Script example for pipeline

Make a directory in your home folder called “RNA-Seq-analysis”.

For example purposes, download `SRR2073144.sra` and `SRR2073145.sra` using wget command.

```
wget ftp://ftp-trace.ncbi.nih.gov/sra/sra-instant/reads/ByRun/sra/SRR/SRR207/SRR2073144/SRR2073144.sra
```

```
wget ftp://ftp-trace.ncbi.nih.gov/sra/sra-instant/reads/ByRun/sra/SRR/SRR207/SRR2073145/SRR2073145.sra
```

1. Convert the SRA files to Fastq files

You will have to use sratoolkit which is available from ncbi. Once it is installed, run the following commands:

```
/home/RNA-Seq-analysis/sratoolkit.2.5.4-1-centos_linux64/bin/fastq-dump SRR2073144.sra > SRR2073143.fastq
```

```
/home/RNA-Seq-analysis/sratoolkit.2.5.4-1-centos_linux64/bin/fastq-dump SRR2073145.sra > SRR2073146.fastq
```

2. Quality metrics using FastQC

```
/home/RNA-Seq-analysis/FastQC/fastqc SRR2073144.fastq
```

```
/home/RNA-Seq-analysis/FastQC/fastqc SRR2073145.fastq
```

3. Remove adapter sequences using cutadapt

```
/home/RNA-Seq-analysis/cutadapt-1.8.3/bin/cutadapt -u 15 -o Sample-7-1-trimmed.fastq SRR2073144.fastq
```

```
/home/RNA-Seq-analysis/cutadapt-1.8.3/bin/cutadapt -u 15 -o Sample-7-2-trimmed.fastq SRR2073145.fastq
```

4. Repeat quality metrics using FastQC to check for any errors*****

```
/home/RNA-Seq-analysis/FastQC/fastqc Sample-7-1-trimmed.fastq
```

```
/home/RNA-Seq-analysis/FastQC/fastqc Sample-7-2-trimmed.fastq
```

5. Perform read alignment using Tophat2

```
PATH=$PATH:/home/deshpan4/RNA-Seq/bowtie2-2.2.6/
```

```
export PATH
```

```
nohup /home/deshpan4/RNA-Seq/tophat-2.1.0.Linux_x86_64/tophat2 -p 8 --max-multihits 1 -i 40 -I 5000 --library-type fr-unstranded --b2-very-sensitive --segment-length 20 --segment-mismatches 2 -F 0 -g 1 -a 10 -G
```

```
/home/deshpan4/RNA-
Seq/Arabidopsis_thaliana/Ensembl/TAIR10/Annotation/Genes/genes.gtf -o Sample-
10-replicate-aligned-tophat2 /home/deshpan4/RNA-
Seq/Arabidopsis_thaliana/Ensembl/TAIR10/Sequence/Bowtie2Index/genome ../1-
Adapter-trimming/Sample-10-replicate-1-trimmed.fastq ../1-Adapter-
trimming/Sample-10-replicate-2-trimmed.fastq &
```

5. Transcript identification using cufflinks

```
nohup /home/deshpan4/RNA-Seq/cufflinks-2.2.1.Linux_x86_64/cufflinks -I 5000 -
-min-intron-length 40 -G /home/deshpan4/RNA-
Seq/Arabidopsis_thaliana/Ensembl/TAIR10/Annotation/Genes/genes.gtf -p 6 -o
Sample-13-cufflinks ../2-Alignment/Sample-13-aligned-
tophat2/accepted_hits.bam &
```

6. Let us take an example of **Sample-7 with Sample-9**. For transcript assembly we will first create **'assembly-7-9.txt'** file.

Add the paths of transcripts.gtf to the **'assembly-7-9.txt'** file:

```
/home/RNA-Seq-analysis/Sample-7/Sample-7-cufflinks/transcripts.gtf
/home/RNA-Seq-analysis/Sample-9/Sample-9-cufflinks/transcripts.gtf
/home/RNA-Seq-analysis/Sample-9-replicate/Sample-9-replicate-
cufflinks/transcripts.gtf
```

Save the text file as **assembly-7-9.txt.**

7. Before running cufflinks add cufflinks to the PATH otherwise it will give an error as 'gtf_to_sam not found on this system'

