Introduction to RNA-Seq data analysis

Cancer Genomics Course, EMBL-EBI

Alexey Larionov

20 June 2018

Plan of the practical session

1. Introduction
   - Environment
2. Alignment with *STAR*
   - STAR index and CTAT genome library
   - Source FASTQ files
   - Alignment
3. Explore RNA-Seq BAM
   - *samtools*
   - *BAMSeek*
   - *IGV*
4. Calling variants using *GATK-3*
5. *STAR-fusion*
   - Detecting fusion transcripts
   - Visualise fusion transcripts using *Chimeraviz*
6. Calculating transcript counts with *Salmon*
7. Differential gene expression with *DESeq2*
   - Prepare R environment and accessory data
   - Import Salmon counts into R using *tximport*
   - Read data to *DESeq-DataSet* object
   - Calculate differentially expressed genes
   - Visualise results

1. Introduction

RNA-Seq data can be analysed in many different ways. Numerous good tools and algorithms have been developed to process RNA-Seq data. This session will provide practical experience with a number of selected tasks and tools, aiming to illustrate the diversity of RNA-Seq data analysis.

This document contains fragments of code, which could be used interactively in terminal or R-Studio. However, to follow the **reproducible research** practieces, it is recommended to run the analysis using scripts and making logs. This document is accompanied with the scripts, which should be reviewed during the practical. Most of the scripts may be executed (some may need to be modifyed before execution). All links provided in this text were verifyed on 14th of June 2018.

Environment

Check the number of cores and amount of memory available on your machine (hint: you may use **htop** command in terminal). Some of the tools allow for multy-processing: the number of requiested threads should not exceed the number of available cores. Data, tools and resources for this session are located in folder **~/Documents/RNA-Seq**
Familiarise youreself with the content of this folder.

2. Alignment with STAR

STAR is a splicing-aware aligner, which could be used to align RNA-Seq data to reference genome. It is able to deal with both short and long reads and has many other interesting features:
https://academic.oup.com/bioinformatics/article/29/1/15/272537

An important practical consideration about STAR is that it requires a large volume of RAM: at least 10x of the size of reference genome. Human geneme is ~3GB. Check that your machine has at least 40GB of RAM for comfortable work during this session.

A very good practical introduction to current STAR features is available here:
https://currentprotocols.onlinelibrary.wiley.com/doi/full/10.1002/0471250953.bi1114s51

A full reference manual is available here:
https://github.com/alexdobin/STAR/blob/master/doc/STARmanual.pdf

STAR index and CTAT genome library

STAR should index reference genome before the alignment. The index is build once for each genome. It could be done by STAR itself using FASTA genome file, as described in the manuals indicated above. However, for the purpose of this practical session we will use index included into Cancer Transcriptome Analysis Toolkit (**CTAT**) libray from the Trinity progect:
https://github.com/NCIP/Trinity_CTAT/wiki

GRCh38 CTAT library takes ~40GB on hard drive. It includes STAR-index and other resources, whch will be needed in this practical session. The bundle was downloaded and placed in your VM in the following folder:
~/Documents/RNA-Seq/resources/ctat_b38_lib

STAR index is located inside the CTAT-lib folder; it is called: **ref_genome.fa.star.idx**

Also the CTAT-lib folder contains

- reference genome: **ref_genome.fa**
- and genome annotation file: **ref_annot.gtf**

Make yourself familiar with the content of ".gtf" file (hint: you may use **head** command in terminal)

Source FASTQ files

FASTQ files for SRR3534924 sample were downloaded from SRA (unrestricted access):
https://trace.ncbi.nlm.nih.gov/Traces/sra/?run=SRR3534924

It is paired-end Illumina RNA-Seq data from a human lung adenocarcinoma cell line. It was recently generated in a study, which investigated mechanisms of resistance in lung cancer:
https://www.ncbi.nlm.nih.gov/pubmed/29669761

~100 MBases of data were extracted from the full SRR3534924 FASTQ files to reduce the sise and to make it feasible to complete the analysis within this short practical session. The extracted slice includes reads for EGFR, KRAS, ALK and some other genes relevant to lung cancer biology and treatment.

The source FASTQ files are located on your machine in the following folder:
~/Documents/RNA-Seq/data/lung_cancer/fastq

Alignment with STAR

STAR alignment is launched by the following command:

```
STAR \
--runThreadN 10 \
--genomeDir "${star_index}" \
--sjdbGTFfile "${gtf_file}" \
--readFilesIn "${fastq1}" "${fastq2}" \
--readFilesCommand zcat \
--outFileNamePrefix "${out_folder}/" \
--outSAMtype BAM Unsorted \
--outSAMstrandField intronMotiff \
--twopassMode Basic
```

The locations of the STAR index, annotation file (gtf) and FASTQ files have been discussed previously. The intended location of the output folder is ~**/Documents/RNA-Seq/data/lung_cancer/alignment**

The script illustrating STAR alignment is located in ~**/Documents/RNA-Seq/scripts** folder. Review this script. You may use this script as an example, or you may execute it to run the alignment.

The script uses **samtools** to sort and index BAM file generated by STAR. Could STAR generate sorted BAM? (hint: you may use **STAR** –**help** command in terminal).

The alignment may take 10-15 min, depending on the number of requested threads and available memory. You may use **htop** command in a separate terminal to monitor resources used by STAR during the run.

While the analysis is running you may familiarise yourself with later parts of this practical e.g.:

- practice IGV for viewing RNA-Seq data as described in the **IGV part of Section 3**, or
- review variant-calling procedure in Section 4 (script **star_gatk3.sh**)

3. Explore RNA-Seq BAM

Explore content of the STAR output folder. What was the percentage of uniquely mapped reads? (hint: you may look into **Log.final.out** file).

samtools

Samtools is a powerful toolset, which can be used for variant calling and for many other tasks concerning BAM/SAM files:
http://www.htslib.org/doc/samtools.html

We already used samtools for sorting and indexing BAM file after the alignment. Other samtools components, which could be useful for BAM assessment include **flagstat** and **view**.

Execute the following commands in the terminal (use acual location of your STAR output folder):

```
cd "${out_folder}"
samtools flagstat Aligned.sort.bam
samtools view Aligned.sort.bam | head
samtools view Aligned.sort.bam | head -n 100 | awk '$6 ~ "N"'
```

What the last command does?
What is special about the read shown by the last command?

BAMSeek

BAMSeek is a handy GUI tool to explore BAM content at low level. You can find it in ~**/Documents/RNA-Seq/tools**
Launch it by doble-clicking on the icon; then use menu to navigate and open the BAM file. When opening a BAM file for the first time, BAMSeek creates its own index for the file. Can you see the content of the BAM file header? Is BAM file sorted? What version of BAM-format is used?

