Functional Genomics: Assignment 3

University of Cambridge

Henrik Åhl

January 18, 2017

Code for getting exon counts

```bash
#!/bin/bash
###################################################################
# FILE DESCRIPTION:
# 1) Run DEXSeq count on preprocessed files.
###################################################################
HOMEDIR="/local/data/public/hpa22/assignments/fga3"
SCRIPTSDIR=$HOMEDIR"/scripts"
mkdir -p $HOMEDIR"/counts_exons"
COUNTSDIR=$HOMEDIR"/counts_exons/"
TOPHAT_OUTPUT=$HOMEDIR"/alignments/rerun_default_tophat_merge/tophat_"
GFF_FILE=$HOMEDIR"/misc/Homo_sapiens.GRCh37.64.DEXSeq.chr.gff"
DEXSeq="python /local/data/public/hpa22/R/lib/DEXSeq/python_scripts/dexseq_count.py -s no"

for ii in $(seq 1 47); do
  FILENO=$(($ii * 4 + 16))
  (
      nice -n 5 $DEXSeq $GFF_FILE $TOPHAT_OUTPUT$FILENO"_dedup.sam" \
         $COUNTSDIR$FILENO"_counts.dat"

      $SCRIPTSDIR/gp $COUNTSDIR # Give permission
  ) &
done
```

Code for identifying differentially expressed genes

```r
#!/usr/bin/Rscript
# source("https://bioconductor.org/biocLite.R")
.libPaths(c("/local/data/public/hpa22/R/lib/", .libPaths()))
library("DESeq"); library("GenomicFeatures"); library("GenomicAlignments")
library("gridGraphics"); library("grid"); library("VennDiagram"); library("BiocParallel")
library("RColorBrewer"); library("gplots")
library("ReactomePA")
library("EnsDb.Hsapiens.v75")
HOMEDIR = "/local/data/public/hpa22/assignments/fga3/"
setwd(HOMEDIR)

#########################
# READ IN AND TREAT DATA
#########################
# Read in files and conversion table for their IDs
files      = system("ls counts/*counts*.dat", intern = TRUE) # Our files
# files = system("find data/processed_data/rnaseq/ -type f", intern=TRUE) # Rory's files
ids.table  = read.table("names/rnaseq.dat", header = TRUE)
sign.level = 0.05

```

```r
# Get list of genes and assign space to resulting SummarizedExperiment object
gene.list  = read.table(files[1])[,1]
counts.tbl = matrix(nrow=length(gene.list), ncol=length(files))
rownames(counts.tbl)=gene.list

# Retrieve metadata (our)
tmp       = vector()
files.ids = as.numeric(unname(sapply(sapply(files, function(x) strsplit(x, "/")[[1]][2]),
    function(y) strsplit(y, "_")[[1]][1])))
for(ii in files.ids) {tmp = append(tmp, which(ids.table[,"SRA_short"] == ii))} # Get order of
     files (our)
metadata  = ids.table[tmp, ]
metadata[,"Condition"] = factor(metadata[, "Condition"]) # This should be a factor

# Assemble counts table from htseq-count files, including all genes
for (ii in 1:length(files)) {next.data = read.table(files[ii]); counts.tbl[,ii] = next.data
    [,2]}
counts.tbl = counts.tbl[1:(nrow(counts.tbl) - 5), ] # Remove some trash

# chaperones / co-chaps to PR + FOXA1 + PRG1 + ERG1, and cofactors to E\alpha
complex.genes = c(
  "ENSG00000120738", # ERG1
  "ENSG00000082175", # PRG1
  "ENSG00000080824", # HSP90
  "ENSG00000004478", # FKB4
  "ENSG00000096060", # FKB5
  "ENSG00000129514", # FOXA1
  "ENSG00000180530", # NRIP
  "ENSG00000107485", # GATA3
  "ENSG00000140332"  # TLE3
)

add.info = function(resSig) {
  rownames(resSig) = resSig[,1]
  resSig$symbol = mapIds(EnsDb.Hsapiens.v75,
                    keys=row.names(resSig),
                    column="SYMBOL",
                    keytype="GENEID",
                    multiVals="first")
  resSig$entrez = mapIds(EnsDb.Hsapiens.v75,
                    keys=row.names(resSig),
                    column="ENTREZID",
                    keytype="GENEID",
                    multiVals="first")
  resSig$name = mapIds(EnsDb.Hsapiens.v75,
                    keys=row.names(resSig),