PATH=$PATH:/home/cufflinks/ <or wherever you've stored gtf_to_sam>

8. Create a folder named 'cuffmerge-7-9'. Run cuffmerge in the following folder:

```
/home/RNA-Seq-analysis/cufflinks-2.2.1.Linux_x86_64/cuffmerge -g /home/RNA-
Seq-analysis/Arabidopsis_thaliana/Ensembl/TAIR10/Annotation/Genes/genes.gtf -
s /home/RNA-Seq-
analysis/Arabidopsis_thaliana/Ensembl/TAIR10/Sequence/WholeGenomeFasta/genome
.fa -p 6 ../assembly-7-9.txt > log.out
```

9. Now create a folder called **cuffdiff-7-9.** Run cuffdiff in the following folder:

```
/home/deshpan4/RNA-Seq/cufflinks-2.2.1.Linux_x86_64/cuffdiff -o
diff_out_quartile_sample_7_sample_9 -b /home/deshpan4/RNA-
Seq/Arabidopsis_thaliana/Ensembl/TAIR10/Sequence/WholeGenomeFasta/genome.fa -
p 6 -L Sample-7,Sample-9 -u ../cuffmerge-7-9/merged_asm/merged.gtf
../../Sample-7/2-Alignment/Sample-7-aligned-tophat2/accepted_hits.bam
../../Sample-9/2-Alignment/Sample-9-aligned-
tophat2/accepted_hits.bam,../../Sample-9-replicate/2-Alignment/Sample-9-
replicate-aligned-tophat2/accepted_hits.bam --library-norm-method quartile --
multi-read-correct &
```

10. Convert BAM to SAM using samtools

```
samtools index /home/RNA-Seq-analysis/Sample-7/2-Alignment/file-1-2-aligned-tophat2/accepted_hits.bam
samtools index /home/RNA-Seq-analysis/Sample-9/2-Alignment/file-1-2-aligned-tophat2/accepted_hits.bam
samtools index /home/RNA-Seq-analysis/Sample-9-replicate/2-Alignment/file-1-2-aligned-tophat2/accepted_hits.bam
samtools view -h /home/RNA-Seq-analysis/Sample-7/2-Alignment/file-1-2-aligned-tophat2/accepted_hits.bam > /home/RNA-Seq-analysis/Sample-7/2-Alignment/file-1-2-aligned-tophat2/accepted_hits.sam
samtools view -h /home/RNA-Seq-analysis/Sample-9/2-Alignment/file-1-2-aligned-tophat2/accepted_hits.bam > /home/RNA-Seq-analysis/Sample-9/2-Alignment/file-1-2-aligned-tophat2/accepted_hits.sam
samtools view -h /home/RNA-Seq-analysis/Sample-9-replicate/2-Alignment/file-1-2-aligned-tophat2/accepted_hits.bam > /home/RNA-Seq-analysis/Sample-9-replicate/2-Alignment/file-1-2-aligned-tophat2/accepted_hits.sam
```

11. Count reads using HTSeq

```
htseq-count Sample-7-aligned.sam /home/RNA-Seq-analysis/Arabidopsis_thaliana/Ensembl/TAIR10/Annotation/Genes/genes.gtf > Sample-7-read-counts.out

htseq-count Sample-9-aligned.sam /home/RNA-Seq-analysis/Arabidopsis_thaliana/Ensembl/TAIR10/Annotation/Genes/genes.gtf > Sample-9-read-counts.out

htseq-count Sample-9-replicate-aligned.sam /home/RNA-Seq-analysis/Arabidopsis_thaliana/Ensembl/TAIR10/Annotation/Genes/genes.gtf > Sample-9-replicate-read-counts.out
```

12. Append header to the 'Sample-7-read-count.out' file

GeneID	Sample-7
AT1G01010	232
AT1G01020	377
AT1G01030	15
AT1G01040	803
AT1G01046	0
AT1G01050	2127
AT1G01060	0
AT1G01070	3364
AT1G01073	0

Similarly, do this for 'Sample-9-read-count.out' and 'Sample-9-replicate-read-count.out' files

13. Open R console, and merge the read counts of the samples using gene identifier

```
>s7<-read.csv("Sample-7-read-count.out",header=T,check.names=F)
>s91<-read.csv("Sample-9-read-count.out",header=T,check.names=F)
```

```
>s92<-read.csv("Sample-9-replicate-read-count.out",header=T,check.names=F)
>s912<-merge(s91,s92,by="GeneID",all.x=T)
>s7s9<-merge(s7,s912,by="GeneID",all.x=T)
```