IGV

IGV (Integrated Genome Viewer) is the *de-facto* standard for exploring and visualising of BAM files. Most likely you have already used it during the course to explore BAM files generated in DNA-sequencing. However, the default IGV settings may need to be adjusted for comfortable viewing of *spliced* RNA alignments, as described here:
https://software.broadinstitute.org/software/igv/splice_junctions

Also, IGV provides special tool for viewing *Sashimi plots*:
http://software.broadinstitute.org/software/igv/Sashimi

Adjusting IGV settings

First, go to **Menu > View > Preferences > Alignments**:

- Select **Show Splice Junction Track**
- Set **Visibility range 1000** (or 500, if 1000 makes IGV incomfortably slow)
- Note the possibility of configuring **Junction Track**

Later, when viewing RNA-Seq data, explore effects **Autoscale** and **Expand/Collapse** options on junction truck (hint: use the *context menu = right click* to access these options). You may also colour alignments by "first of pair strand".

Tutorial RNA-Seq data available on IGV server

You may use RNA-Seq data from IGV Server to explore RNA-Seq data, while STAR is still performing our own alignment:

- Set genome to **hg19** (this is an old version of human reference genome)
- Load data: **Menu > File > Load from Server > Tutorials > RNA-Seq (Body Map)**
- Navigate to **SLC25A3** gene
- Switch "on" **Autoscale** and apply **Expand** option to the junction truck(s)
- Show **Sashimi plots** for both heart and liver (right click on a junction truck -> Sashimi plot)
- Using context menu (right click) on Sashimi plot:
  - Show/Hide **exon coverage data**
  - Set **minimal junctions coverage** to 10 (or pick the threshold that you like :)
  - Find whether the transcripts came from **Forward or Reverse strand**

Is there any evidence for alternative splicing of SLC25A3 gene in these two tissues?

Explore alignments for SRR3534924 sample

After *STAR* completes the alignment, and *samtools* sorts and index BAM file (Section 2), you may explore SRR3534924 sample in IGV. However, it may be a good idea to start the variant calling first (Section 4), and explore the BAM file wile GATK-3 is calling variants.

- Set genome to **hg38** for SRR3534924 sample
- Load sorted indexed BAM file generated in Section 2 (**Aligned.sort.bam**)
- Explore **CCND1** and **CCNB1** genes: try Autoscale, Expand/Collapse, Sashimi plot. Which of these genes does not show evidence of alternative splicing? Are they expressed on forward or reverse strand?
- Another gene that may be interesting to explore is **CSDE1**. Some of its transcripts skip Exon 2. Is it a common event? Is exon 3 used by any transcript? Is this gene expressed on Forward or Reverse strand?
- Look at **ALK** gene: can you see any irregularity in expression of this gene?
- You may also explore **NRAS**, **KRAS**, **HRAS**, **EGFR** and **CDKN3** (if you have time :)
- Why there is no expression of **ACTB**? (hint: I made a slice of the data :)

4. Calling variants using GATK-3

Absence of current workflow for RNA-Seq variant calling

There is no current commonly accepted workflow for short variant calling in RNA-Seq (June 2018). The current web-page for RNA-seq variant-calling at the Broad institute is available here:
https://software.broadinstitute.org/gatk/best-practices/workflow?id=11164
However, at the beginning of June 2018 it just indicated that "This workflow is in development".

GATK Best Practices workflow of 2015

A workshop on RNA-Seq data analysis would be incomplete without the variants calling. So, we will use the latest Broad's Best Practice for RNA-Seq variants calling dating back to 2015:
https://software.broadinstitute.org/gatk/documentation/article.php?id=3891
https://gatkforums.broadinstitute.org/gatk/discussion/3892
https://drive.google.com/drive/folders/1lJpc1AgqyA-NxEmmjWJGTZOeQCJ_MmjU

It's main difference from a DNA-variant-calling workflow is in BAM pre-processing. Because RNA-Seq BAMs contain **N-reads** (remember the read detected by *samtools* in section 3?) the RNA-Seq workflow includes an additional step to split such reads before the variant calling:

```
java -jar "${gatk3}" \
 -T SplitNCigarReads \
 -R "${ref_genome}" \
 -I dedupped.bam \
 -o splitNCigar.bam \
 -rf ReassignOneMappingQuality \
 -RMQF 255 \
 -RMQT 60 \
 -U ALLOW_N_CIGAR_READS \
 --validation_strictness LENIENT
```

The other pre-processing steps (such as removing PCR duplicates, local realignment around indels, BQSR etc) could be applied as appropriate for DNA reads. The actual variant calling is done by Haplotype Caller. Filtering is done using hard filters because VQSR can not be applied to RNA-Seq data. The workflow should use the same reference genome as was used for STAR alignment (included into CTAT genome library).

Variant calling script

A minimal implementation of this procedure is provided in script **star_gatk3.sh**. Review and run this script. The run may take 30-40 min, depending on the number of requiested threads for Haplotype Caller.

Importantly, because this workflow was designed in 2015, it still requires GATK-3 (the current version is GATK-4). Accordingly, the implementation uses Java-8, which was the major Java release at the time.

Exploring results

Check the content of GATK output folder. It should include filtered VCF file along with the deduplicated BAM used for variant calling. The VCF file could be explored like and other VCF generated from DNA-Seq (e.g. *vcfstat* from *bcftools* etc - not done in this session).

For this practical session you may:

- Load the VCF and BAM files to IGV
- Add **All SNPs** annotation from IGV server
- Check one of the known mutations clinically relevant in lung cancer, e.g.:

  EGFR T790M

  - https://cancer.sanger.ac.uk/cosmic/mutation/overview?id=6240
  - https://www.ncbi.nlm.nih.gov/pubmed/29070957

  KRAS G12C

  - https://cancer.sanger.ac.uk/cosmic/mutation/overview?id=516
  - https://www.ncbi.nlm.nih.gov/pubmed/23122493

5. STAR-fusion

Detecting fusion transcripts

Fusion transcripts are emerging as clinically important targets in oncology. STAR-Fusion allows fast and sccurate fusion detection from RNA-Seq data:
https://www.biorxiv.org/content/early/2017/03/24/120295
https://github.com/STAR-Fusion/STAR-Fusion/wiki

STAR-fusion may use BAM files generated by STAR. However, in most cases it is preferrable to run STAR-fusion starting from the original FASTQ files. In this case, STAR-fusion runs STAR-alignment before calling fusions anyway. However, it runs STAR-alignment with the parameters optimised for fusion detection.

STAR fusion launcher is implemented as a Python script. It requires STAR, samtools and CTAT genome library. A STAR-fusion analysis, starting from FASTQ files may be launched like this:

```
PATH="/path/to/STAR-Fusion-v1.4.0":$PATH
PATH="/path/to/STAR-2.6.0c/bin/Linux_x86_64":$PATH
PATH="/path/to/samtools-1.8/bin":$PATH

STAR-Fusion \
--genome_lib_dir "${ctat_lib_folder}" \
--left_fq "${fastq1}" \
--right_fq "${fastq2}" \
--CPU 12 \
--output_dir "${out_folder}"
```

CPU parameter describes the number of threads, requisted for STAR-aligner. Again, the full script is provided in the scripts folder (**star_fus.sh**). Review this script and run STAR-fusion. The run may take 5-10 min, depending on the available RAM and number of requested threads.

