# Exercise Set 2: RNA-seq quantification continued

## Bulk RNA-seq: transcripts quantification

In this week exercise, it will be continued to explore methods for buld RNA-seq data analysis, especially Kallisto and compare the results from this tool with our alignment-based RNA-seq quantifcation from Exercise set 1 (Hisat2)

### Getting started

For this exercise set, it will be continued to work in the working directory of exercise set 1.

```
setwd("student_data/BBT.BI.203_2022/students/leh/ex1")
```

## Quantification with an alignment free approach

### Index generation

kallisto quantifies read files directly without the need for read alignment, but it does perform a procedure called pseudoalignment. Pseudoalignment requires processing a transcriptome file to create a "transcriptome index". To begin, first change directories to where the test files distributed with the kallisto executable are located.

#### Tasks

```
cd kallisto/index
kallisto index -i gencode.v32.transcripts.idx /student_data/BBT.BI.203_2022/references/ex1/fasta/gencod
```

### Counting transcript expression

Now we can process to quantify the transcripts we just indexed.

### Tasks1

**Task 1** Perform the quantification for each sample and store the results in the kallisto/output

```
kallisto quant -i kallisto/index/gencode.v32.transcripts.idx
               -o kallisto/output/BPH_659
               -b 100 BPH_659.chrX_R1.fq.gz BPH_659.chrX_R2.fq.gz

kallisto quant -i kallisto/index/gencode.v32.transcripts.idx
                -o kallisto/output/BPH_665
```

```
                -b 100 BPH_665.chrX_R1.fq.gz BPH_665.chrX_R2.fq.gz

kallisto quant -i kallisto/index/gencode.v32.transcripts.idx
                -o kallisto/output/BPH_701
                -b 100 BPH_701.chrX_R1.fq.gz BPH_701.chrX_R2.fq.gz

kallisto quant -i kallisto/index/gencode.v32.transcripts.idx
                -o kallisto/output/PC_13943
                -b 100 PC_13943.chrX_R1.fq.gz PC_13943.chrX_R2.fq.gz

kallisto quant -i kallisto/index/gencode.v32.transcripts.idx
                -o kallisto/output/PC_15194
                -b 100 PC_15194.chrX_R1.fq.gz PC_15194.chrX_R2.fq.gz

kallisto quant -i kallisto/index/gencode.v32.transcripts.idx
                -o kallisto/output/PC_19403
                -b 100 PC_19403.chrX_R1.fq.gz PC_19403.chrX_R2.fq.gz
```