13. Run DESeq on Sample-7 and Sample-9 (s7s9) sample pairs:

```
>library(DESeq)
```

13.1 For obtaining differentially expressed genes, the sample pairs should be divided into control and test samples. Therefore, Sample-7 is a control sample and Sample-9 is test sample.

```
>condition=factor(c("untreated","treated","treated"))
>cds=newCountDataSet(s7s9,condition)
```

13.2 As first processing step, we will determine effective library size called size factors therefore, we will determine size factors from counts data

```
>norm=estimateSizeFactors(cds)
```

13.3 Next, we will estimate variance or dispersion which is typically square of the coefficient of biological variation

```
>disp=estimateDispersions(norm)
```

For sample comparisons where there are no replicates available for example, Sample-7 and Sample-16, we use blind dispersion method so that condition labels are ignored and variance is estimated

```
>disp=estimateDispersions(norm, method="blind", sharingMode="fit-only")
```

13.4 To calculate differential expression between "untreated" and "treated" conditions, nbinomTest function is called which returns dataframe with p-values, adjusted p-values, fold change, log2fold change and base mean values for each GeneID in the data

```
>res=nbinomTest(disp,"untreated","treated")
```

13.5 To obtain significant p-values, we select those GeneIDs whose adjusted p-values are less than 0.05.

```
>resSig=res[res$padj<0.05,]
>write.csv(resSig,quote=FALSE,row.names=FALSE,"Sample-7-Sample-9-significant-
padj-lessthan-0.05.csv")
```

14. Once significant genes are obtained from DESeq analysis, we will execute edgeR to obtain significantly expressed genes using edgeR script in R:

```
>library(edgeR)
```

14.1 First step of data processing is to filter those GeneIDs whose read counts are non-zeros. Therefore, we will execute a function in which we will identify those IDs which have 0s in both the samples sample-7 and sample-9.

```
>l=apply(counts,1,function(y){all(y==0)})
```

14.2 Second step is to extract the GeneIDs having 0s in both the samples sample-7 and sample-9.

```
>f=counts[l,]
```

14.3 Third step is to match the GeneIDs to row IDs for removal from the dataset

```
>m=match(row.names(f),row.names(counts))
```

14.4 Fourth step is to remove these GeneIDs from the counts data to construct filtered dataset having non-zero read count values in all 3 of them simultaneously. However, zero values in either one of them is allowed.

```
>xx=counts[-m,]
```

14.5 For obtaining differentially expressed genes, the sample pairs should be divided into control (1) and test (2) samples. Therefore, Sample-7 is a control sample (1) and Sample-9 is test sample (2).

```
>group <- factor(c(1,2,2))
>design <- model.matrix(~group)
```

14.6 Once the data is divided into control and test, the read counts are stored as matrix counts containing integer counts using DGEList object

```
>y <- DGEList(counts=xx,group=group)
```

14.7 Once the read counts are stored, calcNormFactors function normalizes RNA composition by scaling factors sets for library sizes which minimize log-fold changes between the samples.

```
>y <- calcNormFactors(y)
```

14.8 Using Cox-Reid (CR) method, dispersion is estimated. It calculates common dispersion for all tags as well as separate dispersions for individual tags using estimateGLMCommonDisp and estimateGLMTagwiseDisp functions

```
>y <- estimateGLMCommonDisp(y,design)
>y <- estimateGLMTagwiseDisp(y,design)
```

14.9 Once dispersion is estimated, generalized linear model likelihood ratio test (glmFit() and glmLRT()) is executed on the data to obtain differentially expressed genes

```
>fit <- glmFit(y,design)
>lrt <- glmLRT(fit,coef=2)
>res <- topTags(lrt,20)
```

res gives list of top 20 differentially expressed genes which gives PValue, false discovery rate (FDR), likelihood ratio (LR) value, log fold-change (logFC) and log counts per million (logCPM) values.

14.10 For sample comparisons where there are no replicates available for example, Sample-7 and Sample-16, we calculate differential expression using exactTest function instead of glmFit.