After the run, explore the content of the output folder. Amongst the other files it should contain **star-fusion.fusion_predictions.tsv** and **star-fusion.fusion_predictions.abridged.tsv** files. Open the last file in LibreOffice Calc or Excel (dont make or save any changes - this file will be used in downstream analysis!).

What fusions are supported mainly by split reads, rather then by spanning fragments? What fusion has clinically relevant annotations?

Visualise fusion transcripts using *Chimeraviz*

Chimeraviz is a simple R tool to visualise results of STAR-Fusion:
https://academic.oup.com/bioinformatics/article/33/18/2954/3835381

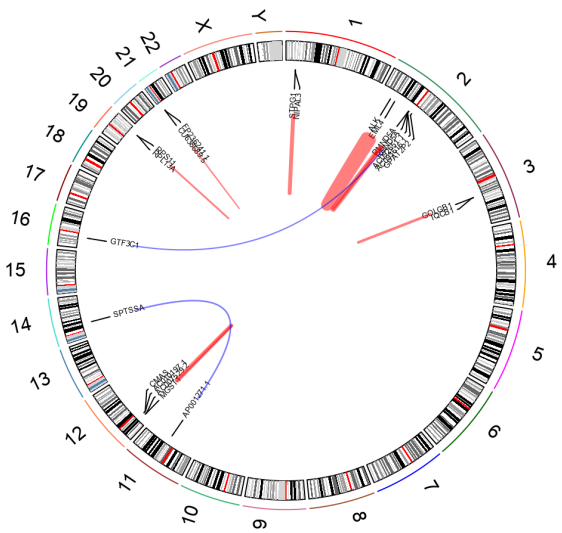
Five lines of code produce simple report, which includes a circus-plot and a table of fusions:

```
# Install chimeraviz library (done once)
source("https://bioconductor.org/biocLite.R")
biocLite("chimeraviz")

# Load chimeraviz library
library(chimeraviz)

# Read STAR-fusion output
fusions <- importStarfusion("star-fusion.fusion_predictions.abridged.tsv", "hg38", 20)

# Generate quick report with circus plot
createFusionReport(fusions, "chimeraviz_report.html")
```



What fusion drives cancer growth in SRR3534924 sample?

If you have free time at the end of this session: try making *Chimeraviz* gene-pair plot for the driver-fusion.

6. Calculating transcript counts with *Salmon*

Arguably, **gene expression analysis** could be perceaved as the main application of RNA-Seq (of course, people focused on fusion detection will disagree :) The first step in the expression measurement is counting reads overlapping the genes (discussed in more details during the lecture). Traditionally, read count was based on genomic alignments, as implemented in *Cufflincs* or *RSEM*. However, alignment-free methods, such as *Kallisto* or *Salmon*, are becoming increasingly popular.

In this session we will use *Salmon*. In fact, instead of relying on another tool for genome-alignment, *Salmon* prforms its own fast and acurate "quasi-mapping" to transcriptome:
https://combine-lab.github.io/salmon/

Preparing transcriptome file for quasi-mapping

Human transcriptome could be obtained/compiled from publicly avaialble sources, such as Ensembl. For this tutorial a combination of cDNA and non-coding RNA sequences was choosen as the transcriptome:

```
wget ftp://ftp.ensembl.org/pub/path/to/cdna/Homo_sapiens.GRCh38.cdna.all.fa.gz
wget ftp://ftp.ensembl.org/pub/current_fasta/homo_sapiens/ncrna/Homo_sapiens.GRCh38.ncrna.fa.gz
zcat Homo_sapiens.GRCh38.cdna.all.fa.gz Homo_sapiens.GRCh38.ncrna.fa.gz > Homo_sapiens.GRCh38.rna.fa.gz
```

This has been done already, you do not need to repeat this step. You can find the transcriptome file in the **~/Documents/RNA-Seq/resources/annotations** folder.

Prepare index

Like most of the other tools performing "alignment-like" tasks, Salmon prepares index for the reference. In this case it prepares index for the reference transcriptome:

```
"${salmon}" index \
 -t "/path/to/Homo_sapiens.GRCh38.rna.fa.gz" \
 -i "/path/to/salmon_index"
```

Again, there is a script **salmon_index.sh**, which you can use as an example, or just review and run it to create the index. The intended location of the index is **~/Documents/RNA-Seq/resources/salmon_index**.

Run Salmon

The quantitation step could be performed using **salmon quant** command:

```
"${salmon}" quant \
--index "/path/to/salmon_index" \
--libType A \
--mates1 "fastq1}" \
--mates2 "${fastq2}" \
--output "${out_folder}" \
--threads 10
```

Use **salmon quant –help-reads** for informatin about the options. The intended output folder is **~/Documents/RNA-Seq/data/lung_cancer/expression**. Script **salmon_quant.sh** is provided as an example or to run the quantification.

Have a look at the content of the output folder after the run. Amongst the other files, there will be a **quant.sf** file. This is the result of Salmon quantification. Have a quick look at the content of this file (hint: you may use **head** command in the terminal).

7. Differential gene expression with *DESeq2*

Detecting differentially expressed genes is an important step in the gene expression analysis. The top differentially expressed genes are often used for molecular classification of cancer, to evaluate prognosis or to predict response to treatment.

There are many statistical approaches for detecting differential expression. We will use the approach implemented in **DESeq2** R package. It assumes Negative Binomial distribution of the counts. This assumption is based on the fact that empirical counts data are closer to the Negative Binomial than, for instance, to Normal or Poisson distributions.

In addition to the sophisticated mathematical methods, *DESeq2* package provides an implementation of these methods and a number of convenience functions for data propcessing and assessment:
https://bioconductor.org/packages/release/bioc/vignettes/DESeq2/inst/doc/DESeq2.html

Source data

quant.sf files were pre-calculated for 3 Tumour-Normal pairs of renal cancer. The data was generated by CAGEKID consortium (https://www.cng.fr/cagekid/index.html). We will use these pre-calculate Salmon counts to detect genes differentially expressed between renal carcinoma and normal tissue.

The data are located in **~/Documents/RNA-Seq/data/kidney_cancer/salmon_counts** folder. The patients IDs are *LT344*, *LR364* and *LR365*. Sufixes *N* and *T* indicate to Normal and Tumour sample respectively.

R-markdown script

For the rest of the workshop we switch from Shell scripts to R environment. The code for our entire Differential Expression analysis is provided in **DESeq2_analysis.Rmd** script. The key fragments of this script will be discussed in the text below. However, plenty of additional information is provided within the script. Review and run the corresponding part of the script along with reading this text.

