USC Department Of Translational Genomics
Keck Medicine of USC

TRGN 599: Applied Data Science and Bioinformatics

UNIT V. Unsupervised Analysis, linear regression, enrichment analysis

Week 11 - Lecture 1

Enrique I. Velazquez Villarreal, M.D., Ph.D., M.P.H., M.S. | **Assistant Professor**

Dept. of Translational Genomics

USC | Keck School of Medicine | Norris Comprehensive Cancer Center

Leader of the USC Bioinformatics Core – *USC CaRE2 Health Equity Center*

David W. Craig, Ph.D. | **Professor and Vice Chair**

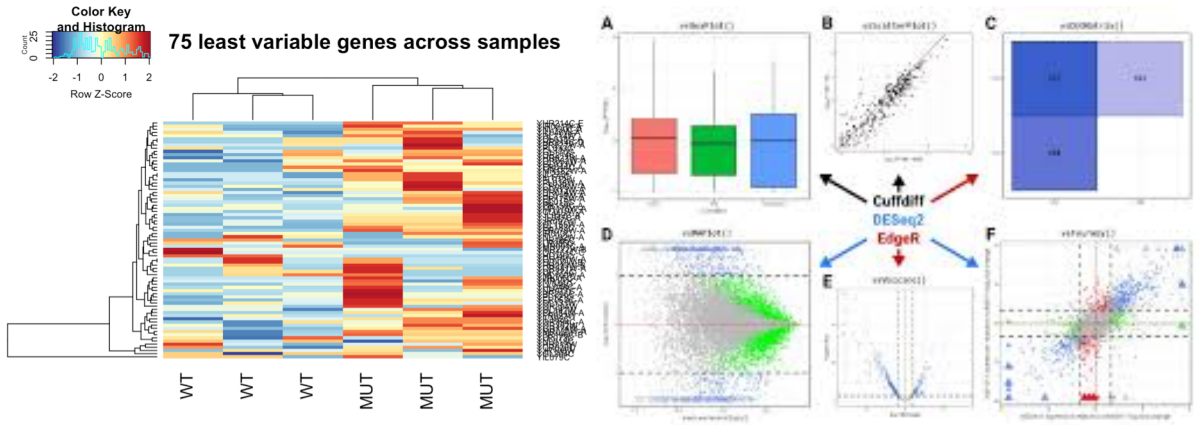
Dept. of Translational Genomics

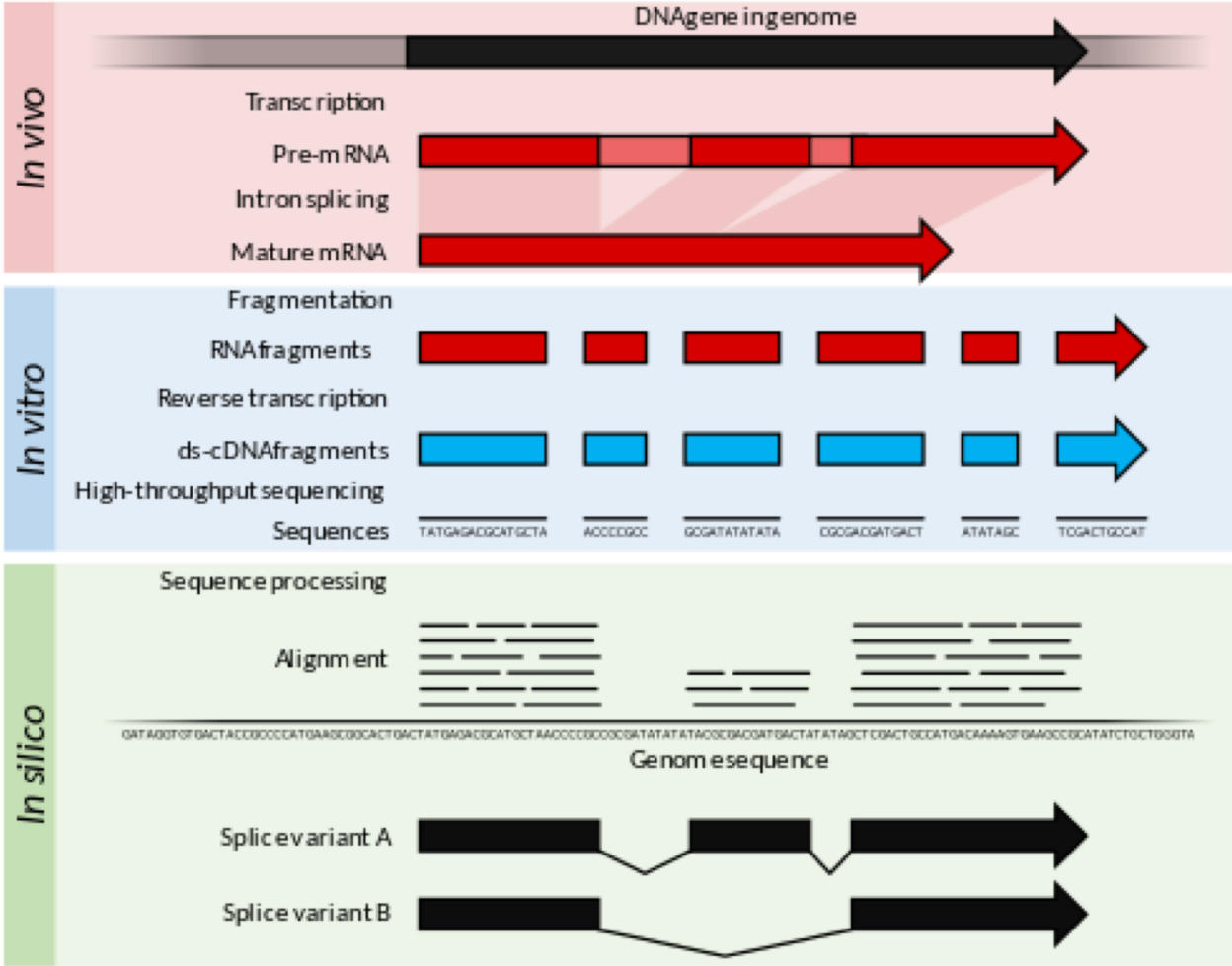
USC | Keck School of Medicine | Norris Comprehensive Cancer Center

Co-Director, Institute of Translational Genomics

Topics

- Next-Generation sequencing transcriptomics



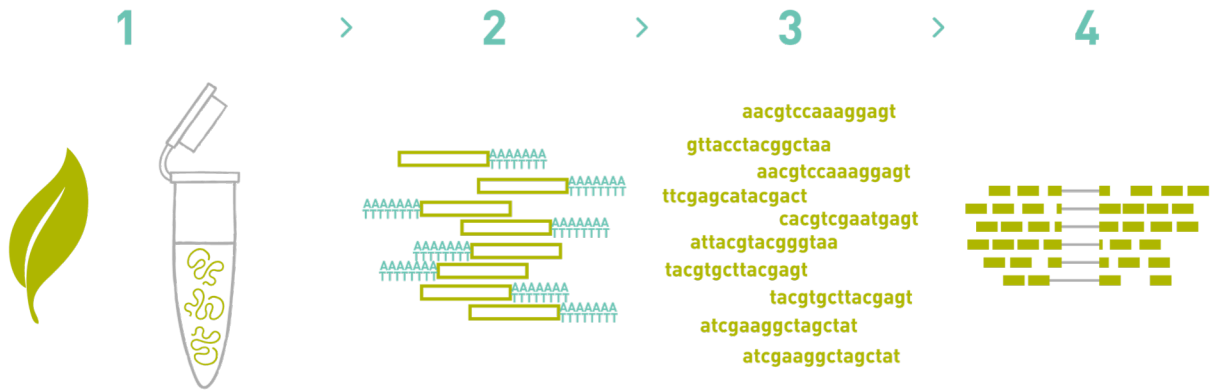


RNAseq experiments

- The rapid development of transcriptome analysis methods steers the focus of methodological research and development.

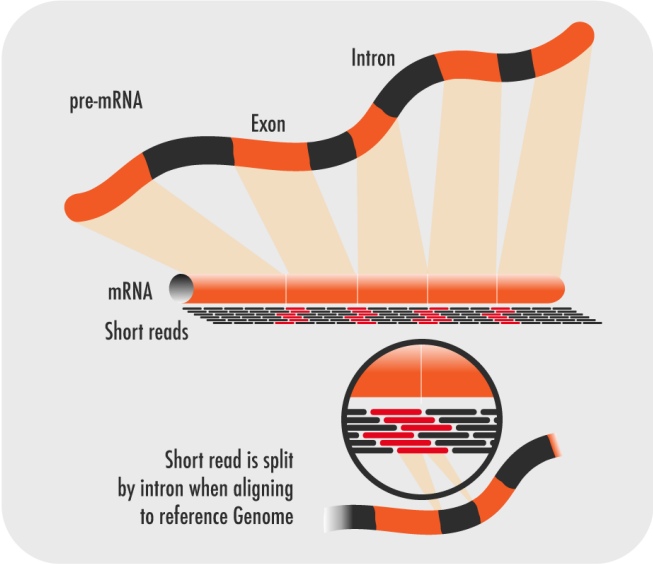
- RNA-seq is one of the most popular applications of high-throughput sequencing technologies.

- Although many concepts from gene expression microarray data analysis can be applied here, there are also specific aspects coming from next-generation sequencing (NGS) approach itself.



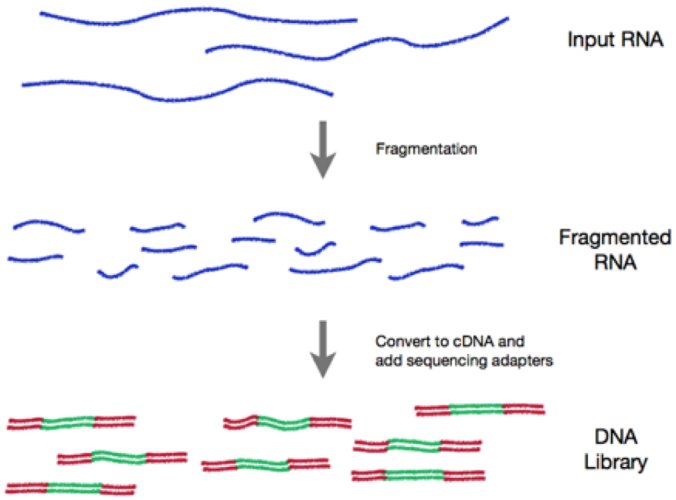
High-throughput RNA sequencing background

- The application of NGs technologies in the field of transcriptomic is a somewhat surprising development.
- The traditional purpose of DNA sequencing is to identify the nucleotide order (the sequence) of a target region.
- This kind of application of NGS is well aligned with the traditional goals and tools of genome sequencing.
- On the contrary, in transcriptomics experiments the quantities of tag sequences present in biological samples are in the focus.



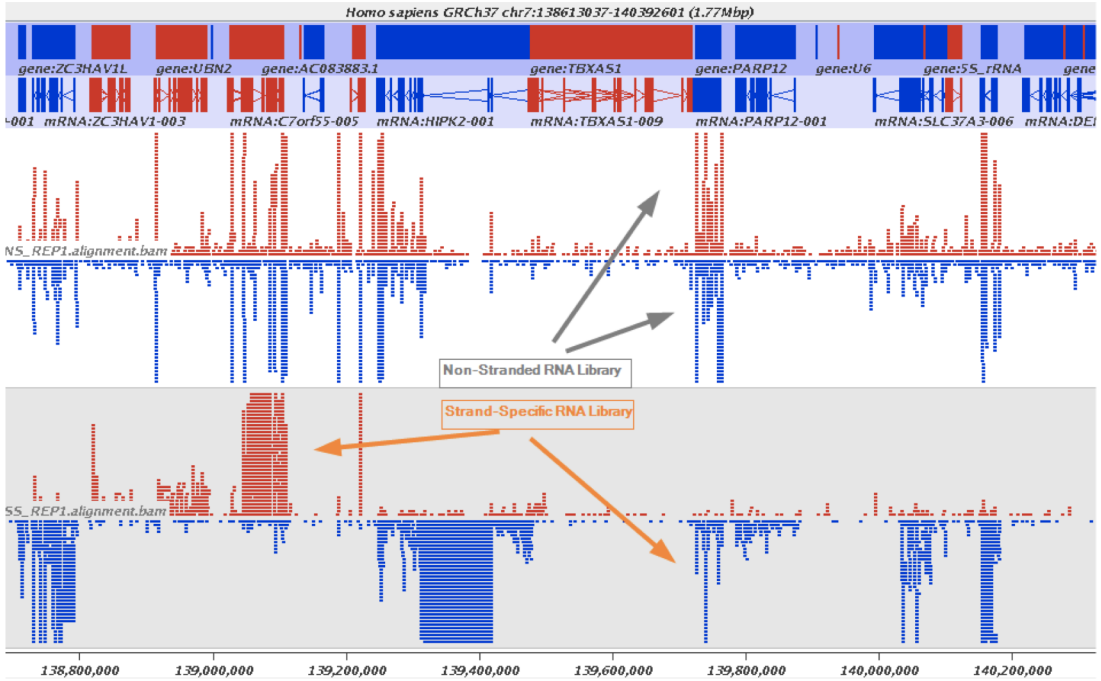
High-throughput RNA sequencing background

- While we are interested in the sequence coming out from the experiment in genomic NGS applications, in RNA-seq the relative abundance of the reads tagging specific pre-defined genomic regions is investigated.
- The idea comes from the concept of gene expression microarrays, where the intensity of a probe is proportional to the concentration of the corresponding messenger RNA (mRNA) molecules in the samples signifying their gene expression levels.
- Similarly, in RNA-seq the number of reads mapped to a gene is proportional to the number of mRNA molecules produced from that gene, and as such, to the expression levels.



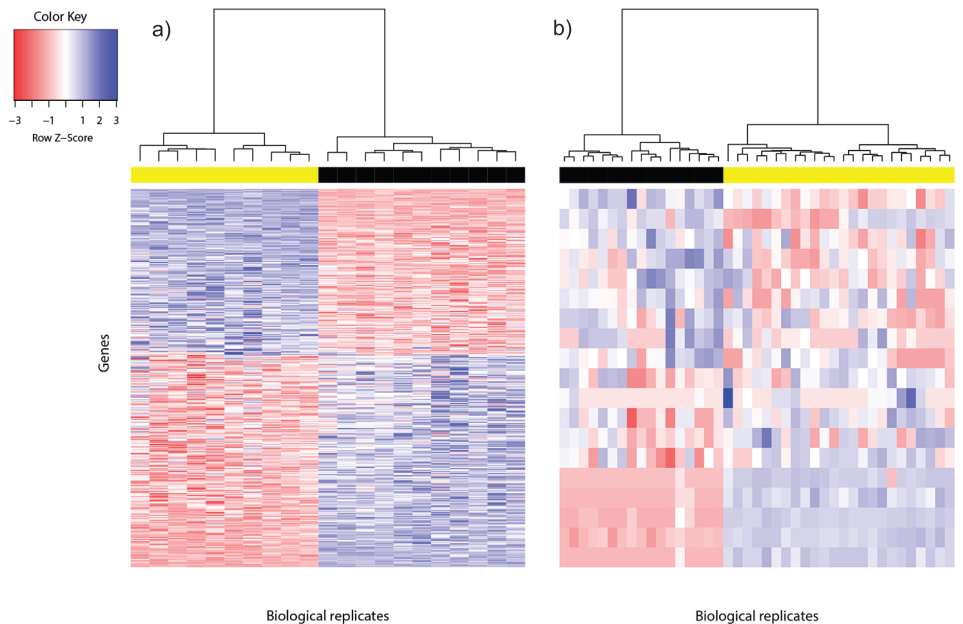
Experimental background

- The general experimental protocol is rather uniform at the superficial level.
- The first step is the RNA extraction from the biological samples.
- A crucial point here is to get rid of the genomic DNA from the samples after reverse transcription relatively new but has an important contribution to epigenetic studies.



