet"  attr(,"package")  [1] "DESeq2"] | resLFC1  [[1]"DESeqResults"attr(,"package")  [1] "DESeq2"] | extracts a result table from a DESeq analysis giving base means across samples, log2 fold changes, standard errors,test statistics, p-values and adjusted p-values. |
| table(resLFC1$padj < 0.1) | table | resLFC1$padj  ["numeric"] | [table] | uses the cross-classifying factors to build a contingency table of the counts at each combination of factor levels. |

**Part-IV**

Running the differential expression pipeline: other comparisons. Fill in the following table given below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **command** | **what is the name of the function in this comma nd?** | **list argument variables of the function and their type** | **what is the name and type of output variable (if data is returned)?** | **What does the function do? Brief** |
| results(dds, contrast=c("cell", "N061011", "N61311")) | results | 1.dds  [[1] "DESeqDataSet"  attr(,"package")  [1] "DESeq2"] | results table | results will extract the results table for a comparison of the last level over the first level. |

# 

# Part-V

Running the differential expression pipeline: multiple testing. Fill in the following table given below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **command** | **what is the name of the function in this comma nd?** | **list argument variables of the function and their type** | **what is the name and type of output variable (if data is returned)?** | **What does the function do? Brief** |
| sum(res$pvalue < 0.05, na.rm=TRUE) | sum | res$pvalue  ["numeric"] | 448["numeric"] | returns the sum of all the values present in its arguments. |
| sum(!is.na(res$pvalue)) | sum | is.na  ["function"] | 6381["numeric"] | returns the sum of all the values present in its arguments. |
| sum(res$padj < 0.1, na.rm=TRUE) | sum | res$padj  ["numeric"] | 267["numeric"] | returns the sum of all the values present in its arguments. |
| resSig <- subset(res, padj < 0.1) | subset | res  [ [1] "DESeqResults"  attr(,"package")  [1] "DESeq2"] | resSig,  results table | Return subsets of vector-like, matrix-like or data-frame-like  objects which meet conditions and assign to resSig |
| head(resSig[ order(resSig$log2FoldChange),]) | head | results table | table | Returns the first or last parts of a vector, matrix, table, data  frame or function. This will get the significant genes with the strongest down-regulation: |
| head(resSig[ order(resSig$log2FoldChange, decreasing=TRUE), ]) | head | results table | table | Returns the first or last parts of a vector, matrix, table, data  frame or function. This will get the significant genes with the strongest up-regulation: |

# Part-VI

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **what is** | **list** | **what is the** | **What does the** |
|  | **the** | **argument** | **name and** | **function do?** |
|  | **name of** | **variables** | **type of output** | **Brief** |
|  | **the** | **of the** | **variable (if** |  |
|  | **function** | **function** | **data is** |  |
|  | **in this** | **and their** | **returned)?** |  |
|  | **comma** | **type** |  |  |
| **command** | **nd?** |  |  |  |
| topGene <-rownames(res)[which.min(res$padj)] | rownames | 1.res  [[1] "DESeqResults"  attr(,"package")  [1] "DESeq2"]  2. which.min ["function"]  3. res$padj  [ "numeric"] | topGene ["character"] | Get the row names of a matrix-like object. |
| plotCounts(dds, gene=topGene, intgroup=c("dex")) | plotCounts | 1.dds:  [‘DESeqDataSet’]  2.gene: [character]  3.intgroup:[interesting groups] | visualize the counts for a particular gene | normalized counts **for a single gene over treatment group** |

Running the differential expression pipeline: plotting results. Fill in the following table given below.