The script is written using **R-markdown**. This allows organising code into convenient **chunks**, adding formatted comments between the chunks and, most importantly, it allows to **knit** a detailed report of the preformed analysis. R-markdown is a widely used tool facilitating **reproducible research** practices in R environment.

Prepare R environment and accessory data

The **Start chunk** of the script cleans the environment (just in case) and shows how the required packages were installed. The installation lines are commented because it needed to be done only once. Then we load **dplyr** and **ggplot2**: these are general libraries commonly used for data handling and plotting. The specialised libraries are loaded later: in the chunks, where they are used.

The following accessory data should be prepared for **DESeq2** analysis:

List of the ".sf" files, which contain *Salmon* counts

```
# Get list of folders with salmon counts
counts_folder=paste(base_folder, "data/kidney_cancer/salmon_counts", sep="/")
salmon_folders <- list.dirs(counts_folder, recursive = FALSE)

# Make list of sf files
salmon_sf_files <- paste(salmon_folders, "quant.sf", sep="/")

# Assign names to the vector of files
names(salmon_sf_files) <- basename(salmon_folders)
```

Data frame with samples description

```
# Prepare vectors with data
run <- c("LR344N", "LR344T", "LR364N", "LR364T", "LR365N", "LR365T")
condition <- c("N","T","N","T","N","T")
patient <- c("LR344", "LR344", "LR364", "LR364", "LR365", "LR365")

# Combine vectors to data frame
samples.df <- data.frame(run, condition, patient)

# Convert condition column to factor
samples.df$condition <- as.factor(samples.df$condition)

# Add rownames
rownames(samples.df) <- samples.df$run

# Check result
samples.df
```

```
##           run condition patient
## LR344N LR344N         N   LR344
## LR344T LR344T         T   LR344
## LR364N LR364N         N   LR364
## LR364T LR364T         T   LR364
## LR365N LR365N         N   LR365
## LR365T LR365T         T   LR365
```

Table linking transcript-ID to gene-ID

```
# Load required library
require(rtracklayer) # contains readGFF function: see ?readGFF

# Read annotations file
annotation_file <- paste(base_folder, "resources/annotations/Homo_sapiens.GRCh38.89.gtf", sep="/")
gene_annotation.df <- readGFF(annotation_file)
```

```r
# Make table linking transcripts to genes
trans2genes.df <- gene_annotation.df %>%
  filter(type == "transcript") %>%
  select(transcript_id, transcript_version, gene_id)

# Combine transcript ID and version
trans2genes.df <- trans2genes.df %>%
  mutate(transcript_id_version=paste(transcript_id, transcript_version, sep=".")) %>%
  select(transcript_id_version,gene_id)

# Check result
head(trans2genes.df)
```

```
##   transcript_id_version         gene_id
## 1     ENST00000456328.2 ENSG00000223972
## 2     ENST00000450305.2 ENSG00000223972
## 3     ENST00000488147.1 ENSG00000227232
## 4     ENST00000619216.1 ENSG00000278267
## 5     ENST00000473358.1 ENSG00000243485
## 6     ENST00000469289.1 ENSG00000243485
```

Table linking gene-ID to gene-name

```r
# Make data frame
geneId2Name.df <- gene_annotation.df %>%
  select(gene_id, gene_name) %>%
  arrange(gene_name) %>%
  distinct()

# Copy gene IDs to rownames
rownames(geneId2Name.df) <- geneId2Name.df$gene_id

# Check result
head(geneId2Name.df) # NB: gene names are not unique !
tail(geneId2Name.df)
```

```
##                         gene_id gene_name
## ENSG00000252830 ENSG00000252830   5S_rRNA
## ENSG00000276442 ENSG00000276442   5S_rRNA
## ENSG00000274408 ENSG00000274408   5S_rRNA
## ENSG00000274059 ENSG00000274059   5S_rRNA
## ENSG00000277313 ENSG00000277313       7SK
## ENSG00000275933 ENSG00000275933       7SK
```

```
## ...
```

```
##                         gene_id  gene_name
## ENSG00000229956 ENSG00000229956 ZRANB2-AS2
## ENSG00000121903 ENSG00000121903    ZSCAN20
## ENSG00000162415 ENSG00000162415     ZSWIM5
## ENSG00000203995 ENSG00000203995     ZYG11A
## ENSG00000162378 ENSG00000162378     ZYG11B
## ENSG00000036549 ENSG00000036549       ZZZ3
```

At this point all the accessory tables have been prepared.

Import Salmon counts into R

tximport R package provides a convenient way of importing transcript abundance data from multiple upstream applications, including *Salmon*, *Kallisto*, *RSEM* and others:
https://bioconductor.org/packages/release/bioc/vignettes/tximport/inst/doc/tximport.html

Loading *Salmon* data can be preformed in the following way:

```
# Load required libraries
require(tximport) # ?tximport
require(rjson) # required for soem tximport features

# Read and convert salmon counts
gene_counts.ls <- tximport(salmon_sf_files, type="salmon", tx2gene=trans2genes.df)

# Check results
head(gene_counts.ls$counts)
```

```
##                    LR344N    LR344T     LR364N    LR364T    LR365N     LR365T
## ENSG00000000457  454.3549  621.4804  429.68625  642.2974  654.4376   280.6368
## ENSG00000000460  128.3026  277.5019  105.65094  334.8424  160.7700  2711.3892
## ENSG00000000938  140.0000  992.0003   71.99995  674.9995  206.0005   349.0004
## ENSG00000000971 2255.8532 3603.0045 1536.92162 9826.9728 3812.2033 25005.4770
## ENSG00000001460  541.3224  380.3891  274.47184  496.8335  357.9931   264.4049
## ENSG00000001461 1121.4062 1220.1628  754.08937 2521.9820 1296.2460   945.2384
```

Note that **tximport** places data into a *list* object. Also it aggregates the transcript-level counts into genes.

Later the **tximport list** will be transformed into **DESeq-DataSet** object that is needed to perform the differential expression tests.

Explore imported data

Gene expression data assessment and QC could be done using **hierarchical clustering** or **PCA** analysis. Note that this text illustrates only selected steps of data QC: review and run the accompanying script for more details.