**Task 2** Generate the pseudoalignment files in BAM format for the trascriptome and for the genome.

```
### transcriptome
kallisto quant -i gencode.v32.transcripts.idx
-o kallisto/output/BPH_659 --pseudobam
--gtf gencode.v32.annotation.chrX.for.kallisto.gtf
--bootstrap-samples 30 BPH_659.chrX_R1.fq.gz BPH_659.chrX_R2.fq.gz

kallisto quant -i gencode.v32.transcripts.idx
-o kallisto/output/BPH_665 --pseudobam
--gtf gencode.v32.annotation.chrX.for.kallisto.gtf
--bootstrap-samples 30 BPH_665.chrX_R1.fq.gz BPH_665.chrX_R2.fq.gz

kallisto quant -i gencode.v32.transcripts.idx
-o kallisto/output/BPH_701 --pseudobam
--gtf gencode.v32.annotation.chrX.for.kallisto.gtf
--bootstrap-samples 30 BPH_701.chrX_R1.fq.gz BPH_701.chrX_R2.fq.gz

kallisto quant -i gencode.v32.transcripts.idx
-o kallisto/output/PC_13943 --pseudobam
--gtf gencode.v32.annotation.chrX.for.kallisto.gtf
--bootstrap-samples 30 PC_13943.chrX_R1.fq.gz PC_13943.chrX_R2.fq.gz

kallisto quant -i gencode.v32.transcripts.idx
-o kallisto/output/PC_15194 --pseudobam
--gtf gencode.v32.annotation.chrX.for.kallisto.gtf
--bootstrap-samples 30 PC_15194.chrX_R1.fq.gz PC_15194.chrX_R2.fq.gz

kallisto quant -i gencode.v32.transcripts.idx
-o kallisto/output/PC_19403 --pseudobam
--gtf gencode.v32.annotation.chrX.for.kallisto.gtf
--bootstrap-samples 30 PC_19403.chrX_R1.fq.gz PC_19403.chrX_R2.fq.gz

### Rewrite the file name before running genome
```

```
### genome
kallisto quant -i gencode.v32.transcripts.idx -o output/BPH_659
--genomebam --gtf gencode.v32.annotation.chrX.for.kallisto.gtf
--bootstrap-samples 30 BPH_659.chrX_R1.fq.gz BPH_659.chrX_R2.fq.gz

kallisto quant -i gencode.v32.transcripts.idx -o output/BPH_665
--genomebam --gtf gencode.v32.annotation.chrX.for.kallisto.gtf
--bootstrap-samples 30 BPH_665.chrX_R1.fq.gz BPH_665.chrX_R2.fq.gz

kallisto quant -i kallisto/index/gencode.v32.transcripts.idx -o output/BPH_701
--genomebam --gtf gencode.v32.annotation.chrX.for.kallisto.gtf
--bootstrap-samples 30 BPH_701.chrX_R1.fq.gz BPH_701.chrX_R2.fq.gz

kallisto quant -i kallisto/index/gencode.v32.transcripts.idx -o output/PC_13943
--genomebam --gtf gencode.v32.annotation.chrX.for.kallisto.gtf
--bootstrap-samples 30 PC_13943.chrX_R1.fq.gz PC_13943.chrX_R2.fq.gz

kallisto quant -i kallisto/index/gencode.v32.transcripts.idx -o output/PC_15194
--genomebam --gtf gencode.v32.annotation.chrX.for.kallisto.gtf
--bootstrap-samples 30 PC_15194.chrX_R1.fq.gz PC_15194.chrX_R2.fq.gz

kallisto quant -i index/gencode.v32.transcripts.idx -o kallisto/output/PC_19403
--genomebam --gtf gencode.v32.annotation.chrX.for.kallisto.gtf
--bootstrap-samples 30 PC_19403.chrX_R1.fq.gz PC_19403.chrX_R2.fq.gz
```

**Task 3** inspect the pseudoalignments with IGV.

### Questions

**Question 1** What do the different columns of the output file contain?

The output annotation files include 5 columns following: target_id, length, eff_length, est_counts, and tpm, respectively. They are known as the results after running the quantification algorithm.

Target_id: indicated the list of transcript.

Length: is known as the length of transcrpt.

Eff_length = effective length should generally be the length minus the average fragment length + 1. Effective length is used in FPKM calculated and be dependent on the length of transcript and the average fragment length.

Est_counts = estimated counts: Estimated counts were used rather than transcripts per million (TPM) because the latter is based on both the assignment of ambiguous reads and the estimation of effective lengths of transcripts.

Tpm = transcripts per million: is a within sample normalization method which takes into account sequencing depth and length biases. However, we will first normalize for length bias and then for sequencing depth bias in kallisto command.

**Question 2** run_info.json is a json file containing information about the run

```
##          Analyzed transcripts Pseudoalignation Unique hits
## BPH_659               1434722          1341798      290888
## BPH_665               1552486          1473638      306998
```

```
## BPH_701                1475037          1404783       316409
## PC_13943               1356064          1301948       304933
## PC_15194               1215200          1162713       268752
## PC_19403               1406716          1344522       292492
```

**Question 3** What dose the genome psedoalignment look like in IGV? It shows the mapping genes by .bamcoverage, the both side mapping in .bam junctions, and mapping/deleting/inserting/mutations through .bam.

