RPKM: reads per kilobase of transcript per million mapped reads

FPKM: fragments per kilobase of transcript per million mapped reads

In RNA-seq, the relative expression of a transcript is proportional to the number of cDNA fragments that originate from it. Paired-end RNA-seq experiments produce two reads per fragment, but that doesn’t necessarily mean that both reads will be mappable. For example, the second read is poor quality. If we were to count reads rather than fragments, we might double-count some fragments but not others, leading to a skewed expression value.

The FPKM is a normalized measure of expression level (having divided out the transcript length and the number of mapped reads).

TPM (transcripts per million), is a linear scaling of the FPKM, such that we could expect a gene with 1 TPM to have one molecule in a population of one million mRNA.

A number of new methods have emerged which allow for rapid quantification of transcript abundances, skipping the alignment step. Sailfish, kallisto, Salmon … these tools can be used to quickly generate gene count matrices, which can then be used by the gene-level differential expression packages such as DESeq2, edgeR, or limma-voom.

The Bioconductor software for importing the abundances from these tools, converting into gene-level counts is called “tximport”.

# for the normalization and transformation assessment, we will examine RNA-seq count matrices prepared by the ReCount project.

# download the files

# download.file("http://bowtie-bio.sourceforge.net/recount/ExpressionSets/wang\_eset.RData", "wang\_eset.RData")

# load this RData file within R

load("wang\_eset.RData")

library(Biobase)

# take a subset of the count matrix and the column data for building a DESeqDataSet

count.matrix <- exprs(wang.eset)[, 10:21]

col.data <- pData(wang.eset)[10:21,]

library(DESeq2)

dds <- DESeqDataSetFromMatrix(count.matrix, col.data, design = ~cell.type)

## our goal in the rest of the assessment will be to estimate a correction for sequencing depth and perform simple EDA of the samples by cell type

# Norm. and transform. Q1

# what tissue has the highest size factor?

dds <- estimateSizeFactors(dds)

sizeFactors(dds)

# inspect col.data to find out which cell type SRX003934 corresponding to

## Norm. and transform Q2

vsd <- varianceStabilizingTransformation(dds, blind=FALSE)

plotPCA(vsd, intgroup="cell.type")

## for the last question, we will make a scatterplot matrix of some of the samples' transformed counts

rmeans <- rowMeans(assay(vsd)) # row mean of rlog-transformed data

idx <- c(1,2,10,7,8,9,12) # pick some samples for visualization

mat <- assay(vsd)[rmeans > 1, idx] # pull out a small matrix of rlog-transformed counts

colnames(mat) <- vsd$cell.type[idx] # name the columns of matrix by cell type

panel.sub <- function(x,y,...) points(cbind(x,y)[sample(length(x),1000),],...)

panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...) {

usr <- par("usr"); on.exit(par(usr))

par(usr = c(0, 1, 0, 1))

r <- abs(cor(x, y))

txt <- format(c(r, 0.123456789), digits = digits)[1]

txt <- paste0(prefix, txt)

if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)

text(0.5, 0.5, txt, cex = cex.cor \* r)

}

pairs(mat, asp=1, col=rgb(0,0,0,.3), lower.panel=panel.cor, upper.panel=panel.sub)

# Read RSEM output files into R

genes <- read.table("SRR1039508.genes.results", header=T)

isoforms <- read.table("SRR1039508.isoforms.results", header=T)

## using split() and sapply(), confirm that the FPKM column in genes is the sum of the FPKM column in isoforms

fpkm.per.gene <- split(isoforms$FPKM, isoforms$gene\_id)

head(sapply(fpkm.per.gene, sum))

head(genes$FPKM)

## or equivalently using dplyr

library(dplyr)

isoforms %>% group\_by(gene\_id) %>% summarize(sum=sum(FPKM))

## make a histogram of the FPKM column in genes

## make a histogram after transforming by: log10(x+1)

# make new tables, removing the genes with FPKM 0

genes2 <- genes[genes$FPKM > 0, ]

genes2$gene\_id <- droplevels(genes2$gene\_id)

isoforms2 <- isoforms[isoforms$gene\_id %in% genes2$gene\_id, ]

isoforms2$gene\_id <- droplevels(isoforms2$gene\_id)

# perform a check that the gene\_id column in genes2 is equal to the levels of the gene\_id in isoforms2

stopifnot(all(genes2$gene\_id == levels(isoforms2$gene\_id)))

# if runs without error, the check passed

# with genes2, make a plot of the effective\_length (x) and the expected\_count (y), with both axes on the log scale

x <- log10(genes2$effective\_length)

y <- log10(genes2$expected\_count)

plot(x, y)

## transcript align Q1

## make a histogram of the FPKM in genes2.

## make a histogram after transforming with log10(x)

x <- genes2$FPKM

par(mfrow=c(1,2))

hist(x)

hist(log10(x))

median(x)

## transcript align Q2

IsoPct.each.gene <- split(isoforms2$IsoPct, isoforms2$gene\_id)

max.iso <- sapply(IsoPct.each.gene, max)

length(max.iso[max.iso>95])/length(max.iso)