Within our **DESeq2** workflow the data assessment and QC could be performed at two points:

- immediately after the data import into the **tximport list**
- at a later stage: after importing data into **DESeq-DataSet** object

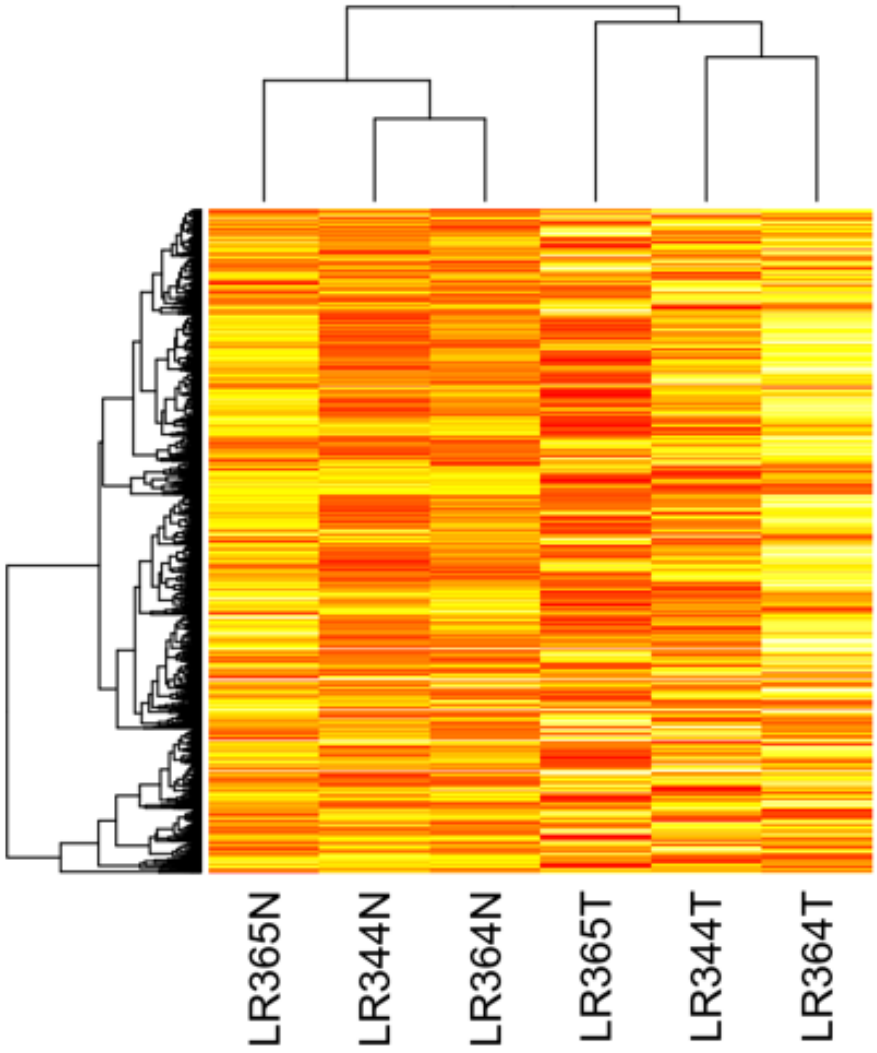
For teaching purposes, we will do data assessment at both of these steps and compare the resulsts.

Hierarchical clustering with heatmap

```
# Extract and log-transform normalised expression values
expr.mx <- gene_counts.ls$abundance
expr.mx <- log2(expr.mx + 1) # +1 in case if some expressions are 0

# Filter low expressed genes
low_expressed_genes <- mean_expr < 1
expr.mx <- expr.mx[! low_expressed_genes, ]

# Plot heatmap
heatmap(expr.mx, labRow=NA)
```

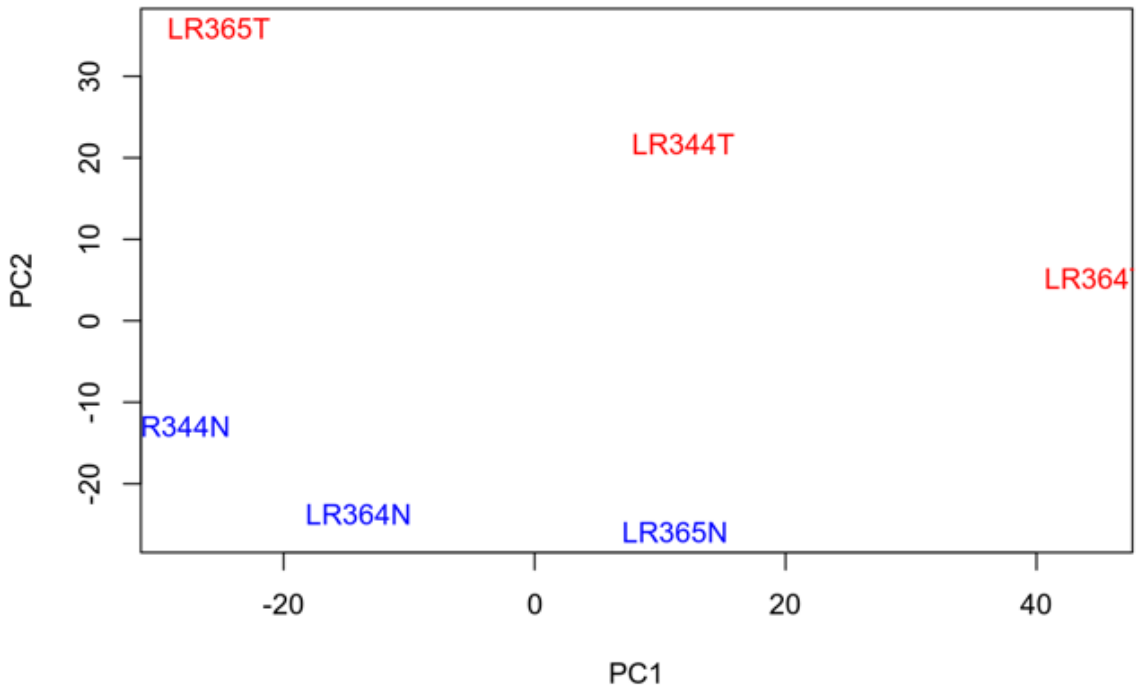


PCA plot

```
# Calculate PCA
expr.pca <- prcomp(t(expr.mx),
                   center = TRUE,
                   scale = TRUE)

# Extravct matrix of PCs
pc.mx <- expr.pca$x

# Plot PC1 vs PC2
PC1 <- pc.mx[,1]
PC2 <- pc.mx[,2]
plot(PC1, PC2, type="n")
text(PC1, PC2, labels = row.names(pc.mx),
    col=c("blue","red","blue","red","blue","red"))
```



Both techniques (**hierarchical clustering** and **PCA** analysis) showed separation of tumour and normal samples, even using the entire set of expressed genes. This confirms a good quality of data and suggests presence of strong differentially expressed genes, which could be detected in downstream analysis.