Howe dose it different from the alignments we generated with Hisat2 in ex1? From th mapping genens, in general, both Hisat2 and Kallisto results show the same mapping genes through .bamcoverage. However, in Hisat2, the mapping sizes shown in .bam are longer in Kallisto, I think because Hisat2 makes the alignments first and then mapping to the reference genome, therefore the mapping sizes in .bam of Hisat2 are oftern longer than Kallisto. However, Kallisto has the longer reaction of mapping than Hisat2 (We could observe it in .bam files of both methods in IGV).

**Question 4** What dose the transcript psedoalignment look like in IGV? Kallisto shows more mapping in transcripts than Hisat2.

## Data matrix generation

### Tasks

Build a data matrix

```
# Load all abudance.tsv files into an R session
setwd("/student_data/BBT.BI.203_2022/students/leh/ex1/kallisto")
BPH_665_df = read.table("/student_data/BBT.BI.203_2022/students/leh/ex1/kallisto/output/BPH_665/abundanc
                        sep = '\t',
                        header = TRUE)
BPH_659_df = read.table("/student_data/BBT.BI.203_2022/students/leh/ex1/kallisto/output/BPH_659/abundanc
                        sep = '\t',
                        header = TRUE)
BPH_701_df = read.table("/student_data/BBT.BI.203_2022/students/leh/ex1/kallisto/output/BPH_701/abundanc
                        sep = '\t',
                        header = TRUE)
PC_13943_df = read.table("/student_data/BBT.BI.203_2022/students/leh/ex1/kallisto/output/PC_13943/abunda
                         sep = '\t',
                         header = TRUE)
PC_15194_df = read.table("/student_data/BBT.BI.203_2022/students/leh/ex1/kallisto/output/PC_15194/abunda
                         sep = '\t',
                         header = TRUE)
PC_19403_df = read.table("/student_data/BBT.BI.203_2022/students/leh/ex1/kallisto/output/PC_19403/abunda
                         sep = '\t',
                         header = TRUE)

kallisto_list <- list(BPH_659_df, BPH_665_df, BPH_701_df, PC_13943_df, PC_15194_df, PC_19403_df)

#Prepare two output matrices est_counts_matrix and tpm_matrix
transcript_identifiers = BPH_665_df$target_id
col_names = c("BPH_659", "BPH_665", "BPH_701", "PC_13943","PC_15194","PC_19403")
est_counts_matrix <- matrix(ncol = 6,
                            nrow = length(transcript_identifiers),
```

```
                              dimnames = list(transcript_identifiers,
                                              col_names))
tpm_matrix <- matrix(ncol = 6, nrow = length(transcript_identifiers),
                     dimnames = list(transcript_identifiers,
                                     col_names))

#Iterate over the kallisto_list and populate the two matrices
for(i in 1 : length(kallisto_list)){
  est_counts_matrix[,i] = kallisto_list[[i]]$est_counts

}
for(i in 1 : length(kallisto_list)){
  tpm_matrix[,i] = kallisto_list[[i]]$tpm

}

#Write the two matrices to two files kallisto/estimated_counts.csv and kallisto/tpm.csv.
write.table(est_counts_matrix,
            file="kallisto/estimated_counts.csv",
            row.names = T,
            col.names = T)

write.csv(tpm_matrix, file = "kallisto/tpm.csv")
```

# Comparison: featureCounts and Kallisto

## Tasks

### Task 1

To generate a gene level estimate for each gene based on the transcripts of that gene.

```
# begin analysis in R
# import the GFF annotation file
library(rtracklayer)
ann <- import("/student_data/BBT.BI.203_2022/references/ex1/gff/gencode.v32.chrX.gff3.gz")

# build a mapping table between transcript id and gene id
# this data frame tells you which transcripts belong to which gene
df <- data.frame(transcript_id = unlist(ann$transcript_id),
                 gene = unlist(ann$Parent),
                 stringsAsFactors=FALSE)
df
```

To use the merge function to mergit df with the estimated transcript abundances from Kallisto (est_counts_matrix.csv). The new data include each row of Kallisto counts matrix and which gene that transcript belongs to.

```
# Input
est_counts_matrix
df

#merge function for estimated count from kallisto
```

```
est_count_dataframe = as.data.frame(est_counts_matrix)
row_names = row.names(est_count_dataframe)
est_count_dataframe$transcript_id <- row.names(est_count_dataframe)
merge_data <- merge(df, est_count_dataframe, by= c("transcript_id"), all = TRUE)

#use the aggregate function to sum up the counts of the transcripts belonging to the same gene
gene_level_counts = aggregate(merge_data[,3:8], by = list(merge_data$gene), FUN=sum)
```

To use the aggregate function to sum up the counts of the transcript belongings to the same gene, results in gene-level counts.

```
#use the aggregate function to sum up the counts of genes from Kallisto
gene_level_counts = aggregate(merge_data[,3:8], by = list(merge_data$gene), FUN=sum)

## Loading the data from featureCount in week 1
featurecounts_data_transcript = as.data.frame(
  read.table("/student_data/BBT.BI.203_2022/files/quantification_transcripts_multioverlap.bed",
             header = TRUE,
             sep="\t",
             stringsAsFactors=FALSE))

featurecount_data_genes = as.data.frame(
  read.table("/student_data/BBT.BI.203_2022/students/leh/ex1/histat2/quantification_genes.bed",
             header = TRUE,
             sep="\t",
             stringsAsFactors=FALSE))

## use the aggregate function to sum up the counts of genes from featureCounts
featurecounts_genes_aggregate = aggregate(featurecount_data_genes[,7:12],
                                          by = list(featurecount_data_genes$Geneid),
                                          FUN=sum)
```

To compute the difference in estimated gene counts between featureCounts and Kallisto for each samples and each gene, in here the ordering of gene names between two data is same.

Coding for making the histograme of the difference of genes by two methods

```
featurecounts_genes_aggregate

### To create data
difference_gene = data.frame(gene_level_counts$Group.1,
                             featurecounts_genes_aggregate[,2:7] - gene_level_counts[,2:7])

difference_gene_median = data.frame(difference_gene$gene_level_counts.Group.1,
                                    apply(difference_gene, 1, median))

differnce_gene_sorted = difference_gene_median[order(difference_gene_median$apply.difference_gene..1..m
                                                     decreasing = TRUE),]


# Drawing histogram
par(mfrow = c(3,3))
hist(difference_gene$X.student_data.BBT.BI.203_2022.students.leh.exercise_1.histat2.alignments.BPH_659.
     main = "BPH 659",
     xlab = "Difference of gene counts")
hist(difference_gene$X.student_data.BBT.BI.203_2022.students.leh.exercise_1.histat2.alignments.BPH_665.
```

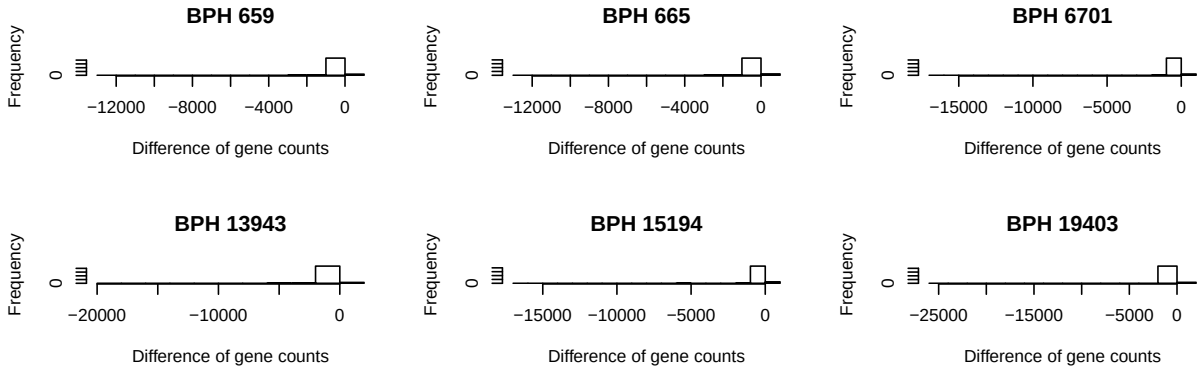


Figure 1: Analysis of expression level characterization

```
     main = "BPH 665",
     xlab = "Difference of gene counts")
hist(difference_gene$X.student_data.BBT.BI.203_2022.students.leh.exercise_1.histat2.alignments.BPH_701.
     main = "BPH 6701",
     xlab = "Difference of gene counts")
hist(difference_gene$X.student_data.BBT.BI.203_2022.students.leh.exercise_1.histat2.alignments.PC_13943
     main = "BPH 13943",
     xlab = "Difference of gene counts")
hist(difference_gene$X.student_data.BBT.BI.203_2022.students.leh.exercise_1.histat2.alignments.PC_15194
     main = "BPH 15194",
     xlab = "Difference of gene counts")
hist(difference_gene$X.student_data.BBT.BI.203_2022.students.leh.exercise_1.histat2.alignments.PC_19403
     main = "BPH 19403",
     xlab = "Difference of gene counts")
```

### Task 2

Plot a histogram of these differences between 6 samples in 2 methods of RNA analysis at transcription level.

```
### To create the difference at transcript level
difference_transcript = data.frame(merge_data$transcript_id, featurecounts_data_transcript[,7:12]-merge_
b_transcript = data.frame(difference_transcript$merge_data.transcript_id, apply(difference_transcript, 
b_transcript_sorted = b_transcript[order(b_transcript$apply.difference_transcript..1..median., decreasi

### Drawing histogram
par(mfrow = c(3,3))
hist(difference_transcript$X.data.projects.taavitsa.BTK6004_2020_ex1.hisat2.alignment.BPH_659.chrX_sorte
     main = "BPH 659",
     xlab = "Difference of transcript counts")
hist(difference_transcript$X.data.projects.taavitsa.BTK6004_2020_ex1.hisat2.alignment.BPH_665.chrX_sorte
```

```
     main = "BPH 665",
     xlab = "Difference of transcript counts")
hist(difference_transcript$X.data.projects.taavitsa.BTK6004_2020_ex1.hisat2.alignment.BPH_701.chrX_sorte
     main = "BPH 701",
     xlab = "Difference of transcript counts")
hist(difference_transcript$X.data.projects.taavitsa.BTK6004_2020_ex1.hisat2.alignment.PC_13943.chrX_sor
     main = "BPH 13943",
     xlab = "Difference of transcript counts")
hist(difference_transcript$X.data.projects.taavitsa.BTK6004_2020_ex1.hisat2.alignment.PC_15194.chrX_sor
     main = "BPH 15194",
     xlab = "Difference of transcript counts")
hist(difference_transcript$X.data.projects.taavitsa.BTK6004_2020_ex1.hisat2.alignment.PC_19403.chrX_sor
     main = "BPH 19403",
     xlab = "Difference of transcript counts")
```

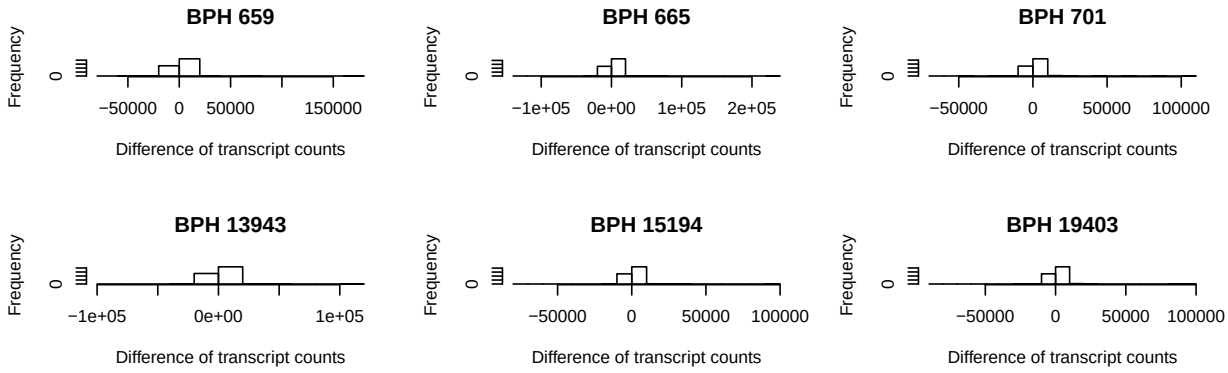


Figure 2: Difference at transcription levels between two methods in each sample

### Task 3

Select the top 50 transcripts with biggest difference in the count between featureCounts and Kallist.

### Task 4

To compute transcript per million (TPM) from the alignment-based transcripts counts from featureCounts.

```
featurecounts_data_transcript
tpm_featurecounts = data.frame(featurecounts_data_transcript$Geneid,
                               featurecounts_data_transcript[, 7:12]/1000000)
```

## Question

### Question 1

Which gene has the biggest median difference across samples in counts? HTATSF1 (ENSG00000102241.12) Median difference of RPL10 in counts using the two methods -9.001700e+00

### Question 2

What is the median difference in the counts of transcripts and genes using the two methods? In general, the median differences in the counts of genes using featureCounts are lower than kallisto. However, at transcrip counts, the median differences in featureCounts have a little higher than kallisto.

### Question 3

I would like to choose kallisto to perform transcript quantification because: mappers will save time compared to aligners. Kallisto is not really mapper in the strict sense of the pseudoaligners and I guess speed and accuracy are the major drivers for kallisto. It could be for instance much better suited for isoform quantification.

### Question 4

### Question 5

Are the difference between TPM measurements from featureCounts and Kallisto similar to the differences observed between raw counts using the two methods?

-1.pdf

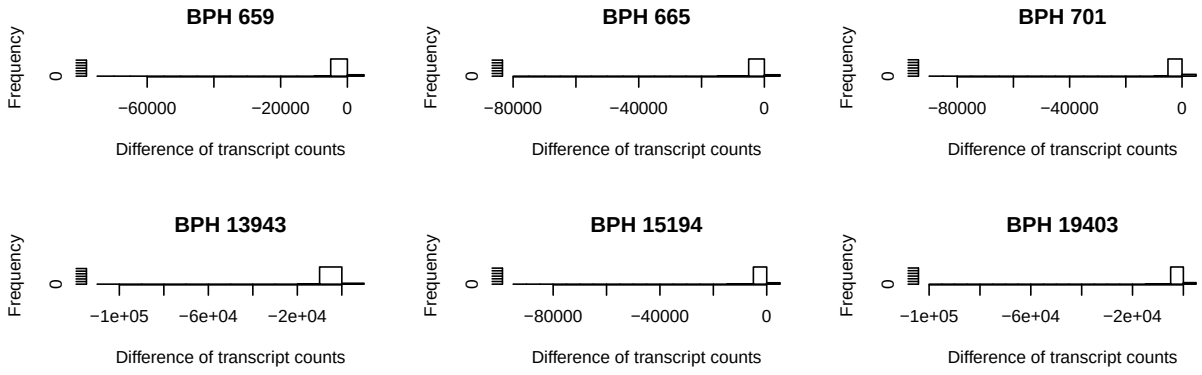


Figure 3: Analysis of tpm

The observation of the differences of tpm looks similar the differences of genes in a comparision of the two methods.