For samples without any biological replicates, we use common BCV (square-root-dispersion) value. For experiments having human data, this value is set to 0.4, for experiments having genetically identical organisms this value is set to 0.1 and for experiments having technical replicates this value is set to 0.01. Below steps show how to obtain DGE for samples without biological replicates.

```
>l=apply(counts,1,function(y){all(y==0)})
>f=counts[l,]
>m=match(row.names(f),row.names(counts))
>xx=counts[-m,]
>group <- factor(c(1,2))
>design <- model.matrix(~group)
>y <- DGEList(counts=xx,group=group)
>y1 <- calcNormFactors(y)
>bcv <- 0.1
>et1 <- exactTest(y1,dispersion = bcv^2)
>et1new <- cbind(et1$table,p.adjust(et1$table$PValue,method="fdr"))
>colnames(et1new) <- c("logFC","logCPM","PValue","adjsutedPValue")
```

14.11 Save all the results of DEG analysis to "Sample-7-Sample-9-edgeR.csv"

```
>res <- topTags(lrt,36000)
>write.csv(res,quote=FALSE,row.names=FALSE,"Sample-7-Sample-9-edgeR.csv")
```

15. Construct BSgenome package for A.thaliana TAIR10 data

However, there is already BSgenome package availabe in CRAN 'BSgenome.Athaliana.TAIR.TAIR9', but we recommend users to construct TAIR10 version of BSgenome which can be constructed using following steps.

Convert genome FASTA to 2bit

```
faToTwoBit genome.fa athalianaTAIR10.2bit
```

Set the 'seqs_srcdir:' in '*BSgenome.Athaliana.TAIR.TAIR10-seed*' file (which is available in Genome folder) by adding the path to wherever the athalianaTAIR10.2bit file is located.

In R run the following command:

```
>setwd("/path/to/seed/file")
>library(BSgenome)
>forgeBSgenomeDataPkg(BSgenome.Athaliana.TAIR.TAIR10-seed)
```

This command will generate a folder which can be used to install the package on the system.

Next, in command-line, run the following command:

```
R CMD build BSgenome.Athaliana10.TAIR.TAIR10
R CMD INSTALL BSgenome.Athaliana10.TAIR.TAIR10_1.4.2.tar.gz
```

16. Alternative splicing analysis using *spliceR*

```
>library(spliceR)
>library(BSgenome.Athaliana10.TAIR.TAIR10)
>cuffDB <-
>readCufflinks(dir=file.path("path/to/diff_out_quartile_sample_7_sample_8/")
,gtf=file.path("path/to/diff_out_quartile_sample_7_sample_8/merged.gtf"),reb
uild=TRUE,genome="Athaliana10")
>cuffDB_spliceR <- prepareCuff(cuffDB)
>mySpliceRList <- spliceR(
     cuffDB_spliceR,
     compareTo="preTranscript",
     filters=c("expressedGenes","geneOK", "isoOK", "expressedIso",
"isoClass"),
     useProgressBar=T
)
>mySpliceRList <-
spliceRPlot(mySpliceRList,evaluate="nr_transcript_pr_gene")
>mySpliceRList <-
spliceRPlot(mySpliceRList,evaluate="mean_AS_transcript",asType="ESI")
```

17. Merging differentially expressed Gene IDs in more than one sample pairs

In order to merge DE gene IDs, user must first extract At gene IDs from GTF file (*merged.gtf*) merged by Cuffmerge for the sample pair under comparison. For example, if the user would like to extract Gene IDs present in more than 1 sample pair in 5 samples i.e. S1, S2, S3, S4, S5 and the comparison is against the first sample S1 such as (S1 vs S2, S1 vs S3, S1 vs S4 and S1 vs S5), then those genes will be extracted which are expressed in more than one sample pairs. First step is to extract At IDs from GTF file as DGE

table obtained from Cuffdiff (*gene_exp.diff*) file contains gene names instead of gene IDs. To extract the gene IDs, perform the following commands in linux shell:

```
#!/bin/bash
cut -f9 merged.gtf > col-9.txt
cut -d';' -f4 col-9.txt > genename.txt
cut -d';' -f5 col-9.txt > geneid1.txt
sed 's/gene_name //g' genename.txt > genename1.txt
sed 's/"//g' genename1.txt > genename2.txt
sed -i '1s/^/gene/' genename2.txt > genename3.txt
sed 's/oId //g' geneid1.txt > geneid2.txt
sed 's/"//g' geneid2.txt > geneid3.txt
sed -i '1s/^/id/' geneid3.txt > geneid4.txt
paste -d',' genename3.txt geneid4.txt > geneid_genename.txt
rm col-9.txt genename.txt geneid1.txt genename1.txt genename2.txt
geneid2.txt geneid3.txt
```

```
#!/usr/bin/env Rscript
args = commandArgs(trailingOnly=TRUE)
sC<-read.csv(args[1],header=T,check.names=F)
sD<-read.csv(args[2],header=T,check.names=F)
sE<-read.csv(args[3],header=T,check.names=F)
geneidname<-read.csv(args[4],header=T,check.names=F)
a1<-sC[which(sC$q_value <= 0.05),]
a2<-sD[which(sD$padj <= 0.05),]
a3<-sE[which(sE$FDR<= 0.05),]
a1g<-merge(a1,geneidname,by="gene",all.x=T)
write.csv(a1g,quote=FALSE,row.names=FALSE,"Sample-1-Sample-2-significant-
padj-lessthan-0.05-Cuffdiff-results.csv")
write.csv(a2,quote=FALSE,row.names=FALSE,"Sample-1-Sample-2-significant-
padj-lessthan-0.05-DESeq-results.csv")
write.csv(a3,quote=FALSE,row.names=FALSE,"Sample-1-Sample-2-significant-
padj-lessthan-0.05-edgeR-results.csv")
```

