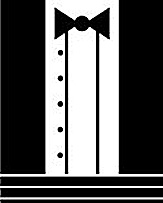
****

**A Tutorial: Genome-based RNA-Seq Analysis**

**Using the TUXEDO Package**

**(updated: 2014-10-21)**

The following details the steps involved in:

* Aligning RNA-Seq reads to a genome using Tophat
* Assembling transcript structures from read alignments using Cufflinks
* Visualizing reads and transcript structures using IGV
* Performing differential expression analysis using Cuffdiff
* Expression analysis using CummeRbund

All required software and data are provided pre-installed on a VirtualBox image. See companion ‘Rnaseq\_Workshop\_VM\_installation.pdf’ for details. Data content and environment configurations are described therein and referenced below.

**Before Running:**

After installing the VM, be sure to quickly update the contents of the rnaseq\_workshop\_data directory by:

% cd rnaseq\_workshop\_2014/

% svn up

This way, you’ll have the latest content, including any recent bugfixes.

**Data Content:**

This demo uses RNA-Seq data corresponding to Schizosaccharomyces pombe (fission yeast), involving paired-end 76 base strand-specific RNA-Seq reads corresponding to four samples: Sp\_log (logarithmic growth), Sp\_plat (plateau phase), Sp\_hs (heat shock), and Sp\_ds (diauxic shift).

There are ‘left.fq’ and ‘right.fq’ FASTQ formatted Illlumina read files for each of the four samples. Also included is a ‘genome.fa’ file corresponding to a genome sequence, and annotations for reference genes (‘genes.bed’ or ‘genes.gff3’).

*Note, although the genes, annotations, and reads represent genuine sequence data, they were artificially selected and organized for use in this tutorial, so as to provide varied levels of expression in a very small data set, which could be processed and analyzed within an approximately one hour time session and with minimal computing resources.*

**Automated and Interactive Execution of Activities**

To avoid having to cut/paste the numerous commands shown below into a unix terminal, the VM includes a script ‘runTrinityDemo.pl’ that enables you to run each of the steps interactively. To begin, simply run:

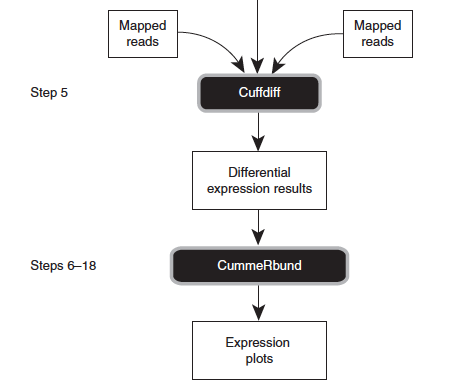
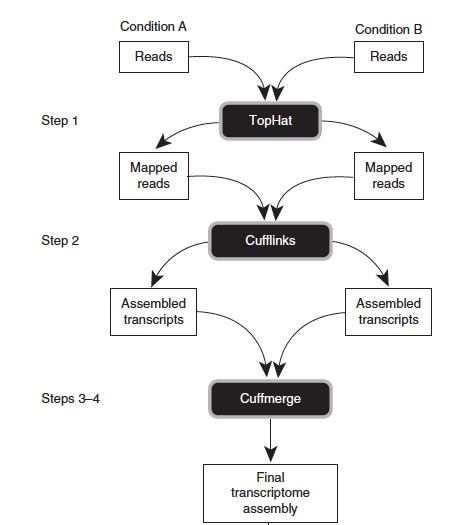
% ./runTuxedoDemo.pl

The protocol followed below is that described in

Trapnell C, Roberts A, Goff L, Pertea G, Kim D, Kelley DR, Pimentel H, Salzberg SL, Rinn JL, Pachter L. Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. Nat Protoc. 2012 Mar 1;7(3):562-78. doi: 10.1038/nprot.2012.016.

<http://www.nature.com/nprot/journal/v7/n3/full/nprot.2012.016.html>

as illustrated below:



**Use Tophat and Cufflinks to align reads and assemble transcripts**

First, prepare the ‘genome.fa’ file for tophat alignment:

% bowtie-build genome.fa genome

**(a) Align reads and assemble transcripts for sample: Sp\_ds:**

Align reads using tophat:

% tophat -I 1000 -i 20 --bowtie1 --library-type fr-firststrand -o tophat.Sp\_ds.dir genome Sp\_ds.left.fq Sp\_ds.right.fq

Rename the alignment (bam) output file according to this sample name:

% mv tophat.Sp\_ds.dir/accepted\_hits.bam tophat.Sp\_ds.dir/Sp\_ds.bam

Index this bam file for later viewing using IGV:

% samtools index tophat.Sp\_ds.dir/Sp\_ds.bam

Reconstruct transcripts for this sample using Cufflinks:

% cufflinks --overlap-radius 1 --library-type fr-firststrand -o cufflinks.Sp\_ds.dir tophat.Sp\_ds.dir/Sp\_ds.bam

Rename the cufflinks transcript structure output file according to this sample:

% mv cufflinks.Sp\_ds.dir/transcripts.gtf cufflinks.Sp\_ds.dir/Sp\_ds.transcripts.gtf

Now, you’re done with running Tuxedo on this sample. You now need to repeat these operations for each of the three other samples, as below:

**(b) Align reads and assemble transcripts for sample: Sp\_hs:**

% tophat -I 1000 -i 20 --bowtie1 --library-type fr-firststrand -o tophat.Sp\_hs.dir genome Sp\_hs.left.fq Sp\_hs.right.fq

% mv tophat.Sp\_hs.dir/accepted\_hits.bam tophat.Sp\_hs.dir/Sp\_hs.bam

% samtools index tophat.Sp\_hs.dir/Sp\_hs.bam

% cufflinks --overlap-radius 1 --library-type fr-firststrand -o cufflinks.Sp\_hs.dir tophat.Sp\_hs.dir/Sp\_hs.bam

% mv cufflinks.Sp\_hs.dir/transcripts.gtf cufflinks.Sp\_hs.dir/Sp\_hs.transcripts.gtf

**(c) Align reads and assemble transcripts for sample: Sp\_log:**

% tophat -I 1000 -i 20 --bowtie1 --library-type fr-firststrand -o tophat.Sp\_log.dir genome Sp\_log.left.fq Sp\_log.right.fq

% mv tophat.Sp\_log.dir/accepted\_hits.bam tophat.Sp\_log.dir/Sp\_log.bam

% samtools index tophat.Sp\_log.dir/Sp\_log.bam

% cufflinks --overlap-radius 1 --library-type fr-firststrand -o cufflinks.Sp\_log.dir tophat.Sp\_log.dir/Sp\_log.bam

% mv cufflinks.Sp\_log.dir/transcripts.gtf cufflinks.Sp\_log.dir/Sp\_log.transcripts.gtf

**(d) Align reads and assemble transcripts for sample: Sp\_plat:**

% tophat -I 1000 -i 20 --bowtie1 --library-type fr-firststrand -o tophat.Sp\_plat.dir genome Sp\_plat.left.fq Sp\_plat.right.fq

% mv tophat.Sp\_plat.dir/accepted\_hits.bam tophat.Sp\_plat.dir/Sp\_plat.bam

% samtools index tophat.Sp\_plat.dir/Sp\_plat.bam

% cufflinks --overlap-radius 1 --library-type fr-firststrand -o cufflinks.Sp\_plat.dir tophat.Sp\_plat.dir/Sp\_plat.bam

% mv cufflinks.Sp\_plat.dir/transcripts.gtf cufflinks.Sp\_plat.dir/Sp\_plat.transcripts.gtf

**Merge separately assembled transcript structures into a cohesive set:**

First, create a file that lists the names of the files containing the separately reconstructed transcripts, which can be done like so, writing each of the four Cufflinks transcript GTF files to a newly created ‘assemblies.txt’ file.

% echo cufflinks.Sp\_ds.dir/Sp\_ds.transcripts.gtf >> assemblies.txt

% echo cufflinks.Sp\_hs.dir/Sp\_hs.transcripts.gtf >> assemblies.txt

% echo cufflinks.Sp\_log.dir/Sp\_log.transcripts.gtf >> assemblies.txt

% echo cufflinks.Sp\_plat.dir/Sp\_plat.transcripts.gtf >> assemblies.txt

# verify that this file now contains both filenames:

% cat assemblies.txt

cufflinks.Sp\_ds.dir/Sp\_ds.transcripts.gtf

cufflinks.Sp\_hs.dir/Sp\_hs.transcripts.gtf

cufflinks.Sp\_log.dir/Sp\_log.transcripts.gtf

cufflinks.Sp\_plat.dir/Sp\_plat.transcripts.gtf

And now we’re ready to merge the transcripts using cuffmerge:

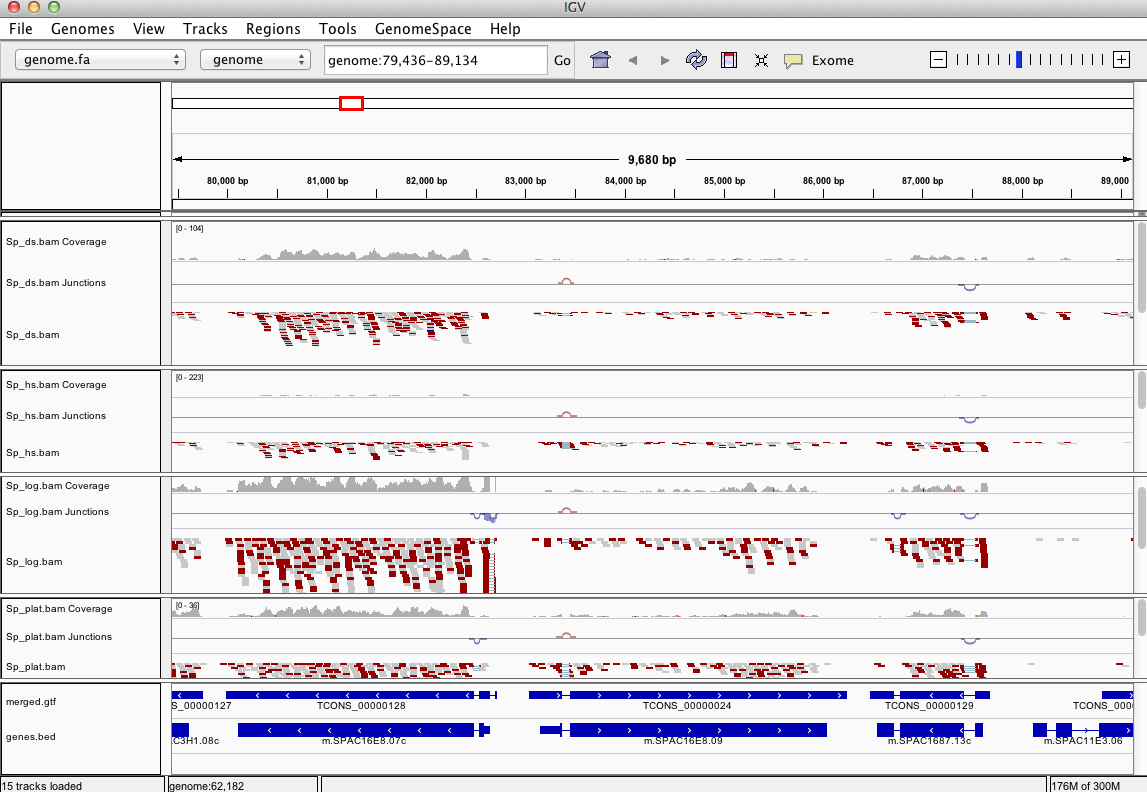
% cuffmerge -s genome.fa assemblies.txt

The merged set of transcripts should now exist as file “merged\_asm/merged.gtf’.

**View the reconstructed transcripts and the tophat alignments in IGV**

% java -Xmx2G -jar /Users/bhaas/IGV/current//igv.jar -g `pwd`/genome.fa `pwd`/merged\_asm/merged.gtf,`pwd`/genes.bed,`pwd`/tophat.Sp\_ds.dir/Sp\_ds.bam,`pwd`/tophat.Sp\_hs.dir/Sp\_hs.bam,`pwd`/tophat.Sp\_log.dir/Sp\_log.bam,`pwd`/tophat.Sp\_plat.dir/Sp\_plat.bam

*(Note, you may need to resize the individual alignment patterns in the viewer by dragging the panel boundaries. Afterwards, go to menu “Tracks” -> “Fit Data To Window” to re-space the contents of the viewer)*



Pan the genome, examine the alignments, known genes and reconstructed genes.

Do the alignments agree with the known gene structures (ex. intron placements)?

Do the cufflinks-reconstructed transcripts well represent the alignments?

Do the cufflinks-reconstructed transcripts match the structures of the known transcripts?

**Identify differentially expressed transcripts using Cuffdiff:**

% cuffdiff --library-type fr-firststrand -o diff\_out -b genome.fa   
-L Sp\_ds,Sp\_hs,Sp\_log,Sp\_plat -u merged\_asm/merged.gtf tophat.Sp\_ds.dir/Sp\_ds.bam tophat.Sp\_hs.dir/Sp\_hs.bam tophat.Sp\_log.dir/Sp\_log.bam tophat.Sp\_plat.dir/Sp\_plat.bam

Examine the output files generated in the diff\_out/ directory.

A table containing the results from the gene-level differential expression analysis can be found as ‘diff\_out/gene\_exp.diff’. Examine the top lines of this file like so:

% head diff\_out/gene\_exp.diff

**Study transcript expression and analyze DE using CummeRbund:**

Use ‘cummeRbund’ to analyze the results from cuffdiff:

% R

(note, to exit R, type cntrl-D, or type “q()” ).

# load the cummerbund library into the R session

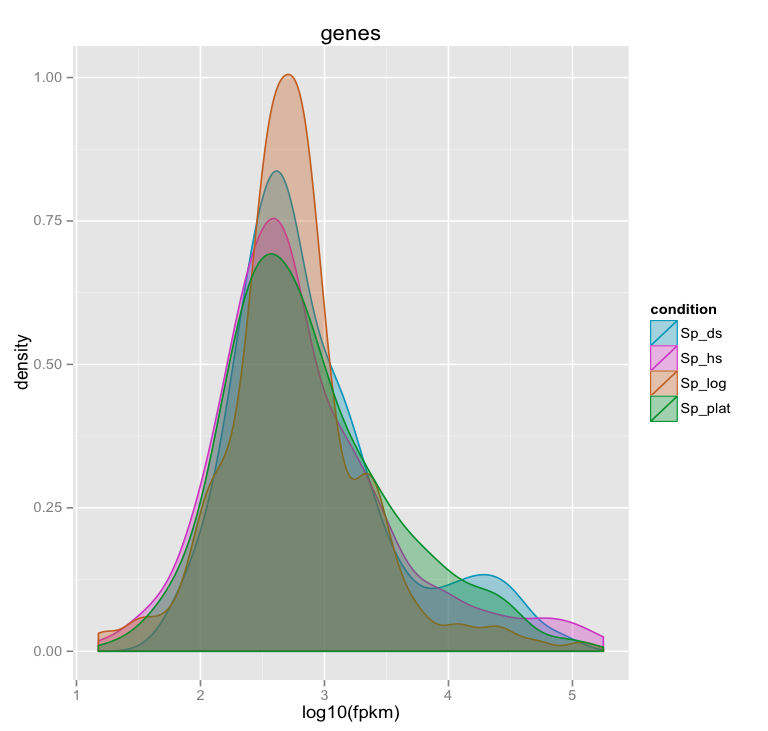
> library(cummeRbund)

# import the cuffdiff results

>cuff = readCufflinks('diff\_out')

# examine the distribution of expression values for the reconstructed transcripts

>csDensity(genes(cuff))

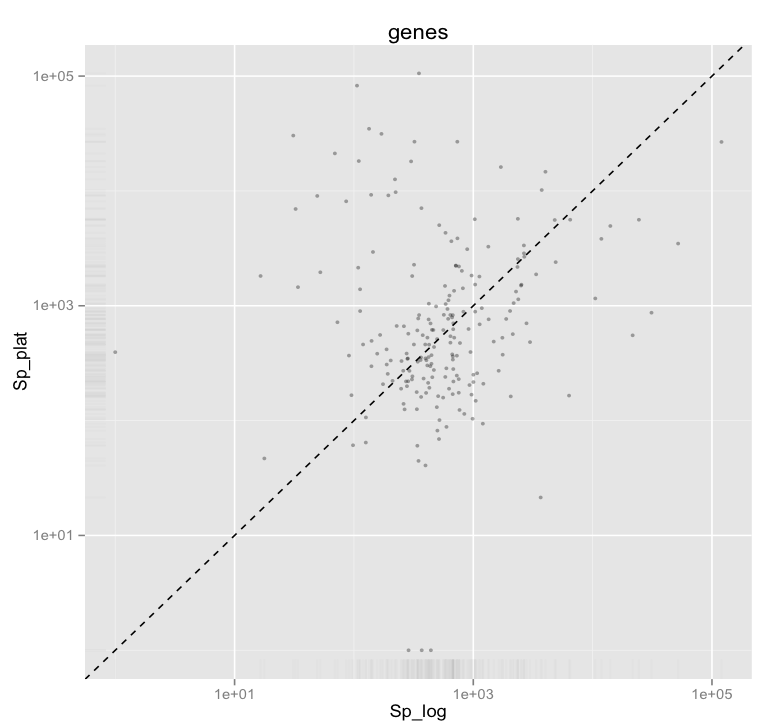


*\*\*(Note, in the Ubuntu VM, you may need to resize the graphics window in order for the plot to render correctly. This is a bug with the version of R being used.)*

# Examine transcript expression values in a scatter plot

Expression values are typically log-normally distributed. This is just a sanity check.

>csScatter(genes(cuff), ‘Sp\_log’, ‘Sp\_plat’)

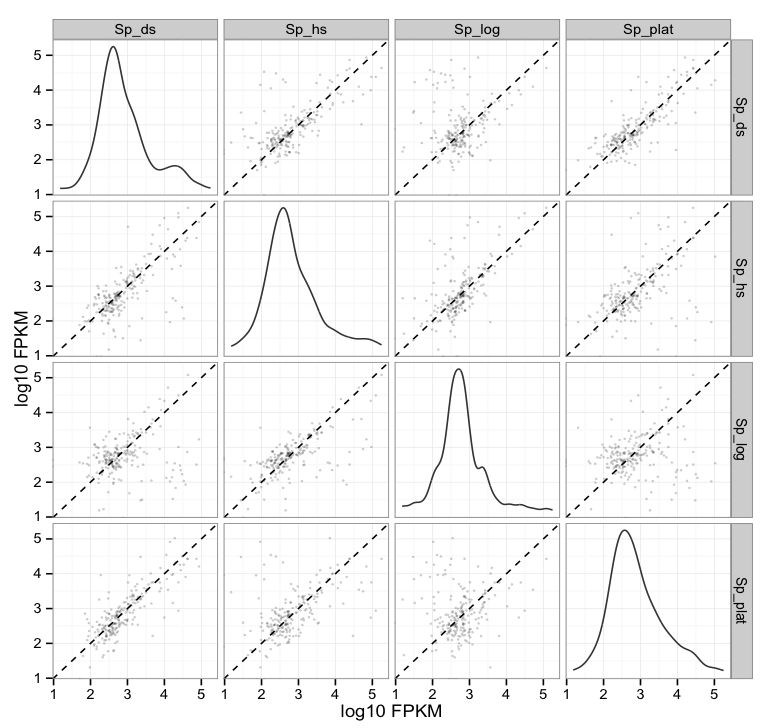


Strongly differentially expressed transcripts should fall far from the linear regression line.

*(Note, in the Ubuntu VM, you may need to resize the graphics window in order for the plot to render correctly. This is a bug with the version of R being used.)*

# Examine individual sample distributions of gene expression values and the pairwise scatterplots together in a single plot.

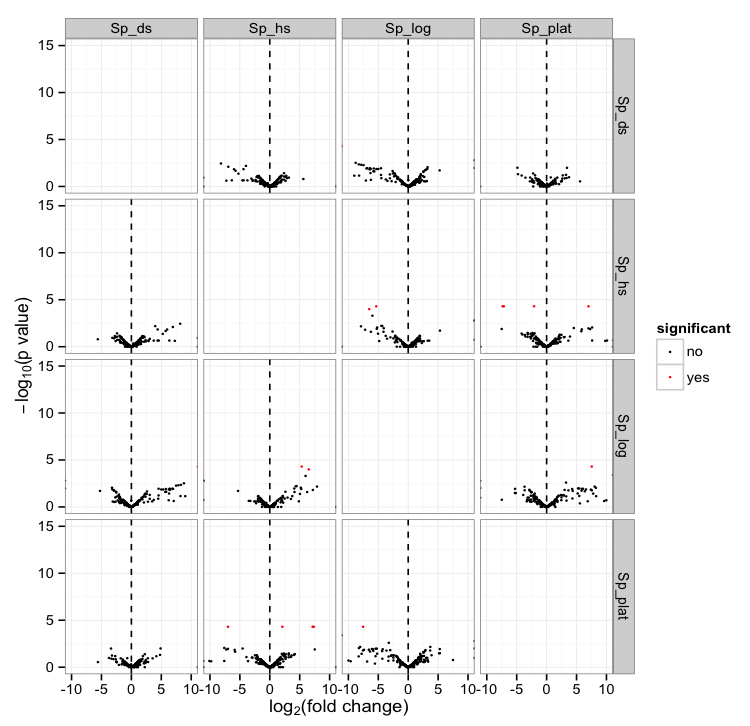
> csScatterMatrix(genes(cuff))



*(Note, in the Ubuntu VM, you may need to resize the graphics window in order for the plot to render correctly. This is a bug with the version of R being used.)*

# Volcano plots are useful for identifying genes most significantly differentially expressed.

> csVolcanoMatrix(genes(cuff))



*(Note, in the Ubuntu VM, you may need to resize the graphics window in order for the plot to render correctly. This is a bug with the version of R being used.)*

## Extract the ‘genes’ that are significantly differentially expressed (red points above)

# retrieve the gene-level differential expression data

> gene\_diff\_data = diffData(genes(cuff))

# how many ‘genes’ are there?

> nrow(gene\_diff\_data)

# from the gene-level differential expression data, extract those that

# are labeled as significantly different.

# note, normally just set criteria as “significant=’yes’”, but we’re adding an

# additional p\_value filter just to capture some additional transcripts for   
# demonstration purposes only. This simulated data is overly sparse and actually  
# suboptimal for this demonstration (in hindsight).

>sig\_gene\_data = subset(gene\_diff\_data,(significant=='yes' | p\_value < 0.01))

# how many genes are significantly DE according to these criteria?

> nrow(sig\_gene\_data)

# Examine the entries at the top of the unsorted data table:

> head(sig\_gene\_data)

gene\_id sample\_1 sample\_2 status value\_1 value\_2 log2\_fold\_change

56 XLOC\_000056 Sp\_ds Sp\_hs OK 33560.500 117.6900 -8.15563

136 XLOC\_000136 Sp\_ds Sp\_hs OK 30094.800 246.0650 -6.93433

146 XLOC\_000146 Sp\_ds Sp\_hs OK 1125.700 70.9957 -3.98694

269 XLOC\_000056 Sp\_ds Sp\_log OK 33560.500 108.8130 -8.26877

315 XLOC\_000102 Sp\_ds Sp\_log OK 0.000 440.3590 Inf

336 XLOC\_000123 Sp\_ds Sp\_log OK 753.187 0.0000 -Inf

test\_stat p\_value q\_value significant

56 -4.86421 0.00360 0.257929 no

136 -4.49008 0.00795 0.291731 no

146 -3.99117 0.00640 0.291731 no

269 -4.76096 0.00450 0.291731 no

315 NA 0.00160 0.143550 no

336 NA 0.00005 0.008700 yes

# You can write the list of significantly differentially expressed genes to a file like so:

> write.table(sig\_gene\_data, 'sig\_diff\_genes.txt', sep = '\t', quote = F)

# examine the expression values for one of your genes that’s diff. expressed:

# select expression info for the one gene by its gene identifier:

# let’s take the first gene identifier in our sig\_gene\_data table:

# first, get its gene\_id

> ex\_gene\_id = sig\_gene\_data$gene\_id[1]

# print its value to the screen:

>ex\_gene\_id

# get that gene ‘object’ from cummeRbund and assign it to variable ‘ex\_gene’

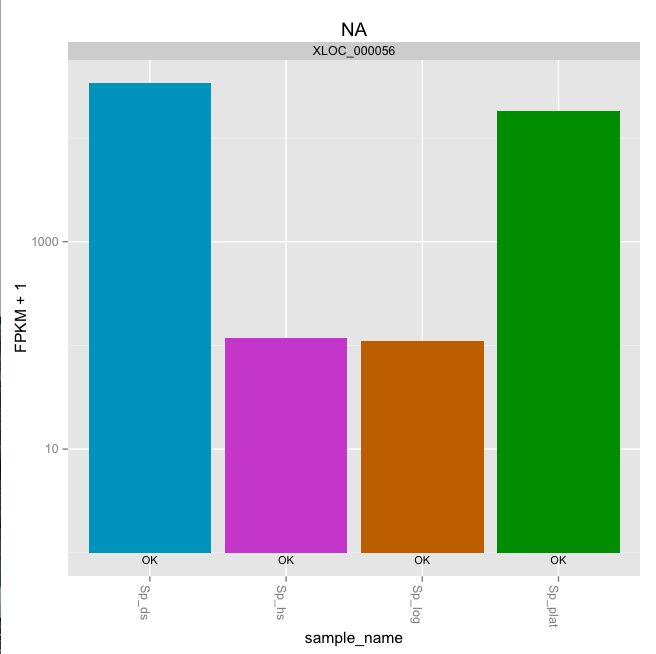
>ex\_gene = getGene(cuff, ex\_gene\_id)

# now plot the expression values for the gene under each condition

# (error bars are only turned off here because this data set is both simulated

# and hugely underpowered to have reasonable confidence levels)

> expressionBarplot(ex\_gene, logMode=T, showErrorbars=F)



*(Note, in the Ubuntu VM, you may need to resize the graphics window in order for the plot to render correctly. This is a bug with the version of R being used.)*

## Draw a heatmap showing the differentially expressed genes

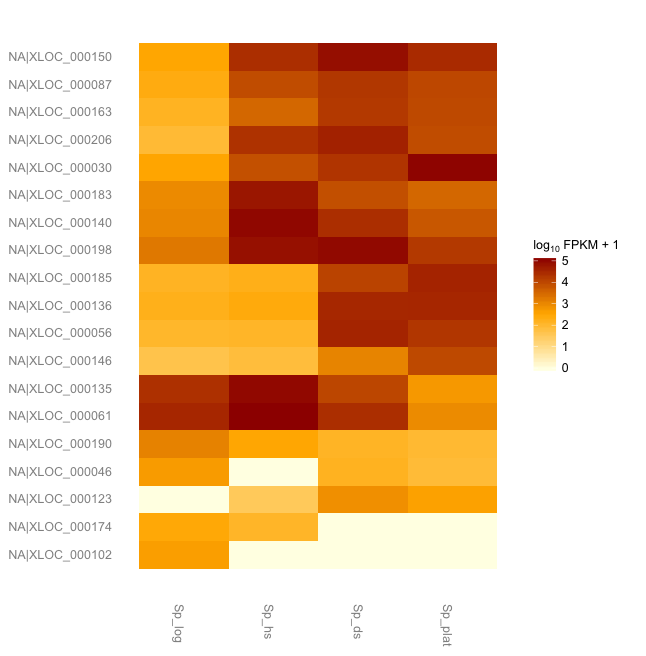
# first retrieve the ‘genes’ from the ‘cuff’ data set by providing a

# a list of gene identifiers like so:

>sig\_genes = getGenes(cuff, sig\_gene\_data$gene\_id)

# now draw the heatmap

>csHeatmap(sig\_genes, cluster='both')



*(Note, in the Ubuntu VM, you may need to resize the graphics window in order for the plot to render correctly. This is a bug with the version of R being used.)*

**More information on using the Tuxedo package can be found at:**

Trapnell C, Roberts A, Goff L, Pertea G, Kim D, Kelley DR, Pimentel H, Salzberg SL, Rinn JL, Pachter L.

Differential gene and transcript expression analysis of RNA-seq experiments with TopHat and Cufflinks. Nat Protoc. 2012 Mar 1;7(3):562-78. doi: 10.1038/nprot.2012.016.

<http://www.nature.com/nprot/journal/v7/n3/full/nprot.2012.016.html>

The CummeRbund manual:

<http://compbio.mit.edu/cummeRbund/manual_2_0.html>

(note, most of the tutorial provided here is based on the above two resources)

and the Tuxedo tool websites:

TopHat: <http://tophat.cbcb.umd.edu/>

Cufflinks: <http://cufflinks.cbcb.umd.edu/>

CummeRbund: <http://compbio.mit.edu/cummeRbund/>