DAY 3 EXERCISES

Oksana Riba-Grognuz

Differential Gene Expression

We will be using one of the most popular programs for differential expression analyses and transcript isoform assembly from spliced alignments of RNA-seq reads: TopHat and Cufflinks. These programs are actively maintained and updated. TopHat relies on Bowtie to carry out short-read alignments. Recently, following the release of Bowtie2, new TopHat and Cufflinkes versions were released. We will be using the latest release of TopHat and Cufflinks with Bowtie1. To do so, login to vital-it and source file "Tophat/latest_versions.bashrc" as shown below. Source command will modify your $PATH as specified in latest_versions.bashrc and update environment variables specifying versions of TopHat and Cufflinks.

```
$ ssh -t username@prd.vital-it.ch ssh dee-serv0X
$ cd /scratch/cluster/weekly/username
$ source TopHat/latest_versions.bashrc
```

YOU NEED R 2.15!!!!

If everything worked the installed software versions should be:

- tophat-2.0.3.Linux_x86_64
- cufflinks-2.0.0.Linux_x86_64

Please check which versions of Cufflinks and TopHat you have. If these are different from the ones listed above, give us a sign and we will set up the correct ones.

```
$ which tophat
$ echo $TOPHAT_VERSION
$ which cufflinks
$ echo $CUFFLINKS_VERSION
```

Tophat: mapping reads to genome.

We will start by preparing all necessary input files to run TopHat. As it uses Bowtie to align short reads to a reference genome, we need to generate Bowie index for genome. For this go to GenomeSubset/Assembly/ folder and run the command bowtie-build on scaffold sequences contained in FASTA file **SINV_subset_1.fa**. Start by looking how to launch this command (HINT: we do not need any options). Note that it is convenient to use the same prefix for input fasta file and bowtie index.

```
$ cd /scratch/cluster/weekly/username/GenomeSubset/Assembly
$ ls
$ bowtie-build -h
```

Please launch the appropriate command.

Q: How many index files were generated?

We will use 75 bp single Illumina RNA-seq data from 3 conditions, each with 4 biological replicates: Queens, Workers and Males, prefixed with Q, W, M respectively. Each replicate represents a pool of individuals. A 3 digit number in file names indicates red fire ant colony from which sample was taken. Have a look in the folder RNA-seq/Raw

```
$ ls /scratch/cluster/weekly/username/RNA-seq/Raw
```

Due to time constraints we will launch TopHat and Cufflinks on 3 files. For the remainder of the pipeline we will use pre-calculated TopHat and Cufflinks results with all files. Go to the directory TopHat.

```
$ cd /scratch/cluster/weekly/username/TopHat
```

In this directory you will find a file called "file.list" that lists 3 file names in column 1 and corresponding fragment sizes in column 2. Fragment sizes must be specified when running Cufflinks on single reads (note that for paired read analyses Cufflinks calculates fragment size distribution based on alignments). Have a look on TopHat parameters.

```
$ tophat -h
```

A file launching TopHat is called runTopHat.sh, have a look on it. This script takes all files in "file.list" and processes 1 by 1. Output directories are named according to sample names (prefix of fastq files).

```
$ less runTopHat.sh
```

In the beginning of this file there is a little trick to avoid manually changing username: the variable with username is set using command "whoami". Thus if you used your username as working directory name you do **NOT** need to change links to input files (genome index and raw files location) in launch script. Optionally you can change TopHat parameters. Use nano or vi editor within Vital-IT, or scp the script to edit locally as we did on Day1. Once done, launch the script.

```
$ ./runTopHat.sh
```

Cufflinks: mapping-based (= reference-based) gene/transcript identification - separately for each sample

While TopHat is running we can get ready for the next part of the pipeline: transcript assembly. Cufflinks assembles putative transcripts based on the alignments generated by TopHat. Have a look on Cufflinks parameters and on respective launch script "runCufflinks.sh". To do this open a new terminal window or tab.

```
$ cufflinks -h
$ less runCufflinks.sh
```

When TopHat is over launch Cufflinks and while it is running examine TopHat output files. Use samtools to look into bam files. Bam is a binary version of SAM (Sequence Alignment/Map) format used for storing large nucleotide sequence alignments.

```
$ ./runCufflinks.sh
$ samtools view W403/accepted_hits.bam | less
```

Column 6 in bam file contains CIGAR strings. These strings specify the structure of alignment. For example if all 75 bp align it will state 75M, meaning 75 Match. If there is a 315bp gap in read alignment it will state 61M315N14M, meaning that first 61bp Match, then there are 315 Non-matching genomic bases, and finally 14bp Match (total 75bp of read matched). CIGAR strings with N, are thus reads likely containing splice junctions (span over several exons).

This transcript identification was performed once per sample. When Cufflinks finished assembling transcripts for each lane have a look on its output files. Next step is to merge all assemblies and thus generate a reference set that can be used for differential expression part. For this part we will be using Tophat/Cufflinks output pre-run on all RNA-seq files. This output is in the folder TopHat_full.

```
$ less M350b/Cufflinks/genes.fpkm_tracking
$ less M350b/Cufflinks/transcripts.gtf
$ cd /scratch/cluster/weekly/username/TopHat_full
```

*Cuffmerge: merging results from the 3*4=12 cufflinks analyses.*

See how to run cuffmerge.

```
$ cuffmerge -h
```

As you can see the program requires an input of gtf file list. Generate one.

```
$ ls -1 */Cufflinks/transcripts.gtf > gtf.list
```

Run Cuffmerge to combine transcripts assembled for each replicate. We will use reference file generated by Cuffmerge for differential expression analysis with Cuffdiff.

```
$ genomeRef=/scratch/cluster/weekly/username/GenomeSubset/Assembly/
SINV_subset_1.fa
$ cuffmerge -s $genomeRef gtf.list
```

Have a look on the assembled merged gtf.

```
$ less merged_asm/merged.gtf
```

Renaming our favorite candidate genes (so we can find them easily)

We will want to be able to easily identify our favorite genes after analysis. Thus identify 4 Vitellogenin proteins (Vg1, Vg2, Vg3, Vg4) and 2 Transformer proteins in the assembled “merged.gtf” we need to extract fasta sequences corresponding to coordinates in

gtf file. After that we will run Blast of these sequences against file "Proteins.fasta" containing the proteins of interest. We will format file with "Proteins.fasta" as Blast database and run translated transcript query against it.

```
$ gffread merged_asm/merged.gtf -g $genomeRef -w merged_asm/merged.fa
$ formatdb -p T -i Proteins.fasta
$ blastall -p blastx -a 1 -m 8 -i merged_asm/merged.fa -d Proteins.fasta
-v 1 -b 1 -e 1.0e-5  > merged_asm/merged.blastx.all
```

One of the Vg4 hits Additionally we will retain only hits with minimum alignment length of 150 for Transformer and over 1500 for Vitellogenins (column 4).

Q: one of the fs

```
cat merged_asm/merged.blastx.all | awk '{if($2~/Tra/ && $4>=150){print
$0} else if($4>=1500){print $0}}' > merged_asm/merged.blastx
```

Blast identifies matching transcript sequences prefixed "TCONS_" by default. As we will be working on gene level, we need to identify corresponding genes prefixed "XLOC_". Use script "get.xloc.sh" to do this. Redirect the output of this script using ">" to a file.

```
$ cat merged_asm/merged.blastx
$ ./get.xloc.sh > myLoci.txt
$ cat myLoci.txt
```

Finally we would like to update our gtf reference file with names listed in column 2 of myLoci.txt. For this we will make a backup copy of original file and then use a script "update_reference.sh" to do changes.

```
$ cp merged_asm/merged.gtf merged_asm/merged_original.gtf
$ ./update_reference.sh
$ ./update_reference.sh myLoci.txt
```

Next we will run Cuffdiff that will calculate differential expression levels for genes and isoforms, as well as differential splicing and promoter use. We will need to estimate our average fragment length (Remember: this is only necessary when analyzing single Illumina reads). Use R (on Vital-IT) to read script file.list and calculate average fragment length (specified for each file in column 2). Launch Cuffdiff or use provided launch script called "runCuffdiff.sh". If you choose to use the launch script then edit input file locations and average fragment length.

```
$ cuffdiff -h
$ ./runCuffdiff.sh
```

Have a look at output files of Cuffdiff.

```
$ ls Diff_FDR0.01
$ less Diff_FDR0.01/gene_exp.diff
$ less Diff_FDR0.01/genes.count_tracking
$ less Diff_FDR0.01/genes.read_group_tracking
```

```
$ grep Vg Diff_FDR0.01/gene_exp.diff
```

CummeRbund: Insanely great plots

We will now use R package cummeRbund developed for the analysis of Cuffdiff output. Make sure you are using the latest R version (2.15). Start by downloading all files contained in Cuffdiff output directory to your computer locally. It will be faster if you "zip -r Diff_FDR0.01.zip Diff_FDR0.01" first. [failproof backup]

```
$ mkdir Diff_FDR0.01
$ scp username@prd.vital-it.ch:/scratch/cluster/weekly/username/
TopHat_full/Diff_FDR0.01/* Diff_FDR0.01/
$ scp username@prd.vital-it.ch:/scratch/cluster/weekly/username/
TopHat_full/myLoci.txt .
$ R
```

```
# install package
# source("http://bioconductor.org/biocLite.R")
# biocLite(c("cummeRbund", "Hmisc", "gplots"))

library(cummeRbund)
options(width=65)

# import data
cuff <- readCufflinks(dir = "Diff_FDR0.01", rebuild=T)

# look at object structure
str(cuff)
genes <- genes(cuff)

# plot gene expression density
dens<-csDensity(genes)
dens

# boxplots per sample
b<-csBoxplot(genes)
b

# assess general similarity between samples
# using a scatter plot comparing gene FPKM values for each comparison
(s<-csScatter(genes,"Q", "W", smooth=T))
(s<-csScatter(genes,"Q", "M", smooth=T))
(s<-csScatter(genes,"M", "W", smooth=T))


# inspect differentially expressed genes
(v<-csVolcano(genes,"Q","W"))
(v<-csVolcano(genes,"Q","M"))
(v<-csVolcano(genes,"M","W"))

#  gene fpkm values
gene.fpkm<-fpkm(genes)
head(gene.fpkm)

# access gene fpkm as a matrix
gene.matrix<-fpkmMatrix(genes)
head(gene.matrix)

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

# gene expression differences
gene.diff<-diffData(genes)
```

```
head(gene.diff)

# extract data for significant differences
sig_gene_data <- subset(gene.diff, (significant == "yes"))
```

#Q: why significant gene names are not unique?

```
sigGenes <- unique(sig_gene_data$gene_id)
length(sigGenes)

# cummeRbund is currently implementing the support for replicates
# we extract replicate values by reading file "genes.read_group_tracking"
repData  <- read.table("Diff_FDR0.01/genes.read_group_tracking",
                       header=T, sep="\t")
sampleId <- paste(repData$condition, repData$replicate, sep="")
repData  <- data.frame(repData, "sampleId"=sampleId)
repFPKM  <- reshape(repData,  idvar="tracking_id",
                    timevar="sampleId", direction="wide",
                    drop=c(names(repData)[2:6], names(repData)[8:9]))
rowNames <- repFPKM[,1]
repFPKM  <- repFPKM[-1]
row.names(repFPKM) <- rowNames
colnames(repFPKM)  <- substr(colnames(repFPKM), 6,7)

#heatmap for significant gene values in replicates
library(gplots)

FPKM.sig     <- repFPKM[sigGenes,]
FPKM.sig     <- log2(FPKM.sig+0.5)
genes.dend   <- as.dendrogram (hclust (as.dist (1-cor (t(FPKM.sig)))))
samples.dend <- as.dendrogram (hclust (as.dist (1-cor (FPKM.sig))))
heatmap.2(as.matrix(FPKM.sig), scale="row",
          Rowv=genes.dend, Colv=samples.dend, trace="none",
          col=colorRampPalette(c("white", "yellow", "red"))(100))


sample.names<-samples(genes)
head(sample.names)
gene.featurenames <- featureNames(genes)
head(gene.featurenames)

# let's look at Vitellogenins
myGene <- getGenes(cuff,"Vg1")
myGene
expressionPlot(myGene)
fpkm(myGene)

myGene <- getGenes(cuff,"Vg2")
expressionPlot(myGene)
fpkm(myGene)

myGene <- getGenes(cuff,"Vg3")
expressionPlot(myGene)
fpkm(myGene)

myGene <- getGenes(cuff,"Vg4")
expressionPlot(myGene)
fpkm(myGene)
```

Q: What do you conclude about expression of different Vitellogenins?

```
# let's take a Vg overexpressed in Queens and find 20 similar gene profiles
mySimilar            <- findSimilar(cuff,"Vg2",n=20)
mySimilar.expression <- expressionPlot(mySimilar,logMode=T,showErrorbars=F)
print(mySimilar.expression)
```

```
# we can also define a profile to search with numeric values
# let's find genes overexpressed in both female castes: queens and workers
sample.names
myProfile             <- c(0,1000,1000)
mySimilar2            <- findSimilar(cuff,myProfile,n=10)
mySimilar2.expression <- expressionPlot(mySimilar2,logMode=T,showErrorbars=F)
print(mySimilar2.expression)

# let's find genes overexpressed in both sexual castes: queens and males
myProfile             <- c(1000,1000,0)
mySimilar2            <- findSimilar(cuff,myProfile,n=10)
mySimilar2.expression <- expressionPlot(mySimilar2,logMode=T,showErrorbars=F)
print(mySimilar2.expression)

# k-means clustering
myGenes<-getGenes(cuff,sig_gene_data$gene_id)
ic <- csCluster(myGenes, k = 4)
head(ic$cluster)
icp <- csClusterPlot(ic)
icp
```

Optional: Alternative Splicing

We know that Transformer genes (Tra1 and Tra2) have sex-specific alternative splicing. Prepare a gff file to examine possible isoforms in Apollo. Fasta sequence for scaffold containing these genes is in the file "SIgn00005.fa". First we get the coordinates of Tra1 and Tra2 genes to Tra.gtf. Then we convert gtf to gff. Finally gff and fasta files must be uploaded to local computer to visualize with Apollo.

```
$ grep Tra merged_asm/merged.gtf > Tra.gtf
$ gffread Tra.gtf -o Tra.gff
$ scp username@prd.vital-it.ch:/scratch/cluster/weekly/username/
TopHat_full/Tra.gtf .
$ scp username@prd.vital-it.ch:/scratch/cluster/weekly/username/
TopHat_full/SIgn00005.fa .
```

Q: Can you identify Tra1 and Tra2 isoforms that are likely to be artefacts?

Q: Which ones look plausible?