R Notebook

```r
library(biomaRt)
library(edgeR)
```

```
## Warning: package 'edgeR' was built under R version 3.5.2
```

```
## Loading required package: limma
```

```r
library(iterpc)
library(DESeq2)
```

```
## Warning: package 'DESeq2' was built under R version 3.5.2
```

```
## Loading required package: S4Vectors
```

```
## Loading required package: stats4
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
## The following object is masked from 'package:limma':
##
##     plotMA
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind,
##     colMeans, colnames, colSums, dirname, do.call, duplicated,
##     eval, evalq, Filter, Find, get, grep, grepl, intersect,
##     is.unsorted, lapply, lengths, Map, mapply, match, mget, order,
##     paste, pmax, pmax.int, pmin, pmin.int, Position, rank, rbind,
##     Reduce, rowMeans, rownames, rowSums, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which, which.max,
##     which.min
```

```
##
## Attaching package: 'S4Vectors'
```

```
## The following object is masked from 'package:base':
##
##     expand.grid
```

```
## Loading required package: IRanges
```

```
## Loading required package: GenomicRanges
## Loading required package: GenomeInfoDb

## Warning: package 'GenomeInfoDb' was built under R version 3.5.2

## Loading required package: SummarizedExperiment

## Loading required package: Biobase

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

## Loading required package: DelayedArray

## Loading required package: matrixStats

##
## Attaching package: 'matrixStats'

## The following objects are masked from 'package:Biobase':
##
##     anyMissing, rowMedians

## Loading required package: BiocParallel

## Warning: package 'BiocParallel' was built under R version 3.5.2

##
## Attaching package: 'DelayedArray'

## The following objects are masked from 'package:matrixStats':
##
##     colMaxs, colMins, colRanges, rowMaxs, rowMins, rowRanges

## The following objects are masked from 'package:base':
##
##     aperm, apply
```

```r
library(GenomicFeatures)
```

```
## Warning: package 'GenomicFeatures' was built under R version 3.5.2

## Loading required package: AnnotationDbi
```

```r
library (EDASeq)
```

```
## Warning: package 'EDASeq' was built under R version 3.5.2

## Loading required package: ShortRead

## Loading required package: Biostrings

## Warning: package 'Biostrings' was built under R version 3.5.2

## Loading required package: XVector

##
## Attaching package: 'Biostrings'

## The following object is masked from 'package:DelayedArray':
##
##     type
## The following object is masked from 'package:base':
##
##     strsplit
## Loading required package: Rsamtools
## Warning: package 'Rsamtools' was built under R version 3.5.2
## Loading required package: GenomicAlignments
## Warning: package 'GenomicAlignments' was built under R version 3.5.2
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------------------------------- tidyverse 1.2.
## v ggplot2 3.1.0       v purrr   0.3.2
## v tibble  2.0.1       v dplyr   0.8.0.1
## v tidyr   0.8.3       v stringr 1.4.0
## v readr   1.3.1       v forcats 0.4.0
## Warning: package 'tibble' was built under R version 3.5.2
## Warning: package 'tidyr' was built under R version 3.5.2
## Warning: package 'dplyr' was built under R version 3.5.2
## Warning: package 'stringr' was built under R version 3.5.2
## Warning: package 'forcats' was built under R version 3.5.2
## -- Conflicts ------------------------------------------------------------------ tidyverse_conflicts(
## x dplyr::collapse()   masks Biostrings::collapse(), IRanges::collapse()
## x dplyr::combine()    masks Biobase::combine(), BiocGenerics::combine()
## x purrr::compact()    masks XVector::compact()
## x purrr::compose()    masks ShortRead::compose()
## x dplyr::count()      masks matrixStats::count()
## x dplyr::desc()       masks IRanges::desc()
## x tidyr::expand()     masks S4Vectors::expand()
## x dplyr::filter()     masks stats::filter()
## x dplyr::first()      masks GenomicAlignments::first(), S4Vectors::first()
## x dplyr::id()         masks ShortRead::id()
## x dplyr::lag()        masks stats::lag()
## x dplyr::last()       masks GenomicAlignments::last()
## x ggplot2::Position() masks BiocGenerics::Position(), base::Position()
## x purrr::reduce()     masks GenomicRanges::reduce(), IRanges::reduce()
## x dplyr::rename()     masks S4Vectors::rename()
## x dplyr::select()     masks AnnotationDbi::select(), biomaRt::select()
## x purrr::simplify()   masks DelayedArray::simplify()
## x dplyr::slice()      masks XVector::slice(), IRanges::slice()
## x tibble::view()      masks ShortRead::view()
```

```
library(dplyr)

#import data
mouse_count_matrix <- read.delim("~/Downloads/mouse_count_matrix.txt", header = TRUE, sep = "\t")
#save data into a new dataframe new_data1
new_data1 <- mouse_count_matrix
#Since the class of gene column is factor, convert to character and save it in gene_exp variable
gene_exp <- as.character(new_data1$gene)
```

```
#use getGeneLengthAndGCContent function from EDASeq package to get the length and gc content of the res
gene_length_gc <- getGeneLengthAndGCContent(gene_exp, "mmusculus_gene_ensembl")
```

```
## Connecting to BioMart ...

## Downloading sequences ...

## This may take a few minutes ...
```

```
#add the length column in the new_data1 dataframe
new_data1$length <- gene_length_gc[1 : nrow(gene_length_gc)]

#since the length is in bp, converting to kb
new_data1$length <- new_data1$length/1000
head(new_data1)
```

```
##                   gene SRX026633 SRX026632 SRX026631 SRX026630 length
## 1 ENSMUSG00000000001       842       765       437       221  3.262
## 2 ENSMUSG00000000003         0         0         0         0  0.902
## 3 ENSMUSG00000000028        42        60        10        17  3.506
## 4 ENSMUSG00000000031         0         0         0         0  2.460
## 5 ENSMUSG00000000037         0         0         0         0  6.079
## 6 ENSMUSG00000000049         1         0         0         1  1.594
```

RPKM

```
#create a new dataframe RPKM(reads per kilobase million)
RPKM <- as.data.frame(cbind(new_data1$gene, new_data1$SRX026633, new_data1$SRX026632, new_data1$SRX0266

colnames(RPKM) <- c("Gene", "SRX026633", "SRX026632","SRX026631", "SRX026630")
len <- ncol(new_data1)
length_col <- new_data1$length

for (i in 2 : len - 1)
{
  #normalise for read depth
  #calculate the total number of reads from each column and divide the read counts for each gene with
  scaling <- sum(as.numeric(new_data1[ , i]), na.rm = TRUE) / 10 ^ 6
  #normalise for gene length
  #Divide the recently normalised data with gene length
  RPKM[ , i] <- (as.numeric(new_data1[ , i])) / (as.numeric(length_col) * scaling)
  RPKM[ , 1] <- new_data1$gene
}
#RPKM and FPKM are closely related terms. RPKM is used for single-end RNA sequence and FPKM is used for
head(RPKM)
```

```
##                   Gene   SRX026633  SRX026632  SRX026631  SRX026630
## 1 ENSMUSG00000000001 173.7119854 119.851437 164.511999 96.7737442
## 2 ENSMUSG00000000003   0.0000000   0.000000   0.000000  0.0000000
## 3 ENSMUSG00000000028   8.0619301   8.745912   3.502581  6.9260598
## 4 ENSMUSG00000000031   0.0000000   0.000000   0.000000  0.0000000
## 5 ENSMUSG00000000037   0.0000000   0.000000   0.000000  0.0000000
## 6 ENSMUSG00000000049   0.4221952   0.000000   0.000000  0.8961091
```

TPM

```
#create a new dataframe TPM(transcripts per million)
TPM <- as.data.frame(cbind(new_data1$gene, new_data1$SRX026633, new_data1$SRX026632, new_data1$SRX02663
```

```r
colnames(TPM) <- c("Gene", "SRX026633", "SRX026632","SRX026631", "SRX026630")

for (i in 2 : len - 1)
{
  #normalize for each gene length
  #divide each count by length
  norm_gene <- as.numeric(new_data1[ , i]) / as.numeric(length_col)
  #normalize for sequence depth
  #add the read counts of already normalized data and divide by 10 ^ 6
  total_reads_scaling <-  sum(as.numeric(norm_gene), na.rm = TRUE) / 10 ^ 6
  #divide the read counts with the scaling
  TPM[ , i] <- as.numeric(norm_gene) / total_reads_scaling
  TPM[ , 1] <- new_data1$gene
}
head(TPM)
```

```
##                 Gene  SRX026633 SRX026632 SRX026631  SRX026630
## 1 ENSMUSG00000000001 545.336654 373.54521 584.82506 333.521262
## 2 ENSMUSG00000000003   0.000000   0.00000   0.00000   0.000000
## 3 ENSMUSG00000000028  25.308939  27.25869  12.45135  23.869989
## 4 ENSMUSG00000000031   0.000000   0.00000   0.00000   0.000000
## 5 ENSMUSG00000000037   0.000000   0.00000   0.00000   0.000000
## 6 ENSMUSG00000000049   1.325404   0.00000   0.00000   3.088353
```

```r
#Difference between FPKM and TPM
#both are correct for biases in gene length and sequencing depth. However, the sum of the normalised re
#In the case of TPM we get the same value from each column. Numbers can tell what proportion of reads a
```

TMM

```r
##TMM(Trimmed Mean of M-values)
#assumes that most genes are not differentially expressed
TMM <- calcNormFactors(new_data1[ , 2: 5], method = "TMM")
TMM
```

```
## [1] 0.9809033 0.9927119 1.0006430 1.0262931
```

CPM

```r
###CPM(Counts per million)
#descriptive measure for the expression level of a gene
CPM <- cpm(new_data1[ , 2 : 5], lib.size = NULL, log = FALSE)
head(CPM)
```

```
##        SRX026633 SRX026632 SRX026631  SRX026630
## [1,] 566.6484962 390.95539 536.63814 315.675954
## [2,]   0.0000000   0.00000   0.00000   0.000000
## [3,]  28.2651269  30.66317  12.28005  24.282766
## [4,]   0.0000000   0.00000   0.00000   0.000000
## [5,]   0.0000000   0.00000   0.00000   0.000000
## [6,]   0.6729792   0.00000   0.00000   1.428398
```