Run this R script using following commands:

```
Rscript mergeGeneID_with_Cuffdiff_res.R cuffdiff_file DESeq_file edgeR_file
geneid_genename.txt
```

Merge DGE genes from Cuffdiff (C), DESeq (D) and edgeR (E) in individual sample pairs. For example, to merge C, D, E genes in S1 vs S2 sample pair (Sample-1-2), run the following script.

```
#!/usr/bin/env Rscript
args = commandArgs(trailingOnly=TRUE)
sC<-read.csv(args[1],header=T,check.names=F)
sD<-read.csv(args[2],header=T,check.names=F)
sE<-read.csv(args[3],header=T,check.names=F)
```

```
sCD<-merge(sC,sD,by="id",all.x=T)
sCDE<-merge(sCD,sE,by="id",all.x=T)
write.csv(sCDE,quote=FALSE,row.names=FALSE,"Sample-1-2-common-genes-
Cuffdiff-DESeq-edgeR.csv")
```

Run this R script using following commands:

```
Rscript merge_CDE.R cuffdiff_file DESeq_file edgeR_file
```

In each of the sample pair files, append sample name to each cuffdiff, DESeq and edgeR result files in R. For example, for sample pair S1 vs S2 (Sample-1-2-common-genes-C-D-E.csv)

```
## In R
>df<-read.csv("Sample-1-2-common-genes-C-D-E.csv",header=T,check.names=F)
>e1<-"S1S2"
>df["Sample1_Sample2"]<-e1
>df1<-df[,c("id","Sample1_Sample2")]
>write.csv(df,quote=FALSE,row.names=FALSE,"Sample-1-2-common-genes-C-D-
E.csv")
>write.csv(df1,quote=FALSE,row.names=FALSE,"Sample-1-2-common-genes-C-D-E-
idname.csv")
```

First merge multiple sample pair files:

```
>s12<-read.csv("Sample-1-2-common-genes-C-D-E-
idname.csv",header=T,check.names=F)
>s13<-read.csv("Sample-1-3-common-genes-C-D-E-
idname.csv",header=T,check.names=F)
>s14<-read.csv("Sample-1-4-common-genes-C-D-E-
idname.csv",header=T,check.names=F)
>s15<-read.csv("Sample-1-5-common-genes-C-D-E-
idname.csv",header=T,check.names=F)
>s123<-merge(s12,s13,by="id",all.x=T)
>s1234<-merge(s123,s14,by="id",all.x=T)
>s12345<-merge(s1234,s15,by="id",all.x=T)
>write.csv(s12345,quote=FALSE,row.names=FALSE,"Sample-1-2-3-4-5-common-genes-
C-D-E-idname.csv")
```

Extract common genes present in more than one column from multiple sample pairs.

```
Rscript extract_common_genes.R Sample-1-2-3-4-5-common-genes-C-D-E-idname.csv
```

18. Post-analysis scripts for constructing expression graphs. To generate the expression graphs, perform the following commands in R console to get the results.

First, divide each raw read count value in a row by the maximum value in that row from S7-S8 to S7-S16. Do this for all rows. Next,

To generate expression profiles of common genes from Cuffdiff-DESeq-edgeR overlap.

```
>a<-read.csv("common-genes-Cuffdiff-DESeq-edgeR.csv",header=T,check.names =
F,row.names = 1)
>a1<-t(a)
>matplot(a1, type = c("b"),pch=1:14,col = 1:14, xlab="Sample pairs",
ylab="Relative Expression")
>legend("bottomleft", legend = 1:14, col=1:14, pch=1:14, cex=0.5)
```

To generate expression profiles of specific experimental genes:

```
> df<-read.csv("FLC-gene.csv",header=T,check.names = F,row.names = 1)
> df1<-t(df)
> matplot(df1, type = c("b"),pch=1,col = 1:2, xlab="Sample pairs",
ylab="Relative Expression")
> legend("bottomleft", legend = 1:2, col=1:2, pch=1:2, cex=0.5)
```

Script for plotting Relative Expression values against sample pairs of common genes from Cuffdiff-DESeq-edgeR overlap:

```
>g1<-read.csv("SCMP.csv",header=T)
>dfgi1<-read.csv("C:\\Users\\Sumukh-MSI\\Downloads\\Article writing\\Apical
shoot\\IJBRA\\GO ontology graphs\\S7-to-S16-FPKM-
id.csv",header=T,check.names=F)
>g1dfgi<-merge(g1,dfgi1,by="id",all.x=T)
>g1dfgi1<-g1dfgi[,c(2:11)]
>g1dfgi1_t<-t(g1dfgi1)
>n<-ncol(g1dfgi1_t)
>set3<-numeric(0)
>for (x in 1:n){
    normcol1<-t((g1dfgi1_t[,x])/(max(t(g1dfgi1)[,x])))
    set3<-rbind(set3,normcol1)
}
>colg1<-as.vector(g1dfgi$id)
>set3t<-data.frame(t(set3))
>colnames(set3t)<-colg1
>l<-length(set3t)
>png(filename="SCMP-BP.jpg",height=706,width=1285)
>matplot(set3t, type = c("b"),pch=1:l,col = 1:l, xlab="Sample pairs",
>ylab="Relative Expression")
>legend("bottomleft", legend = colg1, col=1:l, pch=1:l, cex=1.0, ncol=3)
>dev.off()
```

Script for plotting Relative Expression values against sample pairs of flowering genes from “Against S7” sample pairs

```
>f1<-read.csv("Neg Reg of Flower development.csv",header=T,check.names=F)
>p1<-f1[,1]
>o1<-length(p1)
>matplot(t(f1[,c(2:10)]), type = c("b"),pch=1:l,col = 1:l, xlab="Sample
pairs", ylab="Relative Expression")
>legend("bottomleft", legend = p1, col=1:o1, pch=1:o1, cex=0.5, ncol=4)
```

Script for plotting Relative Expression values against sample pairs of experimental genes

```
>f1<-read.csv("FLC.csv",header=T,check.names=F)
>p1<-f1[,1]
>o1<-length(p1)
>matplot(t(f1[,c(2:10)]), type = c("b"),pch=1:l,col = 1:l, xlab="Sample
pairs", ylab="Relative Expression")
>legend("bottomleft", legend = p1, col=1:o1, pch=1:o1, cex=1.0)
```

Script for obtaining most prevalent interactors in the PPI network on linux shell

```
awk -v RS="[:punct:]" '{for(i=1;i<=NF;i++) words[$i]++;}END{for (i in words)
print words[i]" "i}' file > prevalent-interactors.txt
```

Script for performing post-analysis using SpliceR for classification of transcripts obtained from Cufflinks

```
>library(spliceR)
>library(BSgenome.Athaliana10.TAIR.TAIR10)
>cuffDB <- readCufflinks(dir=file.path("C:/Users/Sumukh-
MSI/Downloads/Coventry/RNA-Seq/Apical-shoot-paper/Cuffdiff-Against-
S7/diff_out_quartile_sample_7_sample_8/"),gtf=file.path("C:/Users/Sumukh-
MSI/Downloads/Coventry/RNA-Seq/Apical-shoot-paper/Cuffdiff-Against-
S7/diff_out_quartile_sample_7_sample_8/merged.gtf"),rebuild=TRUE,genome="Atha
liana10")
>cuffDB_spliceR <- prepareCuff(cuffDB)
mySpliceRList <- spliceR(
     cuffDB_spliceR,
     compareTo="preTranscript",
     filters=c("expressedGenes","geneOK", "isoOK", "expressedIso",
"isoClass"),
     useProgressBar=T
)

#Plot the average number of transcripts pr gene
>mySpliceRList <- spliceRPlot(mySpliceRList,
      evaluate="nr_transcript_pr_gene")

#Plot the average number of Exon skipping/inclucion evens pr gene
```

```
>mySpliceRList <- spliceRPlot(mySpliceRList, evaluate="mean_AS_transcript",
      asType="ESI")
```