**RNAseq analysis methods**

**Prerequisites (already installed on VirtualMachine):**

Python

R

EdgeR (installation described below within R)

HTSeq <http://www-huber.embl.de/users/anders/HTSeq/doc/overview.html> (requires NumPy)

Tophat <https://ccb.jhu.edu/software/tophat/index.shtml>

Bowtie2 <http://bowtie-bio.sourceforge.net/bowtie2/index.shtml>

Pysam <https://code.google.com/p/pysam/downloads/list>

Samtools <http://www.htslib.org/download/>

**Worksheet:**

1. We have RNAseq data from chr22 of the human genome. The reads have already been de-multiplexed, adapter trimmed, and quality trimmed. The necessary reference files (gtf and fasta) have already been downloaded and are available in the data folder. The files were downloaded from:

Fasta file for chr22 is found on this page: <http://hgdownload.cse.ucsc.edu/goldenPath/hg38/chromosomes/>

Gtf file for whole genome is found here:

<ftp://ftp.ensembl.org/pub/release-81/gtf/homo_sapiens>

We want to limit to the genes found on chromosome 22, so I used this command to pull out those genes:

grep ^22 humangenome.gtf > chr22.gtf

A bit more about gtf format:

<http://www.ensembl.org/info/website/upload/gff.html>

1. Index the reference using bowtie build:

bowtie2­build chr22.fa chr22

1. Align the reads to the reference fasta using tophat. Note: bowtie2 must be in your PATH. All programs have already been installed and are located in your PATH.

First it is necessary to make the directory that all data will be written to, note: this directory only needs to be created once.

mkdir alignments

tophat2 -o alignments/1 chr22 data/sample\_01\_1.fasta data/sample\_01\_2.fasta

1. Now that you have a bam file, you will want to check the mapping quality. Note: The output of samtools idxstats is tab-delimited with each line consisting of reference sequence name, sequence length, # mapped reads and # unmapped reads.

samtools index alignments/01/accepted\_hits.bam

samtools idxstats alignments/01/accepted\_hits.bam

1. Sort the resulting bam files by name (HTSeq-count requires name sorting, many other programs require coordinate sorting). Repeat for all 20 alignments.

samtools sort -n alignments/01/accepted\_hits.bam alignments/01/accepted\_hits\_sorted

1. Next use HTseq­count to count the number of reads that map to each gene on a per individual basis, –stranded=no is necessary as this was an unstranded library preparation (it’s important to know how your data was generated), –r specifies that the bam was name sorted.

htseq-count --stranded=no -r name -f bam alignments/01/accepted\_hits\_sorted.bam chr22\_genes.gtf > alignments/01/4fold\_counts.txt

1. Repeat for all 20 samples. Let’s write a script that will do this for us! The shell script that works on the VM machine is:

for i in 0{1..9} {10..20}

do

tophat2 -o alignments/$i chr22 data/sample\_${i}\_1.fasta data/sample\_${i}\_2.fasta

samtools sort -n alignments/${i}/accepted\_hits.bam alignments/${i}/accepted\_hits\_sorted

htseq-count --stranded=no -r name -f bam alignments/${i}/accepted\_hits\_sorted.bam chr22\_genes.gtf > alignments/${i}/4fold\_counts.txt

done

At this point, you will have 20 bam files, one for each individual and counts data for each individual.

1. We’ll now move to R for the remainder of our analyses. Install edgeR and open it. R, bioconductor and edgeR have already been installed on the VM. In R, the following commands were previously run and do not need to be run again.

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

biocLite("edgeR")

Open R on the command line:

R

library(edgeR)

Next, I use a program called edgeR to check how many genes are significantly differentially expressed between the two groups based on gene expression counts.  First, I need to make a table (I have named it “IndividualAssignments.txt”) that tells the program which individuals belong to which group.  Here is the top of the table.

|  |  |  |
| --- | --- | --- |
| files | group | description |
| alignments/1/4fold\_counts.txt | A | A |
| alignments/2/4fold\_counts.txt | A | A |
| alignments/3/4fold\_counts.txt | A | A |
| alignments/4/4fold\_counts.txt | A | A |
| alignments/5/4fold\_counts.txt | A | A |
| alignments/6/4fold\_counts.txt | A | A |

The description column isn’t really necessary, but additional information could be added to that column.

First, I make sure that I am working the directory that has my count tables.  Then, I use this R code to test for significantly differentially expressed genes and make an MDS plot:

# specify the name of the targets table

targets <- readTargets("IndividualAssignments.txt")

# this essentially merges the separate count tables into one count table.

# The counts are separated by group here, so this is used for significance testing

# the skip=5 and comment.char = ”!” are used for skipping comment lines in the table

d <- readDGE(targets, skip=5, comment.char = "!")

# this keeps any genes that are above 0 expression in the analysis.  Also remove features that are listed at the bottom of the HTseq output

# The first line applies a filter to the data, the second line filters out genes that fail

noint = rownames(d) %in% c("\_\_no\_feature","\_\_ambiguous","\_\_too\_low\_aQual","\_\_not\_aligned","\_\_alignment\_not\_unique")  
cpmd = cpm(d)  
keep = rowSums(cpmd > 1) >=2 & !noint  
d = d[keep,]  
d$samples$lib.size = colSums(d$counts)

# estimate the common dispersion to get an idea of the overall degree of inter-library

variability in the data:

d <- estimateCommonDisp(d, verbose=TRUE)

# estimate dispersion values for each gene (“a measure of the degree of inter-library variation for that tag”)

d <- estimateTagwiseDisp(d, trend="none")

#Note, could have run: d <- estimateDisp(d, trend="none", robust=TRUE)

# Perform an exact test to identify genes that are significantly differentially expressed

# topTags shows the 20 most significant genes (expression-wise)

et <- exactTest(d)

topTags(et, n=15)

# example output from topTags (leftmost column are gene names):

                       logFC   logCPM       PValue          FDR

ENSG00000198062  0.569129325 15.56166 1.169797e-09 1.754696e-08

ENSG00000130538  0.584161116 15.11450 4.454391e-09 2.424394e-08

ENSG00000198445  0.506921523 16.31563 4.848789e-09 2.424394e-08

ENSG00000232775 -0.583853212 15.02424 7.269566e-09 2.726087e-08

ENSG00000249263 -0.448024680 12.57215 1.078498e-02 3.235495e-02

ENSG00000172967 -0.220998324 15.79520 1.491246e-02 3.728115e-02

ENSG00000093072 -0.170222087 16.31146 4.944159e-02 1.059463e-01

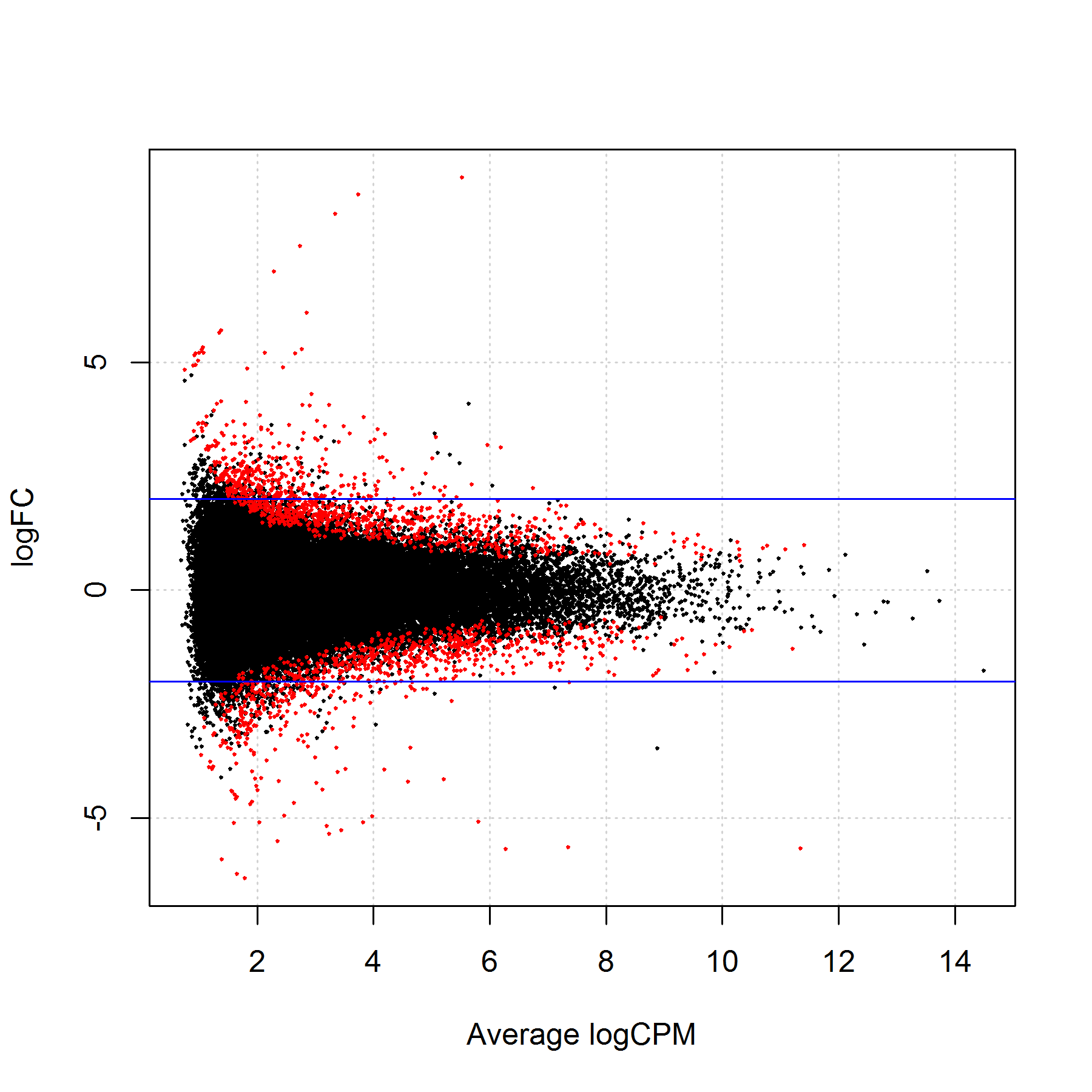
summary(de <- decideTestsDGE(et, p=0.05, adjust="BH"))

detags <- rownames(d)[as.logical(de)]

plotSmear(et, de.tags=detags)

abline(h = c(-1, 1), col = "blue")

# example plot



# Input count data for hierarchical clustering analysis (MDS)

# Plot the data with line 2

y <- cpm(d, prior.count=2, log=TRUE)

plotMDS(y)

# example plot:

