```
##########################################################################
# Mapping and counting RNA-seq reads with Rsubread R Bioconductor package #
##########################################################################

#####
# Toy example
#####
# small sample of reads (56,168 read-pairs) from a single replicate
# sequencing: strand specific, paired-end, Illumina, 100-100nts
# small chromosome fragment (1-839990 nt part of the 3L chromosome of Drosophila simulans contains 96 genes)

#####
# Mapping
#####
# Call Rsubread library
library(Rsubread)

# Indexing reference genome
buildindex(basename="Genome_sequence_ToyExample_index", reference="../Data/Genome_sequence_ToyExample.fa", memory=1500)
# p7 at Rsubread.pdf: An index needs to be built before read mapping can be performed. This function creates a hash
# table for the reference genome, which can then be used by Subread and Subjunc aligners for read
# alignment.
# Highly repetitive subreads (or uninformative subreads) are excluded from the hash table so as to
# reduce mapping ambiguity.

#####?
# Check how many uninformative subreads were found!
# Open Genome_sequence_ToyExample.fa fasta file with "less" unix command in another terminal.
# Can you see potential uninformative regions in the genome?
#####?

# Mapping reads
subjunc(index="Genome_sequence_ToyExample_index",
readfile1="../Data/Trimmed_reads_ToyExample_1.fq",
readfile2="../Data/Trimmed_reads_ToyExample_2.fq",
input_format="FASTQ",
output_format="BAM", output_file="Mapped_reads_ToyExample.BAM",
nthreads=1, phredOffset=64, unique=TRUE,
minFragLength=50, maxFragLength=10000, PE_orientation="fr")
# p2 at Rsubread.pdf: Subjunc perform global alignments. The seed-and-vote paradigm
# enables efficient and accurate alignments to be carried out."
## phredOffset: sanger 33; illumina 64
## PE_orientation: character string giving the orientation of the two reads from the same pair. It
# has three possible values including fr, ff and rf. Letter f denotes the forward
# strand and letter r the reverse strand. fr by default (ie. the first read in the pair
# is on the forward strand and the second read on the reverse strand).

#####?
# Copy the "Summary" here!
#####?

```

```r
#####
# Counting
#####
Counts_ToyExample=featureCounts(files="Mapped_reads_ToyExample.BAM", annot.ext="../Data/Genome_annotation_ToyExample.gtf", isGTFAnnotationFile=T, GTF.featureType="exon", GTF.attrType="gene_id", useMetaFeatures=T, isPairedEnd=T, requireBothEndsMapped=T,   checkFragLength=F, nthreads=1, strandSpecific=2, reportReads=T)
# p12 at Rsubread.pdf: This function assigns mapped sequencing reads to genomic features
## GTF.featureType: a character string giving the feature type used to select rows in the
# GTF annotation which will be used for read summarization. exon by default.
## GTF.attrType: a character string giving the attribute type in the GTF annotation which will be
# used to group features (eg. exons) into meta-features (eg. genes). gene_id by
# default.
## useMetaFeatures: logical indicating whether the read summarization should be performed at the
# feature level (eg. exons) or meta-feature level (eg genes). If TRUE, features in
# the annotation (each row is a feature) will be grouped into meta-features using
# their values using the “gene_id" attribute in the GTF-format annotation file, and reads will assiged
# to the meta-features instead of the features.
## requireBothEndsMapped: logical indicating if both ends from the same fragment are required to be
# successfully aligned before the fragment can be assigned to a feature or metafeature.
## checkFragLength: logical indicating if the two ends from the same fragment are required to satisify
# the fragment length criteria before the fragment can be assigned to a feature or
# meta-feature. The fragment length criteria are specified via minFragLength and maxFragLength.
## strandSpecific: integer indicating if strand-specific read counting should be performed. It has
# three possible values: 0 (unstranded), 1 (stranded) and 2 (reversely stranded).
## reportReads: logical indicating if read counting result for each read/fragment is saved to a
# file. If TRUE, read counting results for reads/fragments will be saved to a tab-
# delimited file that contains four columns including name of read/fragment, sta-
# tus(assigned or the reason if not assigned), name of target feature/meta-feature
# and number of hits if the read/fragment is counted multiple times. Name of the
# file is the same as name of the input read file except a suffix ‘.featureCounts’ is
# added. Multiple files will be generated if there is more than one input read file.

#####?
# What pecentage of the reads were counted in total?
#####?

# p16 at Rsubread.pdf: Description of featureCounts variable

# Names of objects of the featureCounts variable
```

```
names(Counts_ToyExample)

# Counts of the first genes
head(Counts_ToyExample$counts)

# Histogram of the counts
hist(Counts_ToyExample$counts)
hist(log10(Counts_ToyExample$counts))

# Write the count table to a file
write.table(Counts_ToyExample$counts, "Counts_ToyExample.tsv", quote=F, sep="\t")


###################################################################################
# Differential expression analysis of real data with edgeR R Bioconductor package #
###################################################################################
library("edgeR")

######
# Preprocessing
######

# Read file with read counts
Counts=read.table(file="../Data/Dsim_count_table.tsv", header=T, row.names=1)
head(Counts)
# Number of genes
dim(Counts)

# Keep those genes that were expressed in at a reasonable level (25 pairs) in all samples
KeptCounts=Counts[rowSums(Counts>=25)==6, ]
# Number of kept genes
dim(KeptCounts)

#########################
# Pair wise DE analysis #
#########################
# an example

## "Treatment" groups
Group_PW=factor(c("C15","C15","C15","C23","C23","C23"))
# Tell R that a variable is nominal by making it a factor. The factor stores the nominal values as a vector of integers in the range [ 1... k ] (where k is the number of unique values in the nominal variable), and an internal vector of character strings (the original values) mapped to these integers.

## Differential Expression list
DE_list_PW=DGEList(KeptCounts, group=Group_PW)
## DGElist data class
# edgeR stores data in a simple list-based data object called a DGEList. This type of object is
# easy to use because it can be manipulated like any list in R.

## TMM --  Trimmed Mean of M-values -- normalization of read counts
DE_list_PW=calcNormFactors(DE_list_PW, method=c("TMM"))
# The calcNormFactors function normalizes for RNA composition by finding a set of scaling
# factors for the library sizes that minimize the log-fold changes between the samples for most
# genes. The default method for computing these scale factors uses a trimmed mean of M-
# values (TMM) between each pair of samples [*]
# [*]: Robinson, M.D. and Oshlack, A. (2010). A scaling normalization method for differential
# expression analysis of RNA-seq data. Genome Biology 11, R25.

## Estimating dispersions for the pair wise DE analysis
DE_list_PW=estimateCommonDisp(DE_list_PW)
DE_list_PW=estimateTrendedDisp(DE_list_PW)
DE_list_PW=estimateTagwiseDisp(DE_list_PW)
## Pseudo counts
# In general, edgeR functions work directly on the raw counts. For the most part, edgeR does
# not produce any quantity that could be called a “normalized count”.
# An exception is the internal use of pseudo-counts by the classic edgeR functions estimateCommonDisp
# and exactTest. The exact negative binomial test [*] computed by exactTest and the con-
# ditional likelihood [*] used by estimateCommonDisp and estimateTagwiseDisp require the
# library sizes to be equal for all samples. These functions therefore compute normalized counts
# called pseudo-counts by the method of Robinson and Smyth [*]. The pseudo-counts are
# computed for a specific purpose, and their computation depends on the experimental design
# as well as the library sizese. Users are therefore disuaded from interpreting the psuedo-counts
# as general purpose normalized counts.
# [*]: Robinson, M.D. and Smyth, G.K. (2008). Small-sample estimation of negative binomial
# dispersion, with applications to SAGE data. Biostatistics 9, 321–332.
## Average log2 CPM (Counts per million)
# log-CPM value for each count:
# log2((rgi + 0.5)/Ri+1)×10^6)
# rgi: read(pair) count of gene g for sample i
# Ri: the total number of mapped read(pair)s for sample i (i.e. the library size of sample i)

## DE testing with tagwise dispersion
DE_list_PW.tgw=exactTest(DE_list_PW, dispersion="tagwise", pair=c("C15","C23"))
# Once negative binomial models are fitted and dispersion estimates are obtained, we can proceed with testing
# procedures for determining differential expression using the exact test.
# The exact test is only applicable to experiments with a single factor.
# edgeR uses the quantile-adjusted conditional maximum likelihood (qCML) method for ex-
# periments with single factor.
# Compared against several other estimators (e.g. maximum likelihood estimator, Quasi-
# likelihood estimator etc.) using an extensive simulation study, qCML is the most reliable in
# terms of bias on a wide range of conditions and specifically performs best in the situation
# of many small samples with a common dispersion, the model which is applicable to Next-
# Gen sequencing data.
# The qCML method calculates the likelihood by conditioning on the total counts
```

```
for each
# tag, and uses pseudo counts after adjusting for library sizes.

## Which genes were differentially expressed according to the Benjamini-Hochberg corrected p-values?
Result=DE_list_PW.tgw$table
Result$adj.PValue=p.adjust(Result$PValue, method="BH")
Up=Result[Result$adj.PValue<0.05 & Result$logFC>0 ,]
Down=Result[Result$adj.PValue<0.05 & Result$logFC<0 ,]

# Write the DE genes and the Result table to files
write(rownames(Up), "DE_Up_genes.txt")
write(rownames(Down), "DE_Down_genes.txt")
write.table(Result, "Result.tsv", sep="\t")

```