                    column="GENENAME",
                    keytype="GENEID",
                    multiVals="first")
  return(resSig)
}

cont.ind = which((metadata[,"Condition"] == "E2"))
c.cds    = newCountDataSet(counts.tbl[, cont.ind], metadata[cont.ind, "Cell_type"])
c.cds    = estimateSizeFactors(c.cds)
c.cds    = estimateDispersions(c.cds)
c.res    = nbinomTest(c.cds, "MCF7", "T47D")

prog.ind = which(metadata[,"Condition"] == "E2+Progesterone")
p.cds    = newCountDataSet(counts.tbl[, prog.ind ], metadata[prog.ind, "Cell_type"])
p.cds    = estimateSizeFactors(p.cds)
p.cds    = estimateDispersions(p.cds)
p.res    = nbinomTest(p.cds, "MCF7", "T47D")

# Take out significant part
load(".our_DESeq.RData") # Has T.prog.resSig & M.prog.resSig (Cell-line treatment comparisons
    )
```

```r
p.resSig = p.res[which(p.res$padj < sign.level), ]
c.resSig = c.res[which(c.res$padj < sign.level), ]

# Add necessary columns
T.prog.resSig = add.info(T.prog$resSig)
M.prog.resSig = add.info(M.prog$resSig)
p.resSig      = add.info(p.resSig)
c.resSig      = add.info(c.resSig)

# Treatments / cell lines
T.prog.resSig = T.prog.resSig[which(abs(T.prog.resSig$log2FoldChange) > 0),]
M.prog.resSig = M.prog.resSig[which(abs(M.prog.resSig$log2FoldChange) > 0),]
p.resSig      = p.resSig[which(abs(p.resSig$log2FoldChange) > 0),]
c.resSig      = c.resSig[which(abs(c.resSig$log2FoldChange) > 0),]

# Observation: Difference is much bigger between cell lines than between the treatments.
nrow(T.prog.resSig)
nrow(M.prog.resSig)
nrow(p.resSig)
nrow(c.resSig)

M.v.p    = intersect(rownames(M.prog.resSig), rownames(p.resSig))
M.v.c    = intersect(rownames(M.prog.resSig), rownames(c.resSig))

p.pos    = p.resSig[which(p.resSig$log2FoldChange > 0), ]
c.pos    = c.resSig[which(c.resSig$log2FoldChange > 0), ]
p.neg    = p.resSig[which(p.resSig$log2FoldChange < 0), ]
c.neg    = c.resSig[which(c.resSig$log2FoldChange < 0), ]

# Genes regulated by progesterone in both cell lines
prog.reg = intersect(T.prog.resSig[,1], M.prog.resSig[,1])

# Genes differentially regulated by progesterone in both T & M
grid.newpage()
a=draw.pairwise.venn(nrow(T.prog.resSig), nrow(M.prog.resSig), length(prog.reg) , fill=c("
    aquamarine", "coral"),cex=2, category=c("T47D","MCF7"), cat.pos=c(-120,120),cat.cex = 2,
    cat.dist = .1,mar=c(.6,.6,.6,.6),ext.dist=.1)
png("figures/presentation_figures/henrik_mvt.png")
grid.draw(a)
dev.off()

################################################
################################################
################################################

cl.cons      = union(rownames(p.pos[which(rownames(p.pos) %in% rownames(c.pos)), ]),
                     rownames(p.neg[which(rownames(p.neg) %in% rownames(c.neg)), ]))
cl.noncons   = union(rownames(p.pos[which(rownames(p.pos) %in% rownames(c.neg)), ]),
                     rownames(p.neg[which(rownames(p.neg) %in% rownames(c.pos)), ]))
grid.newpage()
a = draw.pairwise.venn(nrow(p.resSig), nrow(c.resSig), length(intersect(c.resSig[,1],p.resSig
    [,1])) , fill=c("aquamarine", "coral"),cex=2, category=c("Progesterone","Control"),
    cat.pos=c(-30,30),cat.cex=2,cat.dist = .1,mar=c(.4,.4,.4,.4),ext.dist=.08,ext.percent =
    .5)
png("figures/presentation_figures/henrik_pvc.png")
grid.draw(a)
dev.off()

noncons = cl.noncons
cons    = cl.cons
M.col   = intersect(noncons, M.prog.resSig[,1])
T.col   = intersect(noncons, T.prog.resSig[,1])
length(M.col)
length(T.col)

################################################
################################################
```

```r
##############################################

