Jing Lu (2009lujing@gmail.com)
7/8/2016

RNAseq Normalization and Standard pair comparison with DESeq2

Programs required:

Installation:

Part0: Pre-process: (optional)

```
# read in raw counts table in which rows are raw counts of  each gene in each sample and column
is sample names How to get raw counts? HTSeq?

rawCnt <- read.table("rawCnt.txt",  header=T, quote=FALSE, sep="\t") ;

#make a experiment design table
condition <- factor( c( cond1, cond2, …..condK) )  # K= 1, 2,…
exp_design <- data.frame( row.names(exp_des=colnames(rawCnt),  sample=colnames(rawCnt),
condition=condition ) ;

#keep genes that have average raw counts 10  in at  least one condition

rawCnt_mean_cutoff = 10 ;
Grps <- levels(exp_design$condition) ;
meanCntPerGrp <- sapply( Grps, function(grp) rowMeans( rawCnt[ , exp_design[
exp_design$condition==grp, ]$sample , drop=FALSE ])) ;

clean_flag <- vector(mode="logical", length=nrow(meanCntPerGrp) ) ;
for ( each_col in 1:ncol(meanCntPerGrp) ){
    clean_flag = clean_flag | ( meanCntPerGrp[ , each_col] > rawCnt_mean_cutoff) ;
}

CntTbl <- rawCnt[ clean_flag , ] ;
Save(CntTbl, exp_design , file="input.Rdata" )
```

Part1: Normalize all samples

```
library(DESeq2) ;

dds <- DEseqDataSetFromMatrix( CntTbl , colData=exp_design, design = ~ condition );
dds <- estimateSizeFactor(dds) ;

#get DESeq2 normalized counts
deseq2_normalizedCnt <- counts( dds, normalized=TRUE) ;
write.csv( deseq2_normalizedCnt , file="deseq2_normalizedCnt.csv") ;
save(dds, file="dds.Rdata")
```

Jing Lu (2009lujing@gmail.com)
7/8/2016

Part2: standard pair-wise comparison

```
dds_obj <-“dds.Rdata” ;
load(dds_obj) ;

sub_dds <- function( dds, conds=c("cond1", "cond2") ){

        subdds <- dds[ , dds$condition %in% conds ] ;

        # relevel condition
        subdds$condition <- factor( as.character(subdds$condition) , levels=conds ) ;
         design(subdds) = ~ condition ;
        res <- results(subdds) ;

        # options used below works better for smaller sample size ,like each condition has equal or
less than 3 biological replicates .
        #res <- results(subdds, independentFiltering = FALSE,cooksCutoff = FALSE) ;

        return( as.data.frame(res)

}


# cond2  VS. cond1
res.cond2_vs_cond1 <- sub_dds(dds) ;
# cond3 VS. cond1
res.cond3_vs_cond1 <- sub_dds( dds, conds=c(“cond3”, “cond1”) )
```

Part3. optional output format of pairwise compare

#output a table more like DESeq which includes the average normalized counts of both of the conditions comparing .

```
sub_dds <- function(dds, conds =c( "cond1", "cond2")  ){
    subdds <- dds[ , dds$condition %in% conds ] ;
    subdds$condition <- factor( as.character(subdds$condition) , levels=conds ) ;
    design(subdds) = ~condition ;  # adjust donor effect
    subdds <- DESeq(subdds , betaPrior = FALSE) ;



    res <- results(subdds, independentFiltering = FALSE,cooksCutoff = FALSE) ;
    tbl <- data.frame( baseMeanA=baseMeanPerLvl[ ,conds[1] ] ,
                baseMeanB=baseMeanPerLvl[ , conds[2]] ,
                log2FoldChange=res$log2FoldChange ,
                pvalue=res$pvalue,
                padj=res$padj ) ;
```

Jing Lu (2009lujing@gmail.com)
7/8/2016

```
    colnames(tbl)[1] = paste("baseMean" , conds[1] , sep="_") ;
    colnames(tbl)[2] = paste("baseMean", conds[2], sep="_") ;
    return(tbl) ;
}


tbl.cond2_vs_cond1 <- sub_dds(dds) ;
tbl.cond3_vs_cond1 <- sub_dds(dds, conds=c( "A", "B") ) ;
```

Sample Files

Sample Visualization—
Quality Control
dispersion graph
Exon intron stats

Setup command

Reference links/publications for tools: