Lab #6

This lab is due March 2nd, send what you have (code,plots and answer to questions) to [afodor@uncc.edu](mailto:afodor@uncc.edu). Make sure the text “Lab #6” is in the subject line…

Our lab on March 1st can be used to finish this lab and to work on the midterm study guide (which will be given out next week). There will not be another lab assignment on the March 1st lab.

The mid-term will be Tuesday March 13th.

-------------------------

(1) Install DeSeq version 1 in your version of R.

In R type:

source("http://bioconductor.org/biocLite.R")

biocLite("DESeq")

library("DESeq")

See (<http://bioconductor.org/packages/release/bioc/html/DESeq.html> ) for more information.

(2) Download this dataset:

http://afodor.github.io/classes/stats2015/longitdunalRNASeqData.zip

(right-click in the browser and say “save as…”)

and grab nc101\_scaff\_dataCounts.txt from the zip archive.

(3) Read the dataset into R using commands something like...

setwd("C:\\classes\\Advanced\_Stats\_Spring2015\\Lab4\_HW1")

myT <- read.table("nc101\_scaff\_dataCounts.txt",header=TRUE,row.names=1)

(4) In order to compare our results to DeSeq at the end of the lab, we will use DeSeq to normalize our spreadsheet.

That can be done with this code…

timepoints <- c("D2","D2","D2", "w12","w12","w12","w20","w20","w20","w20","w20")

cds <- newCountDataSet(myT, timepoints )

cds <- estimateSizeFactors( cds)

cds <- estimateDispersions(cds,sharingMode="gene-est-only")

myTNorm <- counts(cds,normalized=TRUE) + 1

Note that we add a pseudo-count to the normalized spreadsheet that DeSEQ hands back to us. We will only use this myTNorm spreadsheet for all the calculations below. (We will discuss why DeSeq gave us a warning after the call to estimateDispersions in lecture #10).

(5) Show the plot for the variance (with something like apply(myTNorm,1,var) ) vs. the means (with something like apply(myTNorm,1,mean) on a log-log scale. Does the assumption that the mean equals the variance seem reasonable for our dataset?

(6) Consider the first gene in our normalized spreadsheet

> myTNorm[1,]

D2\_01 D2\_02 D2\_03 W12\_01 W12\_02 W12\_03 w20\_01 w20\_02

387.15781 110.99981 333.48564 203.70763 186.55593 167.30492 208.14927 258.15977

w20\_03 w20\_04 w20\_05

242.07300 227.10468 44.37141

>

Consider just the two week (columns 1:3) and twenty week (columns 7:11) samples. If we wanted to use the Poisson test for these two time points, we could consider that in all the two week samples there was an average of 644367.2 sequences per sample

> apply(myTNorm[,1:3],2,sum)

D2\_01 D2\_02 D2\_03

685877.9 605184.5 642039.2

> (685877.9 + 605184.5 + 642039.2) /3

[1] 644367.2

For the first gene for the two week samples, there was an average of 277.21 reads. So the fraction of reads assigned to this gene was 277.21/644367.2 = 0.00043.

For the twenty week timepoint, there was an average of 522870.5 reads per samples and for the first gene had an average of 195.9716 reads for these samples. So we can use the Poisson test to generate a p-value for the null hypothesis that the distribution of the gene was the same at the two time points:

poisson.test( round(195.9716), 522870.5, 0.00043, alternative="two.sided" )$p.value

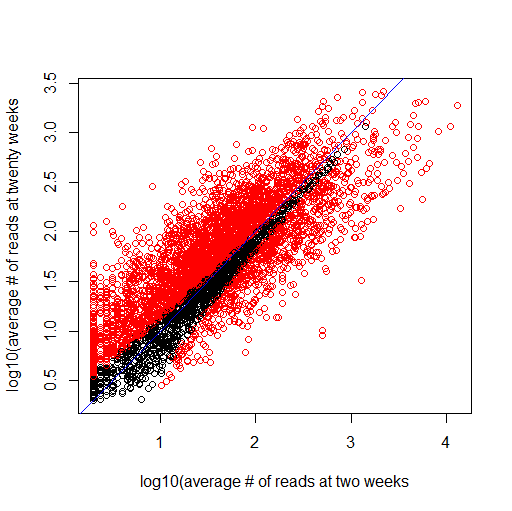
or around 0.057.

Now calculate this p-value for each gene in the spreadsheet for the two week vs. twenty week timepoint. Graph the average number of sequences in the two weeks time point vs. the average number of sequences in the twenty week time point on a log-log scale. Color your points by

ifelse(p.adjust(pValuesPoisson,method="BH") < .1, "red","black" )

(where pValuesPoission is a vector that holds all the results of your Poisson tests. (We’ll see what p.adjust does in the next lecture, but every red symbol is significant under the Poisson test at a 10% false discovery rate).

To check your code, this is what I got (where the blue line is the identity line):



(7) Now we will repeat this statistical test, but instead of the Poisson test, we will use a negative binomial distribution based test. In order to be able to compare our results to DeSeq, we will use DeSeq’s estimate of each genes variance. Which we can get by:

cds <- newCountDataSet(myT, timepoints )

cds <- estimateSizeFactors( cds)

cds <- estimateDispersions(cds,sharingMode="gene-est-only")

res <- nbinomTest(cds, "D2", "w20")

means <- apply(counts(cds,normalized=TRUE), 1,mean)

perGeneEstimates <- fitInfo(cds)$perGeneDispEsts \* means \* means

(We’ll get into this in lecture #10)

Consider again the first gene in the spreadsheet with an average number of reads of 277.21 for the two week samples. perGeneEstimates[1] tells us that DeSeq’s estimate of the variance for this gene (across all samples) is 8822.634. We calculate p and r:

p = 277.21 / 8822.634= 0.03142032

r = 277.21^2 / (8822.634-277.21) = 8.992577

For the samples at twenty weeks, we saw an average of 195.9716 for this gene. We can use the negative binomial distribution with parameters for the two week samples to test the null hypothesis that the distribution of reads is the same at the two timepoints:

pnbinom(round(195.9716), 8.992577, 0.03142032)

which comes out to ~0.199. But we want the two-sided test so we multiply this by two to get our final p-value of ~0.39.

(Note that if the initial one-sided p-values was >0.5, our final p-value would be

2 \* (1-initialPValue))

Now, repeat this negative binomial test for ever gene in the spreadsheet. Make a graph comparing the average # of reads at two weeks vs. twenty weeks (on a log-scale) but this time color by:

ifelse(p.adjust(pValuesNegativeBinomial,method="BH") < .1, "red","black" )

where pValuesNegativeBinomial holds the results of your binomial tests.

Which test (Poisson or negative binomial) is more conservative? Which do you think is more appropriate for our dataset (and why?).

(8) Finally, we will compare our results to DeSeq’s statistical test. As above run

cds <- newCountDataSet(myT, timepoints )

cds <- estimateSizeFactors( cds)

cds <- estimateDispersions(cds,sharingMode="gene-est-only")

res <- nbinomTest(cds, "D2", "w20")

Graph the -10(pValues from your negative test) vs. DeSeq’s results (which you can get from res$pval after running the code above. As a check on your code, here is my graph (next page). You can see that our code give broadly similar results to DeSeq, although our much simpler code is not exactly the same as DeSeq’s more complex analysis (which will touch on in lecture #10).