# Which genes are upregulated / downregulated in T47D?
upreg    = rownames(p.pos[which(rownames(p.pos) %in% rownames(c.pos)), ])
downreg  = rownames(p.neg[which(rownames(p.neg) %in% rownames(c.neg)), ])
upreg.de = mapIds(EnsDb.Hsapiens.v75,keys=upreg, column="ENTREZID", keytype="GENEID", multiVals="first")
upreg.pw = enrichPathway(gene=upreg.de, pvalueCutoff=sign.level, readable=T)
png("figures/presentation_figures/henrik_upreg_pw.png", width=1000, height=400)
barplot(upreg.pw, showCategory = 10)
dev.off()

downreg.de = mapIds(EnsDb.Hsapiens.v75,keys=downreg, column="ENTREZID", keytype="GENEID", multiVals="first")
downreg.pw = enrichPathway(gene=downreg.de, pvalueCutoff=sign.level, readable=T)
png("figures/presentation_figures/henrik_downreg_pw.png", width=1000, height=400)
barplot(downreg.pw, showCategory = 10)
dev.off()

##############################################
##############################################
##############################################

nonconsandprogreg = intersect(noncons, prog.reg)
consandprogreg    = intersect(cons,    prog.reg)
# write(mapIds(EnsDb.Hsapiens.v75,keys=cl.noncons, column="SYMBOL", keytype="GENEID", multiVals="first"), sep="\n", file="indiv/enrique/nc.dat")
# write(mapIds(EnsDb.Hsapiens.v75,keys=cl.cons, column="SYMBOL", keytype="GENEID", multiVals="first"), sep="\n", file="indiv/enrique/c.dat")
# write(mapIds(EnsDb.Hsapiens.v75,keys=nonconsandprogreg, column="SYMBOL", keytype="GENEID", multiVals="first"), sep="\n", file="indiv/enrique/nonconsandprogreg.dat")
# write(nonconsandprogreg, sep="\n", file="indiv/enrique/nonconsandprogreg.dat")
# write(consandprogreg, sep="\n",    file="indiv/enrique/consandprogreg.dat")
# write(mapIds(EnsDb.Hsapiens.v75,keys=consandprogreg, column="SYMBOL", keytype="GENEID", multiVals="first"), sep="\n", file="indiv/enrique/consandprogreg.dat")

cons.de = mapIds(EnsDb.Hsapiens.v75,keys=cl.cons, column="ENTREZID", keytype="GENEID", multiVals="first")
cons.pw = enrichPathway(gene=cons.de, pvalueCutoff=sign.level, readable=T)
png("figures/presentation_figures/henrik_cons_pw.png", width=600, height=400)
barplot(cons.pw, showCategory = 10)
dev.off()
# enrichMap(cons.pw, layout=igraph::layout.kamada.kawai, vertex.label.cex=1)

noncons.de = mapIds(EnsDb.Hsapiens.v75,keys=cl.noncons, column="ENTREZID", keytype="GENEID", multiVals="first")
noncons.pw = enrichPathway(gene=noncons.de, pvalueCutoff=sign.level, readable=T)
png("figures/presentation_figures/henrik_noncons_pw.png", width=600, height=400)
barplot(noncons.pw, showCategory = 10)
dev.off()
# enrichMap(noncons.pw, layout=igraph::layout.kamada.kawai, vertex.label.cex=1)

# Progesterone regulated genes
prog.reg.cl   = intersect(M.v.T, p.v.c)
T.prog.reg.cl = intersect(rownames(T.prog.resSig), p.v.c)
M.prog.reg.cl = intersect(rownames(M.prog.resSig), p.v.c)

##############################################
##############################################
##############################################

M.v.T = union(intersect(rownames(T.prog.resSig[which(T.prog.resSig$log2FoldChange < 0),]), rownames(M.prog.resSig[which(T.prog.resSig$log2FoldChange < 0),])),
              intersect(rownames(T.prog.resSig[which(T.prog.resSig$log2FoldChange > 0),]), rownames(M.prog.resSig[which(T.prog.resSig$log2FoldChange > 0),])))
p.v.c = union(intersect(rownames(p.resSig[which(p.resSig$log2FoldChange < 0),]), rownames(c.resSig[which(c.resSig$log2FoldChange < 0),])),
               intersect(rownames(p.resSig[which(p.resSig$log2FoldChange > 0),]),
                   rownames(c.resSig[which(c.resSig$log2FoldChange > 0),])))

length(M.v.T)
length(p.v.c)

# Genes driven in either M or T
M.v.T.de = mapIds(EnsDb.Hsapiens.v75,keys=M.v.T, column="ENTREZID", keytype="GENEID",
    multiVals="first")
M.v.T.pw = enrichPathway(gene=M.v.T.de, pvalueCutoff=sign.level, readable=T)
barplot(M.v.T.pw,   showCategory = 10)
enrichMap(M.v.T.pw, layout=igraph::layout.kamada.kawai, vertex.label.cex=1)
cnetplot(M.v.T.pw,  categorySize="pvalue", foldChange=M.v.T.de)

# Progesterone regulated cell line-differential genes
prog.reg.cl.de = mapIds(EnsDb.Hsapiens.v75,keys=prog.reg.cl, column="ENTREZID", keytype="
    GENEID", multiVals="first")
prog.reg.cl.de = prog.reg.cl.de[-which(is.na(prog.reg.cl.de))]
prog.reg.cl.pw = enrichPathway(gene=prog.reg.cl.de[-11], readable=T, pvalueCutoff=0.05)
barplot(prog.reg.cl.pw,   showCategory = 30)
enrichMap(prog.reg.cl.pw, layout=igraph::layout.kamada.kawai, vertex.label.cex=1)
cnetplot(prog.reg.cl.pw,  categorySize="pvalue", foldChange=prog.reg.de)

# sign = intersect(M.prog$res[which(M.prog$res$padj < 0.01 & abs(M.prog$res$log2FoldChange) >
     1.2), ][,1], T.prog$res[which(T.prog$res$padj < 0.01 & abs(M.prog$res$log2FoldChange) >
    1.2), ][,1])
# write(sign, sep="\n",file="indiv/enrique/sign.dat")
# p.resSig: significantly expressed genes between cell lines treated with progesterone
# c.resSig: significantly expressed genes between cell lines treated with progesterone
# T.prog.resSig: sig genes between prog and control in T
# M.prog.resSig: sig genes between prog and control in M
# diff     = setdiff(rownames(p.resSig), rownames(c.resSig))
# diff.v.M = intersect(a, rownames(M.prog.resSig))
```

Code for DEXSeq