# Part-VII

Running the differential expression pipeline: annotating and exporting results. Fill in the following table given below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **command** | **what is the name of the function in this comma nd?** | **list argument variables of the function and their type** | **what is the name and type of output variable (if data is returned)?** | **What does the function do? Brief** |
| library("AnnotationDbi") | NA | NA | NA | NA |
| library("org.Hs.eg.db") | NA | NA | NA | NA |
| columns(org.Hs.eg.db) | columns | org.Hs.eg.db:  [ [1] "OrgDb"  attr(,"package")  [1] "AnnotationDbi"] | ["character"] | get a list of all available key types |
| res$symbol <- mapIds(org.Hs.eg.db, keys=row.names(res),column="SYMBOL", keytype="ENSEMBL",multiVals="first") res$entrez <- mapIds(org.Hs.eg.db, keys=row.names(res), column="ENTREZID", keytype="ENSEMBL", multiVals="first") | mapIds | 1. org.Hs.eg.db:  [ [1] "OrgDb"  attr(,"package")  [1] "AnnotationDbi"]  2. keys["methods"] | res$symbol  ["character"]  res$entrez  ["character"] | gets the mapped ids (column) for a set of keys that are of a particular keytype |
| resOrdered <- res[order(res$padj),] | res | order(res$padj)  [integer] | resOrdered  [[1]"DESeqResults"attr(,"package")  [1] "DESeq2"] | to get the results of having the desired external gene IDs and assign to resOrdered |
| head(resOrdered) | head | resOrdered  [[1] "DESeqResults"  attr(,"package")  [1] "DESeq2"] | results table | Returns the first or last parts of a vector, matrix, table, data  frame or function |
| resOrderedDF <- as.data.frame(resOrdered)[1:100,] | as.data.frame | resOrdered  [[1] "DESeqResults"  attr(,"package")  [1] "DESeq2"] | resOrderedDF  ["data.frame"] | to coerce to a data frame. |
| write.csv(resOrderedDF, file="results.csv") | write.csv | 1.resOrderedDF: ["data.frame"]  2.file [function] | NA | prints its required argument to a file. |

# Part-VIII

What does adjusted p-value represent? Why do we need it - why don’t we just use p-value?

Adjusted p-value(padj) is BH-adjusted p values. It helps to control for the fact that sometimes small p-values (less than 5%) happen by chance, which could lead you to incorrectly reject the true null hypotheses. P-adi is used during multiple comparisons. In the case of differential gene expression experiments, we want to eliminate those genes that skew the results with small p-values(less than 5%)by chance. In order to avoid such scenarios we use p-adj values. In other words, the B-H Procedure helps you to avoid Type I errors (false positives).

# Part-IX

For the top 5 genes reported in your CSV output file "results.csv", look them up at the ENSEMBL website and see if their differential expression under dex treatment makes sense. For example, for ENSG00000162692 the URL you need to go to will be [http://useast.ensembl.org/Homo\_sapiens/Gene/Summary?db=core;g=ENSG00000162692](http://useast.ensembl.org/Homo_sapiens/Gene/Summary?db=core%3Bg%3DENSG00000162692)

Input command: nohup Rscript script.all.R > logfile.txt &

Output:

"ENSG00000162692",632.144951496753,-3.01251700449594,0.231277605389117,-13.0255456399572,8.75728246825134e-39,2.84611680218168e-35,"VCAM1","7412"

"ENSG00000163083",649.871309997304,3.12705953312107,0.255889471432044,12.220352465543,2.42033055011129e-34,3.93303714393084e-31,"INHBB","3625"

"ENSG00000143127",322.830580274002,3.34458510618928,0.292101927045739,11.4500617644524,2.34978027834241e-30,2.54559530153762e-27,"ITGA10","8515"

"ENSG00000134243",7040.3224671768,2.07107904743923,0.185868321715306,11.142722053581,7.77073205146793e-29,6.31371979181769e-26,"SORT1","6272"

"ENSG00000116285",3833.87289745161,2.15311201394626,0.196333780000261,10.9665897225806,5.5319704748743e-28,3.59578080866829e-25,"ERRFI1","54206"

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Gene stable ID | Gene name | Biological process | Expression | under dex treatment makes sense |
| ENSG00000162692 | VCAM1 | Inflammatory response | Expressed on inflamed tissue. | Yes, this makes sense, as the drug can interrupt inflammation by moving into cells and suppressing the proteins that go on to promote inflammation. |
| ENSG00000163083 | INHBB | inflammatory response | Expressed on smooth muscle cells of the airway | Yes, this makes sense, as the smooth muscle cells of the airway are known to react to glucocorticoid steroids. [PMID30694689] |
| ENSG00000143127 | ITGA10 | smooth muscle cell-matrix adhesion | Expresses on lung | Yes, this makes sense, as the smooth muscle cells of the airway are known to react to glucocorticoid steroids. [PMID30694689] |
| ENSG00000134243 | SORT1 | regulation of gene expression | Expresses on smooth muscle tissue | Yes, this makes sense, as the smooth muscle cells of the airway are known to react to glucocorticoid steroids. [PMID30694689] |
| ENSG00000116285 | ERRFI1 | 1.kinase binding  2.regulation of interleukin-1 beta production  3. lung epithelium development | Expresses on lung, bronchial epithelial cell | Yes, this makes sense as the smooth muscle cells of the airway are known to react to glucocorticoid steroids.[PMID30694689]Important for normal prenatal and perinatal lung development. |