Read data to *DESeq-DataSet* object

At this step we will read data into the **DESeq-DataSet** object and filter data, re-using the list of low-expressed genes from the previous step (more details about selecting the low-expressed genes are shown in the accompanying script).

```
# Load required library
require(DESeq2)
#?DESeqDataSetFromTximport

# Read data into DESeq2 dataset
dds <- DESeqDataSetFromTximport(gene_counts.ls, colData = samples.df, design = ~condition)

# Remove low-expressed genes from DESeq2 dataset object
dds <- dds[! low_expressed_genes, ]

# Check result
dds
```

```
## class: DESeqDataSet
## dim: 2065 6
## metadata(1): version
## assays(5): counts avgTxLength normalizationFactors mu cooks
## rownames(2065): ENSG00000000457 ENSG00000000460 ... ENSG00000283761 ENSG00000283773
## rowData names(21): baseMean baseVar ... deviance maxCooks
## colnames(6): LR344N LR344T ... LR365N LR365T
## colData names(3): run condition patient
```

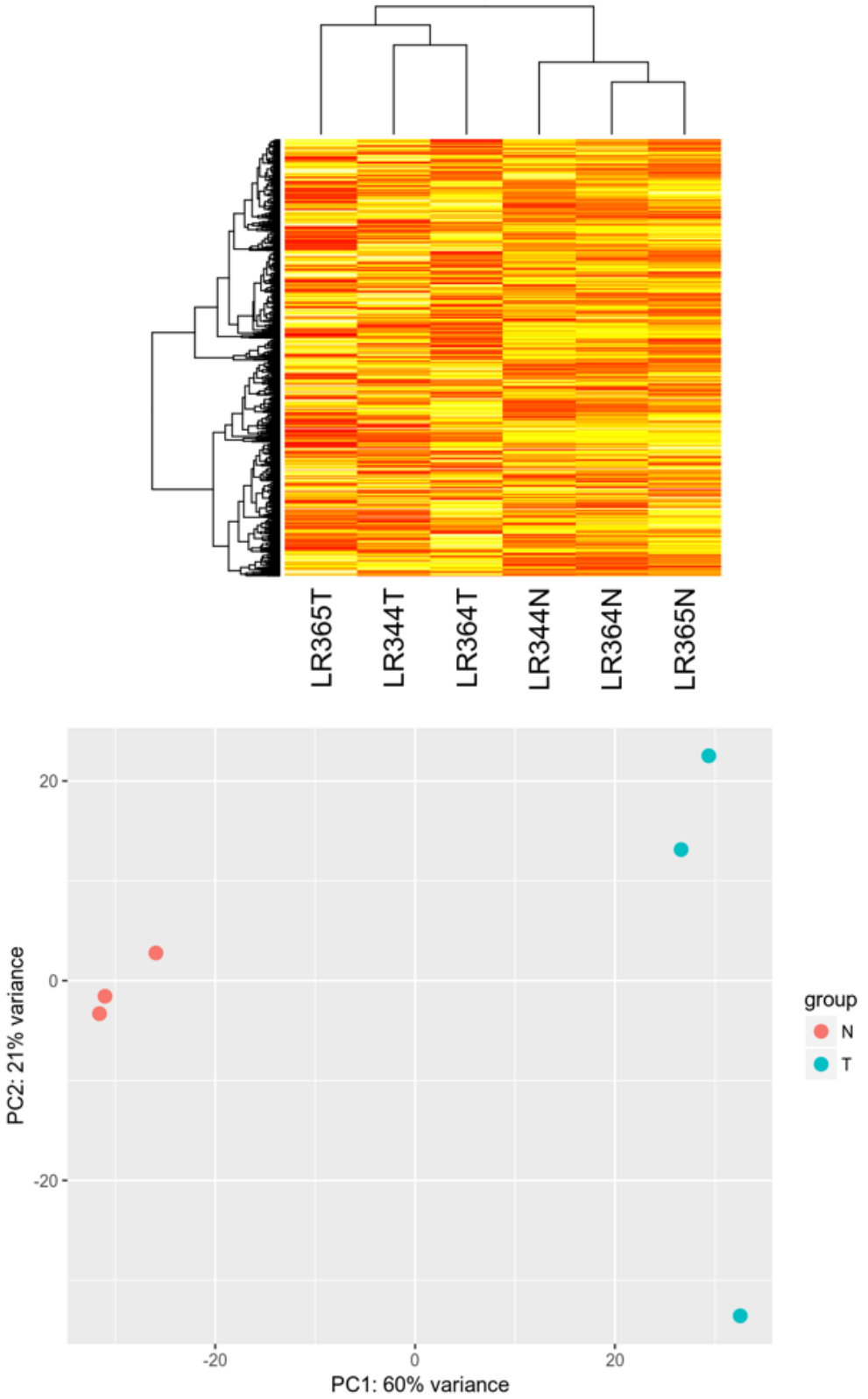
Explore data within *DESeq-DataSet* object

The data have just been placed into **DESeq-DataSet** object. No tests for differential expression have yet been done. However, we may already use some convenience tools provided by *DESeq2* package for data assessment within **DESeq-DataSet** object. These tools include advanced normalisation functions (such as **varianceStabilizingTransformation**) and an internal function to make PCA plot on the vst-transformed data.

```
# Apply VST (for visualising)
vsd <- varianceStabilizingTransformation(dds)

# Heatmap after VST
heatmap(assay(vsd), labRow=NA)

# PCA after VST (built-in DESeq2 function)
plotPCA(vsd)
```



Consistent with the previous assessment, both methods show a good separation of normal and tumour samples after VST. This confirms good qaulity of data and suggests presence of differentially expressed genes.

Calculate differentially expressed genes

Finally, we are in a position to make tests for differential expression.

The testing algorithm includes estimation of effective library size, genes dispesrion and including these parameters into Negative Binomial Generalized Linear Model:
https://genomebiology.biomedcentral.com/articles/10.1186/s13059-014-0550-8
https://bioconductor.org/packages/release/bioc/vignettes/DESeq2/inst/doc/DESeq2.html#theory

Luckily, all this mathematical complexity is wrapped into a single simple function call:

```
dds <- DESeq(dds)
```

The test results have now been added to the **DESeq-DataSet** object.
The table with Fold-Change and adjusted P-values can now be extracted from the **DESeq-DataSet** object using *results()* function:

```
result.df <- as.data.frame(results(dds))
head(result.df)
```

```
##                  baseMean log2FoldChange     lfcSE       stat       pvalue        padj
## ENSG00000000457  516.8618     -0.1215062 0.5610834 -0.2165564 0.8285540686 0.945858318
## ENSG00000000460  722.2178      3.3731988 0.8638281  3.9049422           NA          NA
## ENSG00000000938  370.1845      2.0667952 0.5567747  3.7120853 0.0002055587 0.002680444
## ENSG00000000971 8052.6020      2.3867098 0.7483717  3.1892038 0.0014266527 0.013044657
## ENSG00000001460  377.8124     -0.1443089 0.5027592 -0.2870338 0.7740864756 0.933888418
## ENSG00000001461 1265.1619      0.5165444 0.5161500  1.0007642 0.3169408247 0.657948099
```

Err ... It's exciting to know that ENSG00000000938 has adjusted P-value of 0.002 ...
Still, it seems that some finishing touches might be helpful:

```
# Add gene_id to colimns
result.df <- cbind(gene_id=rownames(result.df), result.df)

# Remove genes with no p-value
uninformative_genes <- is.na(result.df$padj)
result.df <- result.df[!uninformative_genes,]

# Add gene names and order by p-value
result.df <-left_join(result.df, geneId2Name.df, by="gene_id") %>%
  select(gene_id, gene_name, baseMean, log2FoldChange,  padj) %>%
  arrange(padj, desc(log2FoldChange))

# Copy gene names to rownames
rownames(result.df) <- result.df$gene_name

# Check result
head(result.df)
```

```
##                 gene_id gene_name   baseMean log2FoldChange         padj
## NPHS2   ENSG00000116218     NPHS2  3498.4932      -9.798824 1.932462e-59
## FAM151A ENSG00000162391   FAM151A 14583.8887      -8.725494 1.283474e-44
## DIO1    ENSG00000211452      DIO1  5840.0166      -8.609517 9.896296e-41
## MCOLN3  ENSG00000055732    MCOLN3   389.4464      -5.605388 2.656679e-23
## VTCN1   ENSG00000134258     VTCN1   397.4958      -6.727450 3.730972e-22
## KCNJ10  ENSG00000177807    KCNJ10   784.2833      -6.443739 6.421047e-22
```

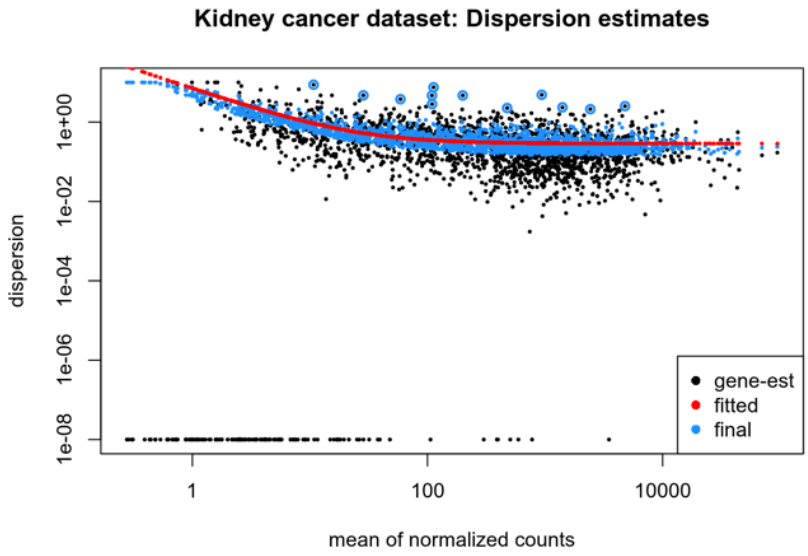
It looks much better now !

Visualising results of differential expression analysis

In addition to the table with p-values and fold-changes, **DESeq2** package provides several functions to visualise results.

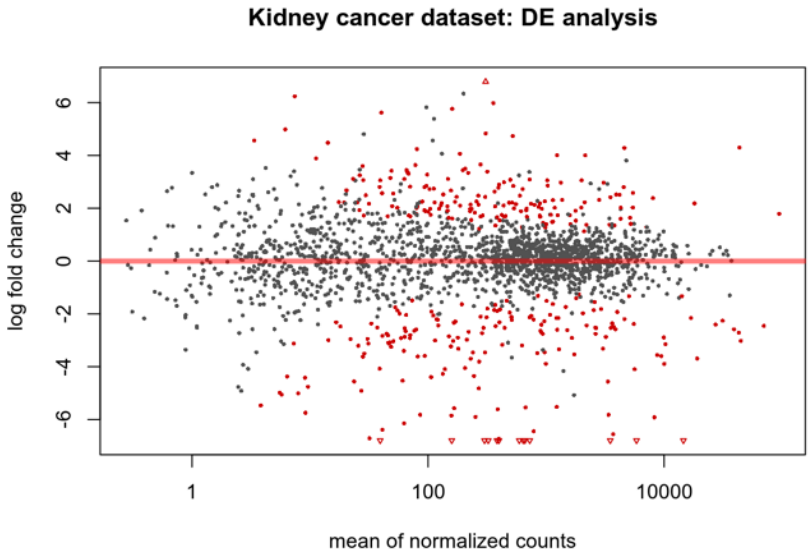
Dispersion estimates

```
plotDispEsts(dds, main="Kidney cancer dataset: Dispersion estimates")
```



MA plot

```
plotMA(dds, main="Kidney cancer dataset: DE analysis")
```

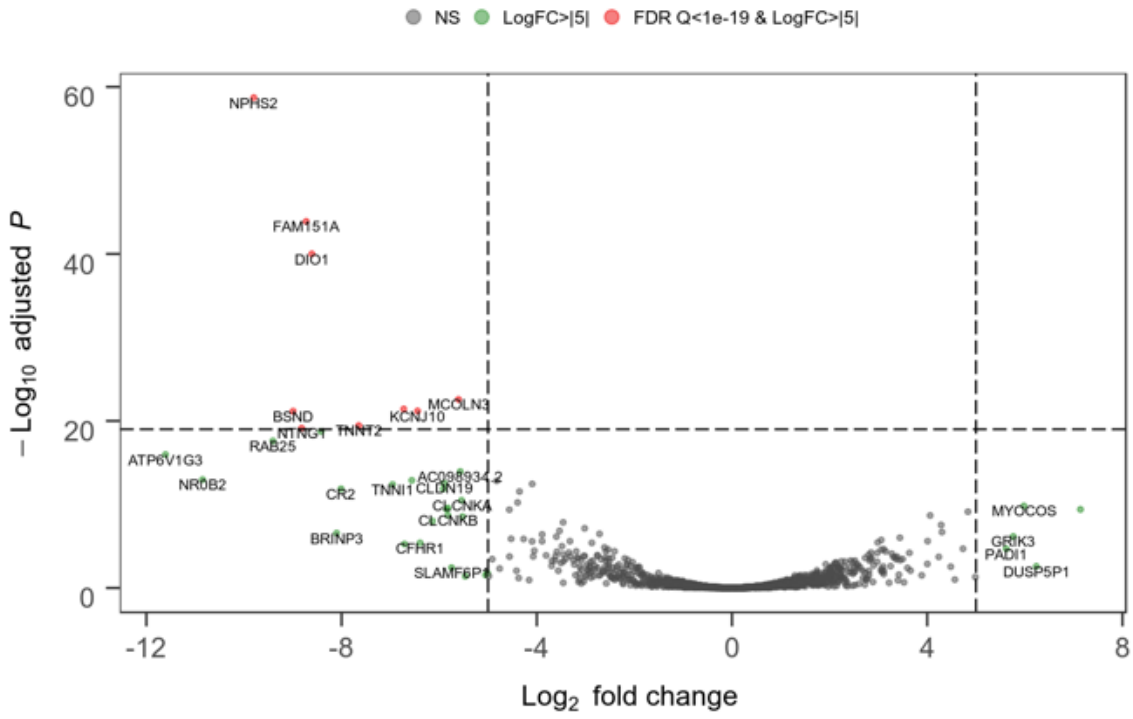


Volcano plot

Surprisingly, it seems that **DESeq2** does not provide function for a *Volcano plot*. However, a quick and nice *Volcano plot* can be generated from **DESeq2** results using a function, available at this URL: https://github.com/kevinblighe/EnhancedVolcano

```
# Load plotting function
source("EnhancedVolcanoDESeq2.R")

# Make plot
EnhancedVolcanoDESeq2(result.df, AdjustedCutoff=10E-20, LabellingCutoff=0.05, FCCutoff=5.0)
```



Focused analysis of top significant genes

Finally, a focused analysis of the most differentially expressed genes may give clues for selecting candidates for downstream analyses, e.g. for inclusion into the gene expression signatures.

Select genes by p-value and fold-change

```
# Set cut-offs for top genes
log_fc_cutoff <- 5
adj_p_cutoff <- 0.001

# Extract results for top genes
top_genes.df <- result.df %>%
  filter(abs(log2FoldChange) > log_fc_cutoff, padj < adj_p_cutoff)
```

```
# Print top genes
top_genes.df
```

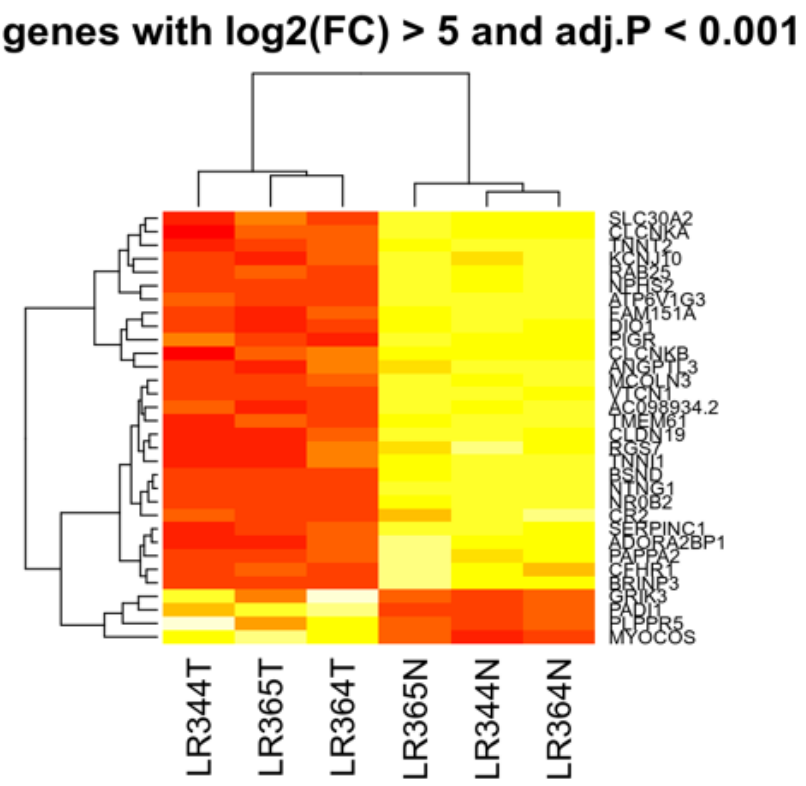
```
##            gene_id  gene_name    baseMean log2FoldChange         padj
## 1  ENSG00000116218      NPHS2  3498.49324      -9.798824 1.932462e-59
## 2  ENSG00000162391    FAM151A 14583.88874      -8.725494 1.283474e-44
## 3  ENSG00000211452       DIO1  5840.01659      -8.609517 9.896296e-41
## 4  ENSG00000055732     MCOLN3   389.44640      -5.605388 2.656679e-23
## 5  ENSG00000134258      VTCN1   397.49580      -6.727450 3.730972e-22
## 6  ENSG00000177807     KCNJ10   784.28326      -6.443739 6.421047e-22
## 7  ENSG00000162399       BSND   383.66461      -8.993231 6.862738e-22
## 8  ENSG00000118194      TNNT2   727.26329      -7.649031 3.566098e-20
## 9  ENSG00000162631      NTNG1   301.16639      -8.819621 7.418327e-20
## 10 ENSG00000116183     PAPPA2   661.52114      -8.420240 1.810247e-19
## 11 ENSG00000132698      RAB25   323.74983      -9.406231 2.417928e-18
## 12 ENSG00000151418   ATP6V1G3   592.94906     -11.606765 1.060404e-16
## 13 ENSG00000234996 AC098934.2   166.42232      -5.568136 1.128515e-14
## 14 ENSG00000131910      NR0B2   158.85090     -10.848179 1.058795e-13
## 15 ENSG00000132855    ANGPTL3  3708.79307      -6.560491 1.292814e-13
## 16 ENSG00000164007     CLDN19   252.88165      -5.898904 2.851007e-13
## 17 ENSG00000159173      TNNI1   640.02290      -6.955768 4.019438e-13
## 18 ENSG00000162896       PIGR  8287.39239      -5.916263 1.340396e-12
## 19 ENSG00000117322        CR2   398.51388      -8.011660 1.340396e-12
## 20 ENSG00000186510     CLCNKA   670.13466      -5.542289 3.134492e-11
## 21 ENSG00000283683     MYOCOS   357.52627       5.980955 1.506737e-10
## 22 ENSG00000143001     TMEM61    85.71740      -5.819244 2.586958e-10
## 23 ENSG00000182901       RGS7   157.89409      -5.851889 3.694829e-10
## 24 ENSG00000117598     PLPPR5   307.45508       7.138759 3.955272e-10
## 25 ENSG00000184908     CLCNKB  3390.11996      -5.820860 2.031877e-09
## 26 ENSG00000158014    SLC30A2  1230.27754      -5.522122 2.868532e-09
## 27 ENSG00000117601   SERPINC1    62.64627      -6.137652 9.567549e-09
## 28 ENSG00000162670     BRINP3    39.27424      -8.101568 2.710679e-07
## 29 ENSG00000163873      GRIK3   160.06775       5.762889 6.890060e-07
## 30 ENSG00000244414      CFHR1    41.03793      -6.392451 3.935346e-06
## 31 ENSG00000236136  ADORA2BP1    31.90155      -6.710285 5.621796e-06
## 32 ENSG00000142623      PADI1    40.22486       5.619188 2.017450e-05
```

Heatmap of top significant genes

```
# Get expression values for the top genes
top_genes_exprs.mx <- expr.mx[top_genes.df$gene_id, ]

# Change gene IDs to gene names
rownames(top_genes_exprs.mx) <- top_genes.df$gene_name

# Plot heatmap
num_genes <- nrow(top_genes.df)
main <- paste(num_genes, "genes with log2(FC) >", log_fc_cutoff, "and adj.P <", adj_p_cutoff, "\n")
par(oma=c(0,0,3,0)) # make space for the main header
heatmap(top_genes_exprs.mx, main=main)
```



Plot normalised counts for a selected gene

```
gene_id <- "ENSG00000158014"
gene_name <- geneId2Name.df[gene_id,]
plotCounts(dds, gene=gene_id, main=gene_name)
```