```
#!/usr/bin/Rscript
########################################
# FILE DESCRIPTION:
# Run DEXSeq on Progesterone / Control
# samples.
########################################

#source("https://bioconductor.org/biocLite.R")
.libPaths(c("/local/data/public/hpa22/R/lib/", .libPaths()))
library("DESeq"); library("GenomicFeatures"); library("GenomicAlignments")
library("gridGraphics"); library("grid"); library("VennDiagram"); library("BiocParallel")
library("RColorBrewer"); library("gplots")
HOMEDIR = "/local/data/public/hpa22/assignments/fga3/"
setwd(HOMEDIR)

########################
# READ IN AND TREAT DATA
########################
files      = list.files("counts_exons", full.names=TRUE)
ids.table  = read.table("names/rnaseq.dat", header = TRUE)
sign.level = 0.05

# Get list of genes and assign space to resulting SummarizedExperiment object
gene.list            = read.table(files[1])[,1]
counts.tbl           = matrix(nrow=length(gene.list), ncol=length(files))
rownames(counts.tbl) = gene.list

# Retrieve metadata (our)
```

```r
files.ids = as.numeric(unname(sapply(sapply(files , function(x) strsplit(x, "/")[[1]][2]) ,
    function(y) strsplit(y, "_")[[1]][1])))
tmp        = vector()
for(ii in files.ids)
{tmp = append(tmp, which(ids.table[,"SRA_short"] == ii))} # Get order of files (our)
metadata  = ids.table[tmp, ]
metadata[,"Condition"] = factor(metadata[, "Condition"])

# Read in the data appropriately
suppressPackageStartupMessages(library("DEXSeq"))
flattenedFile = list.files("misc", pattern="gff$",       full.names=TRUE)
countFiles    = list.files("counts_exons", pattern="", full.names=TRUE)
sampleTable   = data.frame(row.names=metadata[,"SRA_short"], cell_type=metadata[,"Cell_type"
    ], condition=metadata[,"Condition"])
idx           = which(sampleTable[,"condition"] == "E2" | sampleTable[,"condition"] == "E2+
    Progesterone")
sampleTable   = sampleTable[idx, ]
countFiles    = countFiles[idx]

# These guys are factors
sampleTable[,"condition"] = factor(sampleTable[,"condition"])
sampleTable[,"cell_type"] = factor(sampleTable[,"cell_type"])

## Create DEXSeq object from our info
dxd = DEXSeqDataSetFromHTSeq(
  countFiles,
  sampleData     = sampleTable,
  design         = ~ sample + exon + condition:exon,
  flattenedfile = flattenedFile)

# Different gene sets
sign              = readLines("indiv/enrique/sign.dat")
cons              = readLines("indiv/enrique/cons_ensid.dat")     # Too long :(
noncons           = readLines("indiv/enrique/noncons_ensid.dat") # Too long :(
consandprogreg    = readLines("indiv/enrique/consandprogreg.dat")
nonconsandprogreg = readLines("indiv/enrique/nonconsandprogreg.dat")
complex.genes     = c("ENSG00000120738", "ENSG00000082175", "ENSG00000080824", "
    ENSG00000004478", "ENSG00000096060", "ENSG00000129514", "ENSG00000180530",  "
    ENSG00000107485", "ENSG00000140332")

# Do the DEXSeq analysis
diffexp.subset = function(subset){
  # Subset stuff
  subset     = consandprogreg
  subset.dxd = dxd[geneIDs(dxd) %in% subset ,]

  # Normalise
  subset.dxd = estimateSizeFactors(subset.dxd)
  subset.dxd = estimateDispersions(subset.dxd)

  # Plot dispersion estimates
  # plotDispEsts(subset.dxd)

  # Test for differential expression
  subset.dxd  = testForDEU(subset.dxd)
  subset.dxd  = estimateExonFoldChanges(subset.dxd , fitExpToVar="cell_type")
  subset.dxr1 = DEXSeqResults(subset.dxd)

  # Significant?
  table(subset.dxr1$padj < sign.level)
  subset.dxr1[which(subset.dxr1$padj < sign.level), ]
}

# Get the differentially expressed exons and stuff
cons.de     = diffexp.subset(consandprogreg)
noncons.de = diffexp.subset(consandprogreg)
complex.de = diffexp.subset(consandprogreg)
```

```

# Plot whatever
# save(subset.dxr1, file=".complex.dxr")
# cat(unname(unlist(mapIds(EnsDb.Hsapiens.v75,keys= subset.dxr1[which(subset.dxr1$padj <
    sign.level), ][,1], column="SYMBOL", keytype="GENEID", multiVals="first"))), sep="\n")
# png("figures/presentation_figures/henrik_conserved_splicing_1.png", width = 900, height =
    600)
# plotDEXSeq(subset.dxr1, "ENSG00000062716", cex.axis=1.2, cex=1.3, lwd=2, FDR=0.01)
# dev.off()
# png("figures/presentation_figures/henrik_conserved_splicing_2.png", width = 900, height =
    600)
# plotDEXSeq(subset.dxr1, "ENSG00000160862", cex.axis=1.2, cex=1.3, lwd=2, FDR=0.01)
# dev.off()
#
```

Code for integrating ChIP and RNA-seq

```
#!/usr/bin/Rscript
#source("https://bioconductor.org/biocLite.R")
.libPaths(c("/local/data/public/hpa22/R/lib/", .libPaths()))
HOMEDIR    = "/local/data/public/hpa22/assignments/fga3/"
SCRIPTSDIR = paste0(HOMEDIR, "scripts/")
FIGDIR     = paste0(HOMEDIR, "figures/f5c/")
setwd(FIGDIR)

# Genes involved in the PR-ER binding machinery
complex.genes = c(
  "ENSG00000120738", # ERG1
  "ENSG00000082175", # PRG1
  "ENSG00000080824", # HSP90
  "ENSG00000004478", # FKB4
  "ENSG00000096060", # FKB5
  "ENSG00000129514", # FOXA1
  "ENSG00000180530", # NRIP
  "ENSG00000107485", # GATA3
  "ENSG00000140332"  # TLE3
)
gene.list = complex.genes
gene.list = unlist(gene.list)

# Get Ensembl data on TSS start and end sites
mart  = useMart(biomart="ENSEMBL_MART_ENSEMBL", host="grch37.ensembl.org", path="/biomart/
    martservice" ,dataset="hsapiens_gene_ensembl")
annot = getBM(attributes=c("ensembl_gene_id", "start_position", "end_position", "
    chromosome_name", "strand", "transcript_start", "transcript_end"),
              filters="ensembl_gene_id", values=gene.list, mart=mart)
colnames(annot) = c("ID", "Start", "End", "Chr", "Strand")

# Get 10kb region within TSS
neg.strand.idxs = which(annot[,"Strand"] == -1)
annot[neg.strand.idxs,"Start"]  = annot[neg.strand.idxs,"End"]    - 10e3
annot[neg.strand.idxs,"End"]    = annot[neg.strand.idxs,"End"]    + 10e3
annot[-neg.strand.idxs,"End"]   = annot[-neg.strand.idxs,"Start"] + 10e3
annot[-neg.strand.idxs,"Start"] = annot[-neg.strand.idxs,"Start"] - 10e3

# Rearrange a little for BED format and write to file
result = annot[, c(4,2,3,1)] # Change order
result[, "Chr"] = sapply(result[,"Chr"], function(x) paste0("chr", x)) # Append chr
    identifier
write.table(result, paste0(FIGDIR, "henrik_present.bed"), sep ="\t", quote=F, row.names=F,
    col.names=F)

# Run intersect and get genes the genes that overlap with binding sites (-10k)
system("rm -f henrik_intersect.bed")
system(paste0("/local/data/genome_informatics/programs/bedtools2/bin/bedtools",
              " intersect -a henrik_present.bed -b ", "erDiffPeaks_merged.bed", " > henrik_intersect.bed"))

# Port our genes through bioMart
genes     = unique(read.table("henrik_intersect.bed")[,4])
ensembl   = useMart(biomart="ENSEMBL_MART_ENSEMBL", host="grch37.ensembl.org", path="/biomart/martservice" ,dataset="hsapiens_gene_ensembl")
gene.symb = getBM(attributes = c('ensembl_gene_id','hgnc_symbol', "external_gene_name"), filters = 'ensembl_gene_id', values = genes, mart = ensembl)[,3]
print(gene.symb)
```