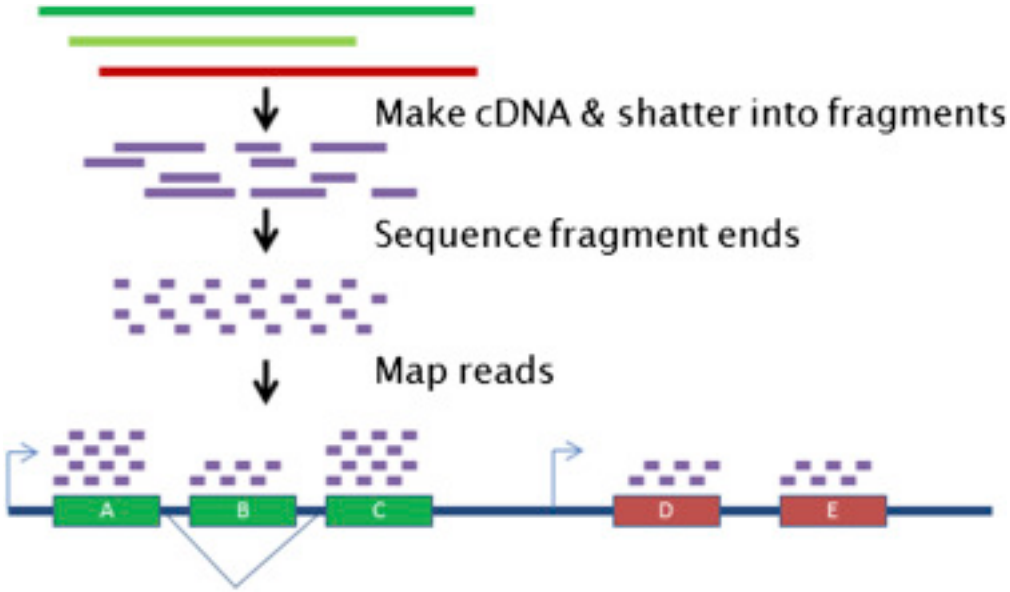
Preparing count tables

- The actual data analysis of an RNA-seq experiments starts with the FASTQ files containing the unfiltered (raw) reds.
- Quality checking and filtering is an essential step, although certain researchers reason that excluding reads from an analysis because of their low quality can introduce bias to quantitative studies.
- Most often, filtered reads must be mapped to indexed genomes.
- The majority of RNA-seq experiments is performed on model organisms, most often on human or mouse samples.



Preparing count tables

- External software are available for performing these task quickly and efficiently.
- TopHat and Cufflinks are two popular software to achieve this goal.
- It has to be noted that mapping transcriptome-originated reads are a substantially more complex problem than mapping genomic reads.
- Many reads spam exon boundaries , meaning that they will show partial matches to nonadjacent genomic regions.



Preparing count tables

- Since alternative splicing is more abundant than thought previously, transcript-read mapping cannot solely rely on the location of partial reads at the two sides of introns.
- Read mapping results alignment files, most often BAM files.
- In your Rmarkdown from the class, some bam files are chosen for demonstrative purposes from a stock dataset available from bioconductor package RNAseqData.HNRNPC.bam.chr14.

Rmarkdown_TRGN599_Week_11_Lecture_1

Enrique I. Velazquez Villarreal, MD, PhD, MPH, MS

3/25/2019

Next-Generation Sequencing in Transcriptomics

RNAseq

Preparing count tables

```
# install package: http://bioconductor.org/packages/release/data/experiment/html/RNAseqData.HNRNPC.bam.chr14.html

#if (!requireNamespace("BiocManager", quietly = TRUE))
#    install.packages("BiocManager")
#BiocManager::install("RNAseqData.HNRNPC.bam.chr14", version = "3.8")

library(RNAseqData.HNRNPC.bam.chr14)

alnFiles <- RNAseqData.HNRNPC.bam.chr14_BAMFILES

exp.des <- factor(rep(c('Control','HNRNPC knockdown'),each=4))

names(exp.des) <- RNAseqData.HNRNPC.bam.chr14_RUNNAMES
```

Preparing count tables

- In this experiment, mRNA was isolated from human HeLa cell lines.
- The heterogeneous nuclear ribonucleoprotein C (HNRNPC) gene is knocked down in four samples, while the other four samples are used as controls.
- The goal of this experiment is to investigate the general effect of HNRNPCgene on transcription.
- To reduce the size of this example dataset, the BAM files contain only reads mapped to chromosome 14, where the HNRNPC gene itself is located.

```
library(RNAseqData.HNRNPC.bam.chr14)

alnFiles <- RNAseqData.HNRNPC.bam.chr14_BAMFILES
```

Preparing count tables

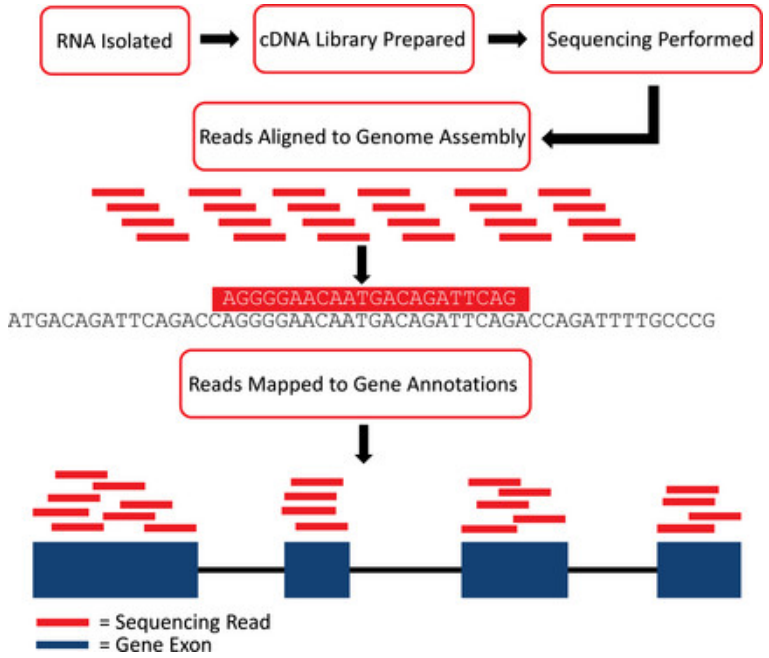
- The experimental conditions van be recorded in a named vector of factors:

```
exp.des <- factor(rep(c('Control','HNRNPC knockdown'),each=4))

names(exp.des) <- RNAseqData.HNRNPC.bam.chr14_RUNNAMES
```

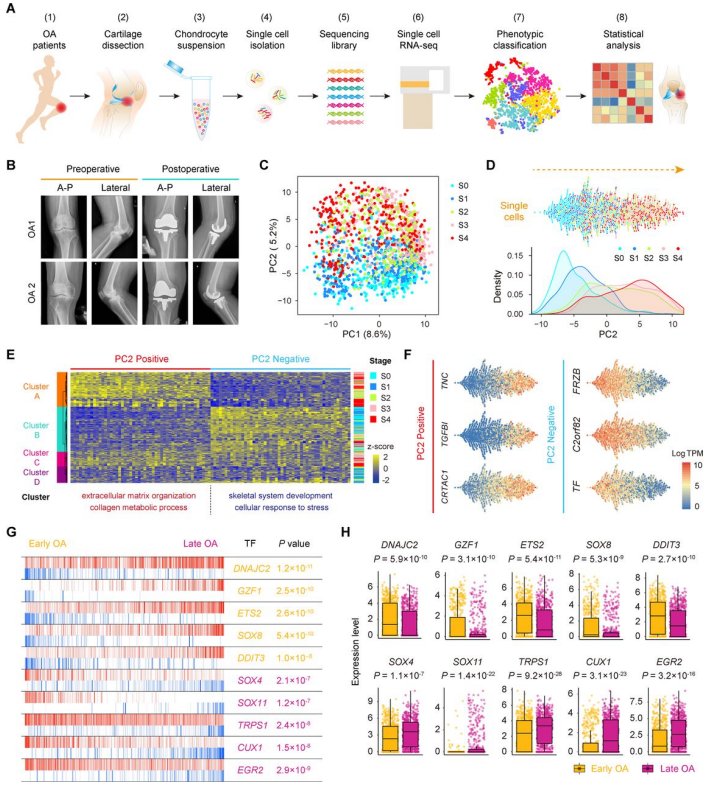
Alignment files to read counts

- A crucial data preparatory step in RNA-seq analysis is the preparation of the count tables.
- This is a simple matrix or data frame containing the investigated genomic features:
  - genes in rows OR
  - exons in rows OR
  - transcripts in rows AND
  - individual samples in columns.



Alignment files to read counts

- Here, the differential gene expression is analyzed.
- Therefore a table is needed that contains the represented genes and the number of reads mapped to their coding regions.
- There are several ways to map the genomic coordinates stored in the BAM files to the actual gene (or exon, or transcript) locations.
- Here, a transcript database package is employed from Bioconductor.



Alignment files to read counts

- To decide which database package can be used in an analysis:
  - The source organism should be taken into account.
  - Also, the build version of the genome, to which the reads were aligned should be investigated.

- Here, the reads are from human samples, and they are aligned to the hg19 genome build.
- Based on this, the TxDb.Hsapiens.UCSC.hg19.knowGene database package is applied.
- There are many other TxDB type packages in the repository, and there are also tools provide to build one for genomes and versions not yet available.

```
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
```

Alignment files to read counts

```
txdb <- TxDb.Hsapiens.UCSC.hg19.knownGene
seqlevels(txdb)
```

```
##  [1] "chr1"                  "chr2"
##  [3] "chr3"                  "chr4"
##  [5] "chr5"                  "chr6"
##  [7] "chr7"                  "chr8"
##  [9] "chr9"                  "chr10"
## [11] "chr11"                 "chr12"
## [13] "chr13"                 "chr14"
## [15] "chr15"                 "chr16"
## [17] "chr17"                 "chr18"
## [19] "chr19"                 "chr20"
## [21] "chr21"                 "chr22"
## [23] "chrX"                  "chrY"
## [25] "chrM"                  "chr1_gl000191_random"
## [27] "chr1_gl000192_random"  "chr4_ctg9_hap1"
## [29] "chr4_gl000193_random"  "chr4_gl000194_random"
## [31] "chr6_apd_hap1"         "chr6_cox_hap2"
## [33] "chr6_dbb_hap3"         "chr6_mann_hap4"
## [35] "chr6_mcf_hap5"         "chr6_qbl_hap6"
## [37] "chr6_ssto_hap7"        "chr7_gl000195_random"
## [39] "chr8_gl000196_random"  "chr8_gl000197_random"
## [41] "chr9_gl000198_random"  "chr9_gl000199_random"
## [43] "chr9_gl000200_random"  "chr9_gl000201_random"
## [45] "chr11_gl000202_random" "chr17_ctg5_hap1"
## [47] "chr17_gl000203_random" "chr17_gl000204_random"
## [49] "chr17_gl000205_random" "chr17_gl000206_random"
## [51] "chr18_gl000207_random" "chr19_gl000208_random"
## [53] "chr19_gl000209_random" "chr21_gl000210_random"
## [55] "chrUn_gl000211"        "chrUn_gl000212"
## [57] "chrUn_gl000213"        "chrUn_gl000214"
## [59] "chrUn_gl000215"        "chrUn_gl000216"
## [61] "chrUn_gl000217"        "chrUn_gl000218"
## [63] "chrUn_gl000219"        "chrUn_gl000220"
## [65] "chrUn_gl000221"        "chrUn_gl000222"
## [67] "chrUn_gl000223"        "chrUn_gl000224"
## [69] "chrUn_gl000225"        "chrUn_gl000226"
## [71] "chrUn_gl000227"        "chrUn_gl000228"
## [73] "chrUn_gl000229"        "chrUn_gl000230"
## [75] "chrUn_gl000231"        "chrUn_gl000232"
## [77] "chrUn_gl000233"        "chrUn_gl000234"
## [79] "chrUn_gl000235"        "chrUn_gl000236"
## [81] "chrUn_gl000237"        "chrUn_gl000238"
## [83] "chrUn_gl000239"        "chrUn_gl000240"
## [85] "chrUn_gl000241"        "chrUn_gl000242"
## [87] "chrUn_gl000243"        "chrUn_gl000244"
## [89] "chrUn_gl000245"        "chrUn_gl000246"
## [91] "chrUn_gl000247"        "chrUn_gl000248"
## [93] "chrUn_gl000249"
```

Alignment files to read counts

- In the beginning, all the chromosomes are represented in the database
- Since the example BAM files contain reads only form chromosome 14, it is a good idea also to restrict the transcript database to this region because it speeds up the calculations.

```
seqlevels(txdb) <- c("chr14")
```

Alignment files to read counts

- Next, the gene annotation is picked up by applying the gene() function. If the task focuses on exon- or alternative transcript-level differential expression, the corresponding functions should be employed.

```
genelist <- genes(txdb)
txlist <- transcripts(txdb)
exonlist <- exons(txdb)
```

```
genelist
```

```
## GRanges object with 781 ranges and 1 metadata column:
##             seqnames              ranges strand |     gene_id
##                <Rle>           <IRanges>  <Rle> | <character>
##       10001    chr14   71050957-71067384      - |       10001
##   100113389    chr14   45580078-45580176      + |   100113389
##   100113391    chr14   20794600-20794698      - |   100113391
##   100124539    chr14   91592770-91592896      + |   100124539
##   100126297    chr14 101507700-101507781      + |   100126297
##         ...      ...                 ...    ... .         ...
##        9870    chr14   75127955-75179807      - |        9870
##        9878    chr14   21945335-21967319      + |        9878
##        9895    chr14 102829300-102968818      + |        9895
##        9950    chr14   93260650-93306304      + |        9950
##        9985    chr14   24641234-24649463      + |        9985
##   -------
##   seqinfo: 1 sequence from hg19 genome
```

Alignment files to read counts

```
txlist
```

```
## GRanges object with 2858 ranges and 2 metadata columns:
##           seqnames              ranges strand |     tx_id     tx_name
##              <Rle>           <IRanges>  <Rle> | <integer> <character>
##       [1]    chr14   19377594-19378574      + |     50694  uc010tkp.2
##       [2]    chr14   19553365-19584942      + |     50695  uc001vuz.1
##       [3]    chr14   19553365-19584942      + |     50696  uc001vva.1
##       [4]    chr14   19553365-19584942      + |     50697  uc010ahc.1
##       [5]    chr14   19595544-19595584      + |     50698  uc021rmy.1
##       ...      ...                 ...    ... .       ...          ...
##    [2854]    chr14 106576814-106598011      - |     53547  uc001ysv.3
##    [2855]    chr14 106804495-106805235      - |     53548  uc001ysw.1
##    [2856]    chr14 106829593-106833706      - |     53549  uc001ysx.1
##    [2857]    chr14 107034169-107035191      - |     53550  uc001ysz.3
##    [2858]    chr14 107258705-107259792      - |     53551  uc001yta.1
##    -------
##    seqinfo: 1 sequence from hg19 genome
```

Alignment files to read counts

```
txlist
```

```
## GRanges object with 2858 ranges and 2 metadata columns:
##           seqnames              ranges strand |     tx_id     tx_name
##              <Rle>           <IRanges>  <Rle> | <integer> <character>
##       [1]    chr14   19377594-19378574      + |     50694  uc010tkp.2
##       [2]    chr14   19553365-19584942      + |     50695  uc001vuz.1
##       [3]    chr14   19553365-19584942      + |     50696  uc001vva.1
##       [4]    chr14   19553365-19584942      + |     50697  uc010ahc.1
##       [5]    chr14   19595544-19595584      + |     50698  uc021rmy.1
##       ...      ...                 ...    ... .       ...          ...
##    [2854]    chr14 106576814-106598011      - |     53547  uc001ysv.3
##    [2855]    chr14 106804495-106805235      - |     53548  uc001ysw.1
##    [2856]    chr14 106829593-106833706      - |     53549  uc001ysx.1
##    [2857]    chr14 107034169-107035191      - |     53550  uc001ysz.3
##    [2858]    chr14 107258705-107259792      - |     53551  uc001yta.1
##    -------
##    seqinfo: 1 sequence from hg19 genome
```

Alignment files to read counts

```
exonlist
```

```
## GRanges object with 13688 ranges and 1 metadata column:
##             seqnames              ranges strand |   exon_id
##                <Rle>           <IRanges>  <Rle> | <integer>
##        [1]    chr14   19377594-19378574      + |    179298
##        [2]    chr14   19553365-19553937      + |    179299
##        [3]    chr14   19558717-19558831      + |    179300
##        [4]    chr14   19558991-19559164      + |    179301
##        [5]    chr14   19560718-19560973      + |    179302
##        ...      ...                 ...    ... .       ...
##    [13684]    chr14 107276018-107276044      - |    192981
##    [13685]    chr14 107281126-107281145      - |    192982
##    [13686]    chr14 107281146-107281249      - |    192983
##    [13687]    chr14 107282819-107282992      - |    192984
##    [13688]    chr14 107283004-107283085      - |    192985
##    -------
##    seqinfo: 1 sequence from hg19 genome
```

Alignment files to read counts

- The next step is to read the files one by one and record the number of reads mapped to the individual genes into a data frame.

- This task is implemented here by applying a for loop, so there is a single file open at a time.

- It is also a good idea to pre-allocate a data frame with the appropriate number of rows for avoiding the running out of memory before starting the actual calculation.

Alignment files to read counts

- Here, first we open the alignment file, convert the read regions to a Granges class data structure in the reads variable, and then the countOverlap() function counts the number of reads in the genes.
- Last, cbind is used to append the result to a new column to the counts dta frame.

```
# Calculate counts per gene

library(GenomicAlignments)

counts <- data.frame(row.names=genelist$gene_id)

for (i in 1:length(RNAseqData.HNRNPC.bam.chr14_BAMFILES)){
  print(paste("Reading:",alnFiles[i]))
  aln <- readGAlignments(alnFiles[i])
  reads <- GRanges(seqnames='chr14',ranges=ranges(aln),strand='*')
  genecounts<-countOverlaps(genelist, reads)
  a <- data.frame(genecounts)
  names(a) <- RNAseqData.HNRNPC.bam.chr14_RUNNAMES[i]
  counts <- cbind(counts,a)
}
```

Alignment files to read counts

- The very same could be achieved on transcript- or exon-level by replacing the appropriate variable in the countOverlaps() function of the loop.

- There are genes in the example file without any read mapped on them.

- It is a very useful to eliminate rows that have zeros in all samples from the counts table.

```
head(counts)
```

```
##           ERR127306 ERR127307 ERR127308 ERR127309 ERR127302 ERR127303
## 10001           228       312       258       288       350       426
## 100113389         7         1         6         4         5         5
## 100113391         0         5         0         0         5        19
## 100124539         5        10         9         4         3         0
## 100126297         0         0         0         0         0         0
## 100126308         0         0         0         0         0         0
##           ERR127304 ERR127305
## 10001           420       330
## 100113389         8         0
## 100113391         9         0
## 100124539         1         1
## 100126297         0         0
## 100126308         0         0
```

```
dim(counts)
```

```
## [1] 781   8
```

Alignment files to read counts

- Here, the numbers of genes that are not represented at all or by less than ten reads in all samples together are shown.
- These can because a headache for counting statistics.
- These empty or small count rows can be eliminated with a simple data frame slicing.

```
## eliminate genes without any reads

sum(rowSums(counts)==0)
```

```
## [1] 206
```

```
sum(rowSums(counts)<10)
```

```
## [1] 262
```

Alignment files to read counts

- Now, take a closer look at the structure of this count table revelas that the row names are Entrez Gene identifiers (IDs).

```
counts <- counts[!rowSums(counts)==0,]
counts1 <- counts

## Add Symbols as row names
```

Alignment files to read counts

- It is excellent for most analysis; however, numerical IDs are not very biologist-friendly.
- Most researchers prefer to have gene symbols in their data tables.
- It is straightforward to replace gene IDs with official gene symbols using, for example, the org.HS.eg.dg package.

```
library(org.Hs.eg.db)
```

```
##
```

```
sym <- select(org.Hs.eg.db, row.names(counts), keytype="ENTREZID","SYMBOL")
```

```
## 'select()' returned 1:1 mapping between keys and columns
```

```
row.names(sym) <- sym$ENTREZID
sym$SYMBOL[is.na(sym$SYMBOL)]<-sym$ENTREZID[is.na(sym$SYMBOL)]
counts <- cbind(sym$SYMBOL,counts)
row.names(counts) <- as.character(counts$"sym$SYMBOL")
counts$"sym$SYMBOL" <- NULL
```

Alignment files to read counts

- After these preparations, the count table is more informative.
- The first four columns are the control (wild-type) samples, while the second four are the knockouts (KOs).
- Below, we see the read counts of two example genes from the RNAseqData.HNRNPC.bam.chr14 dataset. Individual samples are represented by columns.

```
counts["HNRNPC",]
```

```
##         ERR127306 ERR127307 ERR127308 ERR127309 ERR127302 ERR127303
## HNRNPC       5422      6320      5896      5558       172       196
##         ERR127304 ERR127305
## HNRNPC       316       282
```

```
counts["HSPA2",]
```

```
##        ERR127306 ERR127307 ERR127308 ERR127309 ERR127302 ERR127303
## HSPA2       2756      2768      2908      2535      1158       972
##        ERR127304 ERR127305
## HSPA2       960      1222
```

Alignment files to read counts

- At this point, it is worth saving the count table to a file, so later on different analysis methods can be applied to this table without recreating it over and over again.

```
# Creating a table
write.table(counts,file="HNRNPC_chr14_gene_read_counts.txt")
# data is ready
```

```
""
"MED6"
"SNORD127"
"SNORD126"
"SNORA11B"
"MIR300"
"MIR541"
"MIR665"
"MIR543"
"MIR208B"
"MIR889"
```

Differential expression in simple comparison

- The simplest experimental arrangements compare samples from two conditions:
  - Disease patient samples to healthy controls
  - Wild-type genotypes to targeted knock-down or Natural mutants
  - Drug compound to placebo treatments.
- These are perhaps the most frequently occurring escenarios.
- The common feature in all of these studies is that the samples are categorized into two mutually exclusive groups.
- The goal is to elucidate the features differentiating these groups.

Differential expression in simple comparison

- In the case of RNA-seq experiments, the changes in gene expression levels characterize these features.
- Theoretically, the concentration (the abundance) of mRNA molecules (or any other RNA target group) is proportional to the gene expression levels in any sample.
- The more abundant RNA molecules have a proportionally higher chance to produce fragments for sequencing.
- Therefore, the number of reads of the highly expressed genes will be higher among the results.

Differential expression in simple comparison

- Further, the theory states that the number of reads is a linear function of the gene expression levels.
- As such being the case, there is a simple linear dependence between the direct mesuement coming from the experiment and the gene expression levels.
- This relationship is dramatically different in the case of the microarray experiments, where the signal we are processing (fluorescence intensities) has a poorly defined (aka unknown) relationship to the gene expression levels.

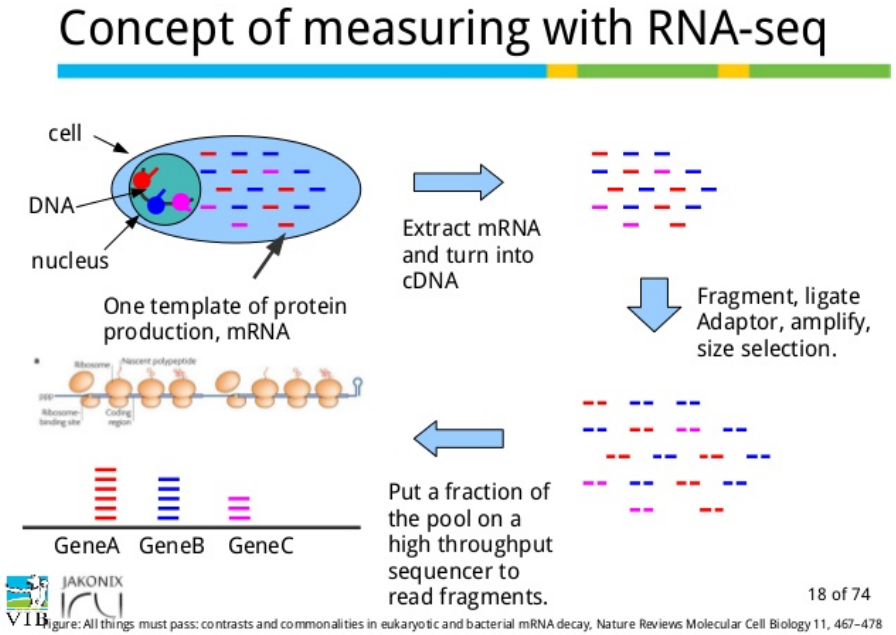


Figure: All things must pass: contrasts and commonalities in eukaryotic and bacterial mRNA decay, Nature Reviews Molecular Cell Biology 11, 467–478

Differential expression in simple comparison

- Since gene expression levels in the samples are approximated by the read counts in the count table, the table prepared previously can be employed for further calculations.

Differential expression in simple comparison

```
# Differential expression calculation with edgeR
counts <- read.table("HNRNPC_chr14_gene_read_counts.txt")
```

A naïve t-test approach

- To compare the gene expression levels in different conditions, the first instinct is to perform a simple t-test.
- The read counts of any gene can be accessed in the table.
- The first four values are for the control samples, and the second four are for the KO samples. It is straightforward to perform a t-test with R.

A naive t-test approach

```
# naive t test approach
counts["SPTLC2",]
```

```
##        ERR127306 ERR127307 ERR127308 ERR127309 ERR127302 ERR127303
## SPTLC2      2318      2818      2590      2598      4000      4658
##        ERR127304 ERR127305
## SPTLC2      3210      2630
```

```
t.test(counts["SPTLC2",1:4],counts["SPTLC2",5:8])
```

```
##
##  Welch Two Sample t-test
##
## data:  counts["SPTLC2", 1:4] and counts["SPTLC2", 5:8]
## t = -2.2882, df = 3.3174, p-value = 0.09768
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -2419.3526   332.3526
## sample estimates:
## mean of x mean of y
##    2581.0    3624.5
```

A naïve t-test approach

- The probability that the means of counts are different in the two groups is 0.097 for this particular gene meaning that the difference is not significant according to this test.

- This approach could be generalized to the entire count table with a simple for loop that would perform the same test for each row one by one.

- However this would be tedious and time-consuming.

- We employ the apply() function instead to demonstrate efficient code writing.

```
tt.p.vals <- apply(counts,1,function(dat){t.test(x = dat[1:4], y = dat[5:8])$p.value})
```

A naïve t-test approach

- Since the statistical test was perform for around 500 cases, multiple hypothesis testing really applies.
- For this purpose, the p-calues gained from the t-tests are adjusted with the false discovery rates method.
- After this step, the earlier count table is supplemented with the raw and adjusted p-values.

```
tt.adj.p.vals <- p.adjust(tt.p.vals,method="fdr")
counts <- cbind(counts,tt.p.vals,tt.adj.p.vals)

head(counts[order(tt.adj.p.vals),])
```

```
##          ERR127306 ERR127307 ERR127308 ERR127309 ERR127302 ERR127303
## HSPA2         2756      2768      2908      2535      1158       972
## ARHGEF40      1083      1158      1061      1005       440       342
## IRF2BPL       2610      2622      2855      2468       652       610
## FOS            538       536       564       466       250       196
## HNRNPC        5422      6320      5896      5558       172       196
## ATG2B         1024      1194      1174      1098       572       642
##          ERR127304 ERR127305    tt.p.vals tt.adj.p.vals
## HSPA2          960      1222 4.038036e-06  0.0007739568
## ARHGEF40       328       334 3.135935e-06  0.0007739568
## IRF2BPL        811       965 3.171795e-06  0.0007739568
## FOS            146       246 7.400111e-05  0.0065922034
## HNRNPC         316       282 7.049118e-05  0.0065922034
## ATG2B          716       572 8.025291e-05  0.0065922034
```

A naïve t-test approach

- Some genes appear with significant differences.

- Let us order the counts table according to the adjusted p-values and save its part with significant genes for future reference.

```
counts <- counts[order(tt.adj.p.vals),]
gsign.tt <- counts[counts$tt.adj.p.vals<0.03,]

dim(gsign.tt)
```

```
## [1] 56 10
```

```
gsign.tt["BDKRB1",]
```

```
##         ERR127306 ERR127307 ERR127308 ERR127309 ERR127302 ERR127303
## BDKRB1         26        20        18        16         4         2
##         ERR127304 ERR127305   tt.p.vals tt.adj.p.vals
## BDKRB1          8         8 0.002172456     0.0235691
```

A naïve t-test approach

- According to the simple t-test, the gene SPTLC2 does not show any significant difference between the two group.
- However, the count numbers seem always to be much smaller in the control samples than in the other group.
- This observation can be incorporated into t-test by changing the null hypothesis to an alternative.
- Till now, the null hypothesis declares that the two groups have equal means.
- From now on, the alternative hypothesis states that the mean is less in the first group than in the second group.

```
counts["SPTLC2",]
```

```
##        ERR127306 ERR127307 ERR127308 ERR127309 ERR127302 ERR127303
## SPTLC2      2318      2818      2590      2598      4000      4658
##        ERR127304 ERR127305 tt.p.vals tt.adj.p.vals
## SPTLC2      3210      2630 0.0976845     0.2330647
```

```
t.test(counts["SPTLC2",1:4],counts["SPTLC2",5:8], alternative = "less")
```

```
##
##  Welch Two Sample t-test
##
## data:  counts["SPTLC2", 1:4] and counts["SPTLC2", 5:8]
## t = -2.2882, df = 3.3174, p-value = 0.04884
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##       -Inf -10.86239
## sample estimates:
## mean of x mean of y
##    2581.0    3624.5
```

A naïve t-test approach

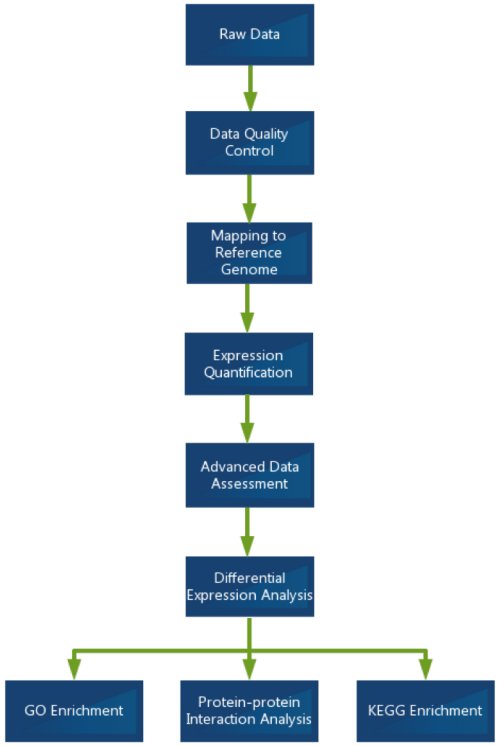
- Now the raw p-value is below the usual significance threshold, so this gene belong to the list of significantly different genes.
- How this approach can be generalized is clearly not easy.
- Besides this aspect, the t-test is not the best approach to compare read counts of two groups of RNA-seq samples.
- First of all, t-test assumes that the gene expression shows normal distribution.
- It is hard to prove or disprove.

A naïve t-test approach

- Remmember, genes without any reads were abandoned.
- The distribution of observed read counts was severely distorted at that point.
- Second, in a typical RNA-seq experiment, the two groups consist of only a handful of samples.
- Ideally, t-test expects to have over a dozen data points in the groups to reliably calculate the means.
- The math works for any number of samples over two or three.

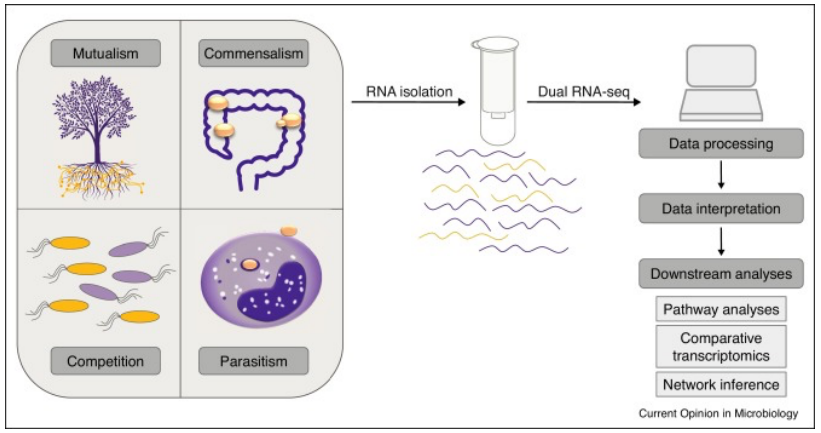
A naïve t-test approach

- Third, t-test (and any other general statistical test) starts to behave poorly on too few values.
- Unfortunately it is a common problem that there are many genes with zero or only a hand ful of reads in certain samples.
- With low read counts and only integers to be operated on, the variance of observation is escalated in a way that is not handled by general statistical test.
- A brief note: Using read count as an estimator of low gene expression level is problematic also at general levels.



A naïve t-test approach

- It is known that stochastic effects are magnified and can cause the escalation of false-positive results.
- For this reason, small read counts should not be combined with moderate sample sizes.
- If the goal is to measure the differences between genes with low general expression levels, high read counts are needed for each sample.
- This consideration should be taken into account already at the planning phase of the experiment.
- In the data analysis phase, it is common practice to exclude genes with very small read counts in any of the sample groups from the analysis.



Single factor analysis with edgeR

- As explained, although using a simple t-test (or other general mean comparing test) for finding differentially expressed genes is feasible, it is not a reliable approach.
- Dedicated libraries are available to address this issue.
- The package edgeR (Robinson, McCarthy, and Smyth 2010) provides an exact test approach closely related to but not identical with Fisher's exact test.
- Let us reload the count table from the saved file to have a fresh look.

Single factor analysis with edgeR

```
# Installing packages: https://bioconductor.org/packages/release/bioc/html/edgeR.html

#if (!requireNamespace("BiocManager", quietly = TRUE))
#    install.packages("BiocManager")
#BiocManager::install("edgeR", version = "3.8")

# exact test with edgeR
counts <- read.table("HNRNPC_chr14_gene_read_counts.txt")

library(edgeR)
```

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

```
##
## Attaching package: 'limma'
```

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

Single factor analysis with edgeR

- The first step in the analysis is to assemble a data structure holding the read count and information about which sample belongs to which experimental group.

```
library(RNAseqData.HNRNPC.bam.chr14)
exp.des <- factor(rep(c('Control','HNRNPC knockdown'),each=4))
names(exp.des) <- RNAseqData.HNRNPC.bam.chr14_RUNNAMES
genexp <- DGEList(counts=counts,group=exp.des)
genexp
```

```
## An object of class "DGEList"
## $counts
##          ERR127306 ERR127307 ERR127308 ERR127309 ERR127302 ERR127303
## MED6           228       312       258       288       350       426
## SNORD127         7         1         6         4         5         5
## SNORD126         0         5         0         0         5        19
## SNORA11B         5        10         9         4         3         0
## ZBTB42         574       630       599       486       380       319
##          ERR127304 ERR127305
## MED6           420       330
## SNORD127         8         0
## SNORD126         9         0
## SNORA11B         1         1
## ZBTB42         388       436
## 570 more rows ...
##
## $samples
##                          group lib.size norm.factors
## ERR127306              Control   781706            1
## ERR127307              Control   858265            1
## ERR127308              Control   854358            1
## ERR127309              Control   762832            1
## ERR127302 HNRNPC knockdown   725390            1
## ERR127303 HNRNPC knockdown   766539            1
## ERR127304 HNRNPC knockdown   779962            1
## ERR127305 HNRNPC knockdown   766379            1
```

Single factor analysis with edgeR

- There are two variables in the genexp data structure.
- The counts variable is identical with the read table.
- The crucial information is in the samples variable that reveals the sample designation.
  - Which is a control sample and which one is a KO.
- These definitions were provided by us, it is of no surprise here.
- The lib.size column shos the total read conts for all genes in the individual samples.

Single factor analysis with edgeR

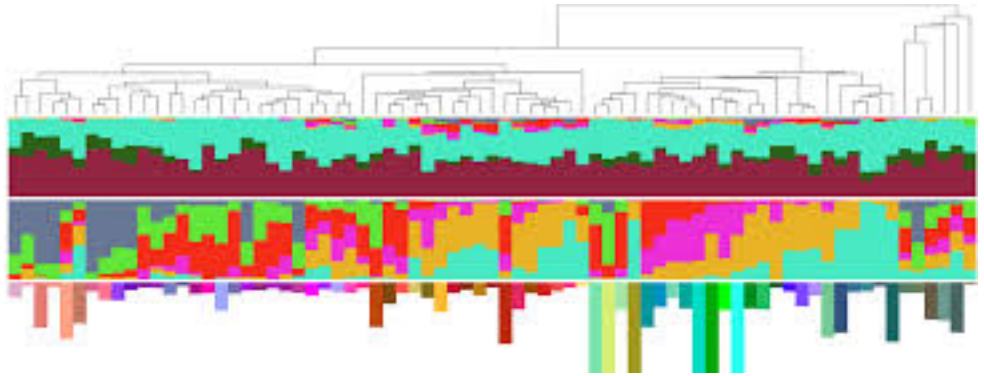
- The observed counts should be scaled to the library sizes before comparing the read counts.
- Let us calculate the normalization factors and examine the change in the norm.factor column of the samples variable.

```
genexp <- calcNormFactors(genexp)
genexp
```

```
## An object of class "DGEList"
## $counts
##          ERR127306 ERR127307 ERR127308 ERR127309 ERR127302 ERR127303
## MED6           228       312       258       288       350       426
## SNORD127         7         1         6         4         5         5
## SNORD126         0         5         0         0         5        19
## SNORA11B         5        10         9         4         3         0
## ZBTB42         574       630       599       486       380       319
##          ERR127304 ERR127305
## MED6           420       330
## SNORD127         8         0
## SNORD126         9         0
## SNORA11B         1         1
## ZBTB42         388       436
## 570 more rows ...
##
## $samples
##                          group lib.size norm.factors
## ERR127306              Control   781706    0.9636785
## ERR127307              Control   858265    0.9949162
## ERR127308              Control   854358    0.9662924
## ERR127309              Control   762832    0.9991513
## ERR127302 HNRNPC knockdown   725390    0.9905937
## ERR127303 HNRNPC knockdown   766539    1.0144734
## ERR127304 HNRNPC knockdown   779962    1.0774209
## ERR127305 HNRNPC knockdown   766379    0.9977457
```

Single factor analysis with edgeR

- Although the differences are not too high, they can be deal breakers in a real experiment with samples with smaller total read counts.
- This case can arise when the sequencing process of one of the samples was problematic, and many reads have been fallen out during quality filtering.
- In the next step, the variance of the gene expression of individual gene is calculated.
- This is the point where statisticians become upset when they realize that ther is onbly a handful of samples in the measurement groups.
- They say that there should be at least 30 sanmples to use general statistical methods, but that is practically impossible in a RNA-seq (or any other high-throughput molecular biology) experiment.



Single factor analysis with edgeR

- The developers of edgeR had a smart solution for this problem.
- Although they expect to have only a handful of samples in the experimental groups (four control and four Kos in the example), the reason that the expression levels of individual genes are hardly independent.
- So, they first estimate the gene expression variance for all genes, then use this common variability measure to moderate the gene-wide variability, compensating for the small sample size.
- They use dispersion measure to calculate the variability of genes in the sample groups.
- They define common dispersion as the variability derived rom all the genes in the same sample group while tag-wise dispersion is the variability of the same gene (or exon, or transcript).

Single factor analysis with edgeR

```
genexp <- estimateCommonDisp(genexp)
genexp
```

```
## An object of class "DGEList"
## $counts
##          ERR127306 ERR127307 ERR127308 ERR127309 ERR127302 ERR127303
## MED6           228       312       258       288       350       426
## SNORD127         7         1         6         4         5         5
## SNORD126         0         5         0         0         5        19
## SNORA11B         5        10         9         4         3         0
## ZBTB42         574       630       599       486       380       319
##          ERR127304 ERR127305
## MED6           420       330
## SNORD127         8         0
## SNORD126         9         0
## SNORA11B         1         1
## ZBTB42         388       436
## 570 more rows ...
##
## $samples
##                       group lib.size norm.factors
## ERR127306           Control   781706    0.9636785
## ERR127307           Control   858265    0.9949162
## ERR127308           Control   854358    0.9662924
## ERR127309           Control   762832    0.9991513
## ERR127302 HNRNPC knockdown   725390    0.9905937
## ERR127303 HNRNPC knockdown   766539    1.0144734
## ERR127304 HNRNPC knockdown   779962    1.0774209
## ERR127305 HNRNPC knockdown   766379    0.9977457
##
## $common.dispersion
## [1] 0.03044734
##
## $pseudo.counts
##             ERR127306   ERR127307  ERR127308    ERR127309  ERR127302
## MED6     237.89338015 287.1885115 245.505045 296.86245856 382.753308
## SNORD127   7.24905009   0.8197895   5.735218   4.13043296   5.439698
## SNORD126   0.01235354   4.7558705   0.000000   0.00894809   5.568578
## SNORA11B   5.24592527   9.2793048   8.597830   4.15732276   3.195431
## ZBTB42   598.68992977 579.7635384 570.134951 501.09180455 415.441595
##             ERR127303  ERR127304    ERR127305
## MED6     4.304390e+02 392.732789 339.16547154
## SNORD127 5.050924e+00   7.567900   0.02711416
## SNORD126 1.914896e+01   8.411819   0.04519177
## SNORA11B 3.170028e-03   0.919916   1.03276835
## ZBTB42   3.223636e+02 362.733641 447.97595370
## 570 more rows ...
##
## $pseudo.lib.size
## [1] 785768.4
##
## $AveLogCPM
## [1] 8.707554 3.046485 3.099257 2.957368 9.245208
## 570 more elements ...
```

Single factor analysis with edgeR

- There are many new variables in the data structure now.
- The pseudo.counts variable represents the read numbers normalized with the library sizes.
- The variable common.dispersion holds information about the variability of all gene measures in the table.

Single factor analysis with edgeR

```
genexp <- estimateTagwiseDisp(genexp)
genexp
```

```
## An object of class "DGEList"
## $counts
##          ERR127306 ERR127307 ERR127308 ERR127309 ERR127302 ERR127303
## MED6           228       312       258       288       350       426
## SNORD127         7         1         6         4         5         5
## SNORD126         0         5         0         0         5        19
## SNORA11B         5        10         9         4         3         0
## ZBTB42         574       630       599       486       380       319
##          ERR127304 ERR127305
## MED6           420       330
## SNORD127         8         0
## SNORD126         9         0
## SNORA11B         1         1
## ZBTB42         388       436
## 570 more rows ...
##
## $samples
##                          group lib.size norm.factors
## ERR127306          Control   781706    0.9636785
## ERR127307          Control   858265    0.9949162
## ERR127308          Control   854358    0.9662924
## ERR127309          Control   762832    0.9991513
## ERR127302 HNRNPC knockdown   725390    0.9905937
## ERR127303 HNRNPC knockdown   766539    1.0144734
## ERR127304 HNRNPC knockdown   779962    1.0774209
## ERR127305 HNRNPC knockdown   766379    0.9977457
##
## $common.dispersion
## [1] 0.03044734
##
## $pseudo.counts
##             ERR127306   ERR127307  ERR127308    ERR127309  ERR127302
## MED6     237.89338015 287.1885115 245.505045 296.86245856 382.753308
## SNORD127   7.24905009   0.8197895   5.735218   4.13043296   5.439698
## SNORD126   0.01235354   4.7558705   0.000000   0.00894809   5.568578
## SNORA11B   5.24592527   9.2793048   8.597830   4.15732276   3.195431
## ZBTB42   598.68992977 579.7635384 570.134951 501.09180455 415.441595
##             ERR127303  ERR127304    ERR127305
## MED6     4.304390e+02 392.732789 339.16547154
## SNORD127 5.050924e+00   7.567900   0.02711416
## SNORD126 1.914896e+01   8.411819   0.04519177
## SNORA11B 3.170028e-03   0.919916   1.03276835
## ZBTB42   3.223636e+02 362.733641 447.97595370
## 570 more rows ...
##
## $pseudo.lib.size
## [1] 785768.4
##
## $AveLogCPM
## [1] 8.707554 3.046485 3.099257 2.957368 9.245208
## 570 more elements ...
##
## $prior.df
## [1] 10
##
## $prior.n
## [1] 1.666667
##
## $tagwise.dispersion
## [1] 0.02161196 0.18620898 0.78806405 0.08505375 0.02024114
## 570 more elements ...
##
## $span
## [1] 0.3
```

Single factor analysis with edgeR

- The most important new element here is in tagwise.dispersion variable.
- This has the same length as the number of rows in the count table, and it represent the gene-wide variation.
- One can take a look at the most and least variable genes.

Single factor analysis with edgeR

```
head(genexp$counts[order(genexp$tagwise.dispersion),])
```

```
##           ERR127306 ERR127307 ERR127308 ERR127309 ERR127302 ERR127303
## TTC5            324       378       372       322       516       568
## ENTPD5         1009      1116      1044       967       996      1002
## MAPK1IP1L      2586      2834      2836      2608      2216      2468
## SLIRP           596       669       702       632       874       944
## ATG2B          1024      1194      1174      1098       572       642
## HAUS4           764       880       822       772       716       705
##           ERR127304 ERR127305
## TTC5            600       500
## ENTPD5         1136      1056
## MAPK1IP1L      2620      2480
## SLIRP          1140       958
## ATG2B           716       572
## HAUS4           852       798
```

```
tail(genexp$counts[order(genexp$tagwise.dispersion),])
```

```
##              ERR127306 ERR127307 ERR127308 ERR127309 ERR127302 ERR127303
## SYNDIG1L             0         0         1         0         0         0
## SYT16                0         0         0         0         0         0
## DLK1                 0         2         0         5         0         6
## PLD4                 0         0         0         0         0         0
## LINC00640            0         2         0         0         0         0
## LOC100506071         0         2         0         0         0         0
##              ERR127304 ERR127305
## SYNDIG1L             0         0
## SYT16                2         0
## DLK1                 0         0
## PLD4                 2         0
## LINC00640            0         0
## LOC100506071         0         0
```

Single factor analysis with edgeR

- It is not surprising that the genes with the highest variability have a lot of zero counts in several samples.
- Now everything is ready for performing the exact test and picking uyp differentially expressed genes.

```
genediff <- exactTest(genexp)
genediff
```

```
## An object of class "DGEExact"
## $table
##                logFC   logCPM       PValue
## MED6       0.53328984 8.707554 0.0009302573
## SNORD127   0.01978656 3.046485 1.0000000000
## SNORD126   2.67855268 3.099257 0.0260527953
## SNORA11B -2.31974092 2.957368 0.0007484396
## ZBTB42   -0.53912849 9.245208 0.0004392354
## 570 more rows ...
##
## $comparison
## [1] "Control"          "HNRNPC knockdown"
##
## $genes
## NULL
```

Single factor analysis with edgeR

```
topTags(genediff)
```

```
## Comparison of groups:  HNRNPC knockdown-Control
##                 logFC    logCPM        PValue           FDR
## HNRNPC     -4.551355 11.886747 4.632204e-127 2.663517e-124
## EXOC3L4    -3.272266  8.059754  4.854015e-55  1.395529e-52
## NFATC4     -2.548098  8.007588  2.936992e-33  5.629235e-31
## IRF2BPL    -1.755284 11.067865  5.182513e-25  7.449862e-23
## ZFHX2      -1.849398  7.899158  8.008260e-21  7.962447e-19
## ARHGEF40   -1.523177  9.833366  8.308641e-21  7.962447e-19
## LINC00641 -2.308960  6.642652  5.078366e-19  4.171515e-17
## FOXN3       1.292207 10.293310  1.746678e-16  1.255425e-14
## HSPA2      -1.295521 11.241673  1.516631e-15  9.689586e-14
## MIS18BP1    1.380263  9.499867  5.944031e-15  3.417818e-13
```

Single factor analysis with edgeR

- The genediff data structure holds information about the level of differential expression and its significance in its table variable.
- Although adjusted p-values can be printed by employing the topTags() function (see the FDR column)
- We have to hack the genediff variable a bit to store these values there

```
genediff$table <- cbind(genediff$table,FDR=p.adjust(genediff$table$PValue,method="fdr"))
gsign <- genediff$table[genediff$table$FDR<0.03,]
gsign <- gsign[order(gsign$FDR),]
dim(gsign)
```

```
## [1] 182   4
```

```
head(gsign)
```

```
##              logFC    logCPM        PValue           FDR
## HNRNPC   -4.551355 11.886747 4.632204e-127 2.663517e-124
## EXOC3L4  -3.272266  8.059754  4.854015e-55  1.395529e-52
## NFATC4   -2.548098  8.007588  2.936992e-33  5.629235e-31
## IRF2BPL  -1.755284 11.067865  5.182513e-25  7.449862e-23
## ARHGEF40 -1.523177  9.833366  8.308641e-21  7.962447e-19
## ZFHX2    -1.849398  7.899158  8.008260e-21  7.962447e-19
```

Single factor analysis with edgeR

- The logFC column holds the 2-based logarithm of differential expression values.
- They are comparable to the logFC values coming of microarray data analysis.
- Therefore, they are familiar for data analysts and downstream methods developed earlir for microarray results can handle them without problems.
- It is educational to compare these latest results to those originating form the t-test

Single factor analysis with edgeR

```
head(gsign, 10)
```

```
##               logFC    logCPM        PValue           FDR
## HNRNPC     -4.551355 11.886747 4.632204e-127 2.663517e-124
## EXOC3L4    -3.272266  8.059754  4.854015e-55  1.395529e-52
## NFATC4     -2.548098  8.007588  2.936992e-33  5.629235e-31
## IRF2BPL    -1.755284 11.067865  5.182513e-25  7.449862e-23
## ARHGEF40   -1.523177  9.833366  8.308641e-21  7.962447e-19
## ZFHX2      -1.849398  7.899158  8.008260e-21  7.962447e-19
## LINC00641 -2.308960  6.642652  5.078366e-19  4.171515e-17
## FOXN3       1.292207 10.293310  1.746678e-16  1.255425e-14
## HSPA2      -1.295521 11.241673  1.516631e-15  9.689586e-14
## MIS18BP1    1.380263  9.499867  5.944031e-15  3.417818e-13
```

```
head(gsign.tt, 10)
```

```
##           ERR127306 ERR127307 ERR127308 ERR127309 ERR127302 ERR127303
## HSPA2          2756      2768      2908      2535      1158       972
## ARHGEF40       1083      1158      1061      1005       440       342
## IRF2BPL        2610      2622      2855      2468       652       610
## FOS             538       536       564       466       250       196
## HNRNPC         5422      6320      5896      5558       172       196
## ATG2B          1024      1194      1174      1098       572       642
## PLEKHH1         590       602       589       555       821       763
## WDR25           735       850       819       738       474       466
## HEATR5A        7962     10118      8762      8618      3570      4725
## SNHG10           89       123       104       123       182       210
##           ERR127304 ERR127305    tt.p.vals tt.adj.p.vals
## HSPA2           960      1222 4.038036e-06  0.0007739568
## ARHGEF40        328       334 3.135935e-06  0.0007739568
## IRF2BPL         811       965 3.171795e-06  0.0007739568
## FOS             146       246 7.400111e-05  0.0065922034
## HNRNPC          316       282 7.049118e-05  0.0065922034
## ATG2B           716       572 8.025291e-05  0.0065922034
## PLEKHH1         817       832 5.259161e-05  0.0065922034
## WDR25           406       359 1.043649e-04  0.0075012253
## HEATR5A        4422      3506 2.400289e-04  0.0115013856
## SNHG10          207       199 1.879790e-04  0.0115013856
```