## transcript align Q3

## make a plot of 'max.iso' on the x-axis, and genes2$FPKM on the y-axis (with log="y")

par(mfrow=c(1,1))

plot(max.iso, genes2$FPKM, log="y")

boxplot(split(log10(genes2$FPKM), cut(max.iso,5)), xlab="max.iso", ylab="log10 FPKM")

## calculate the number of isoforms per gene, and plot the maximum IsoPct over the number of isoforms

num.iso <- as.numeric(table(isoforms2$gene\_id))

plot(num.iso, max.iso)

barplot(table(num.iso))

barplot(table(num.iso[num.iso < 15]))

## load a single BAM files for a paired-end sequencing experiment

library(pasillaBamSubset)

bam.file <- untreated3\_chr4()

library(Rsamtools)

bf <- BamFile(bam.file)

# obtain the exons-by-gene object 'ebg' using this TxDb

library(TxDb.Dmelanogaster.UCSC.dm3.ensGene)

txdb <- TxDb.Dmelanogaster.UCSC.dm3.ensGene

ebg <- exonsBy(txdb, by="gene")

# subset to only the genes on chromosome 4

chr4.idx <- all(seqnames(ebg) == "chr4")

ebg.sub <- ebg[chr4.idx]

library(GenomicAlignments)

se <- summarizeOverlaps(ebg.sub, bf,

mode="Union",

singleEnd=FALSE,

ignore.strand=TRUE,

fragments=FALSE)

assay(se)[1]

# in this assessment, we will perform a gene-level differential expression analysis of brain

# tissues vs all other tissues. then, we will look to see the annotation of these genes

load("wang\_eset.RData")

library(Biobase)

count.matrix <- exprs(wang.eset)[,10:21]

col.data <- pData(wang.eset)[10:21,]

library(DESeq2)

dds <- DESeqDataSetFromMatrix(count.matrix, col.data, design=~cell.type)

## now we make a new factor column in the colData of dds, which is "brain" for the cerebellum

# and mixed brain samples, and "other" for the other samples. Set "other" as the reference level (the denominator for the fold changes)

dds$type <- factor(ifelse(dds$cell.type %in% c("cerebellum","mixed.brain"),

"brain", "other"))

dds$type <- relevel(dds$type, "other")

# reset the design of the DESeqDataSet

design(dds) <- ~type

# Run the differential expression function, and create the default results table 'res' for comparing brain vs other.

# make the default MA-plot

dds <- DESeq(dds)

res <- results(dds)

plotMA(res)

res2 <- results(dds, lfcThreshold = 10)

plotMA(res2, ylim=c(-10,10))

res[which.min(res$padj),]

# use plotCounts to make a plot of the normalized counts for this gene

idx <- which.min(res$padj)

plotCounts(res, idx, intgroup="type")

# create a results table 'res2' with a lfcThreshold of 2.

res2 <- results(dds, lfcThreshold = 2)

plotMA(res2, ylim=c(-10, 10))

# use summary() on the results table you just made which tested for absolute value of log2 fold changes

# larger than 2 (so a fold change of more than 4 or less than 1/4). How many genes in the set with FDR less than 0.1

# have a positive LFC?

summary(res2)

## continute using the results table 'res' from the brain vs other sample comparison

plotCounts(dds, which.min(res$padj), intgroup='type')

# make normalized counts plots for the top 9 genes

par(mfrow=c(3,3))

for (i in 1:9) plotCounts(dds, order(res$padj)[i], intgroup='type')

# we have empirically found a set of genes which seem specific to brain, we examine their annotation below

par(mfrow=c(1,1))

top <- rownames(res)[head(order(res$stat, decreasing=TRUE), 20)]

# use org.Hs.eg.db to determine the gene symbol of the top gene in this list. what is the SYMBOL?

library(org.Hs.eg.db)

select(org.Hs.eg.db, keys=top, columns="SYMBOL", keytype="ENSEMBL")

# use org.Hs.eg.db to determine the GENENAME of the top genes.

select(org.Hs.eg.db, keys=top, columns="GENENAME", keytype="ENSEMBL")

## in this assessment, we will download a dataset which has 2 biological conditions and 3 experimental batches

## and see how well SVA does at detecting the batches.

# download.file("http://bowtie-bio.sourceforge.net/recount/ExpressionSets/bottomly\_eset.RData", "bottomly\_eset.RData")

load("bottomly\_eset.RData")

library(Biobase)

# build a DESeqDataSet from this object

count.matrix <- exprs(bottomly.eset)

col.data <- pData(bottomly.eset)

library(DESeq2)

dds <- DESeqDataSetFromMatrix(count.matrix, col.data, design = ~strain)

# the experiment.number column is a numeric, so make sure to turn it into a factor

dds$experiment.number <- factor(dds$experiment.number)

# estimate the size factor so we can get normalized counts later:

dds <- estimateSizeFactors(dds)

# run the varianceStabilizingTransformation() on the dds and then make a PCA plot with c("strain","experiment.number") as the intgroup to label

vsd <- varianceStabilizingTransformation(dds)

plotPCA(vsd, intgroup=c("strain", "experiment.number"))

# we can see that both strain and experimental batch have an effect on the normalized, transformed counts

# because we known the experimental batches, we could just use DESeq() with ~experiment.number + strain

# to look for strain specific differences controlling for batch. but suppose we were givien this data without the batch

# information, we could use SVA to identify the hidden structure

# run sva-seq to find 2 surrogate variables

library(sva)

dat <- counts(dds, normalized=T)

idx <- rowMeans(dat) > 1

dat <- dat[idx,]

mod <- model.matrix(~ strain, colData(dds))

mod0 <- model.matrix(~ 1, colData(dds))

svseq <- svaseq(dat, mod, mod0, n.sv=2)

plot(svseq$sv[,1], svseq$sv[,2], col=dds$experiment.number, pch=16)

legend("bottom", levels(dds$experiment.number), pch=16, col=1:3)

text(svseq$sv[,1], svseq$sv[,2], 1:ncol(dds), pos=1)

# we will look for differential exon usage in the same experimental data as in the video

# we will use a different subset of the genes (to speed up the time required)

# build a DEXSeq dataset object

library("pasilla")

inDir <- system.file("extdata", package="pasilla", mustWork=TRUE)

countFiles <- list.files(inDir, pattern="fb.txt$", full.names=T)

flattenedFile <- list.files(inDir, pattern="gff$", full.names=T)

sampleTable = data.frame(row.names = c( "treated1", "treated2", "treated3","untreated1", "untreated2", "untreated3", "untreated4" ), condition = c("knockdown", "knockdown", "knockdown", "control", "control", "control", "control" ))

library(DEXSeq)

dxd = DEXSeqDataSetFromHTSeq(countFiles, sampleData=sampleTable,

design= ~ sample + exon + condition:exon, flattenedfile=flattenedFile )

# now we will subset to 1000 genes on chr2L which do not have low counts

rmean <- rowMeans(counts(dxd))

# use rowRanges to pull out chr2L

dxd2L <- dxd[seqnames(rowRanges(dxd)) == "chr2L" & rmean > 10, ]

# subset to first 1000

dxd2L = dxd2L[1:1000,]

# Exon usage Q1

# what is the gene name of the gene with the exon with the smallest adjusted p-value for differential exon usage?

dxd2L <- estimateSizeFactors(dxd2L)

dxd2L <- estimateDispersions(dxd2L)

dxd2L <- testForDEU(dxd2L)

dxd2L <- estimateExonFoldChanges(dxd2L, fitExpToVar = "condition")

res <- DEXSeqResults(dxd2L)

res[which.min(res$padj),]

# make a DEXSeq plot of the DEXSeq results object for this gene

plotDEXSeq( res, "FBgn0000256", legend=TRUE, cex.axis=1.2, cex=1.3, lwd=2, norCounts = T, displayTranscripts = T )

# in this assessment, we will examine the isoform-level abundances which are saved as output from cufflinks

# goal: for each gene: how often is the most highly expressed isoform the same across two biological conditions?

# create a CuffSet object

library(cummeRbund)

myDir <- system.file("extdata", package="cummeRbund")

gtfFile <- system.file("extdata/chr1\_snippet.gtf", package="cummeRbund")

cuff <- readCufflinks(dir=myDir, gtfFile = gtfFile, genome="hg19", rebuild=T)

# extract the annotation information with the annotation() function. this gives exon-level information.

gene.features <- annotation(genes(cuff))

head(gene.features)

isoforms.per.gene <- gene.features[!duplicated(gene.features$isoform\_id), c("gene\_id","isoform\_id")]

isoforms.per.gene <- isoforms.per.gene[order(isoforms.per.gene$isoform\_id),]

head(isoforms.per.gene)

gene.tab <- table(isoforms.per.gene$gene\_id)

# how many genes have only 1 isoform

sum(gene.tab==1)

# the fpkm() function returns a data.frame of the FPKM estimates for each isoform and sample

isoform.fpkm <- fpkm(isoforms(cuff))

head(isoform.fpkm)

table(isoform.fpkm$sample\_name)

# extract out tables for the iPS and hESC samples

ips <- isoform.fpkm[isoform.fpkm$sample\_name == "iPS",]

hesc <- isoform.fpkm[isoform.fpkm$sample\_name == "hESC",]

# check that the isoform\_id from our FPKM tables and our isoforms-per-gene table are identical

stopifnot(all(ips$isoform\_id == isoforms.per.gene$isoform\_id))

stopifnot(all(hesc$isoform\_id == isoforms.per.gene$isoform\_id))

# use sapply(), split() and which.max() to identify for each sample the index of the isoform with the largest FPKM

ips.max <- sapply(split(ips$fpkm, isoforms.per.gene$gene\_id), which.max)

hesc.max <- sapply(split(hesc$fpkm, isoforms.per.gene$gene\_id), which.max)

sum(ips.max == hesc.max)/400

indx <- which(gene.tab>1)

ips.max2 <- ips.max[indx]

hesc.max2 <- hesc.max[indx]

mean(ips.max2 == hesc.max2)