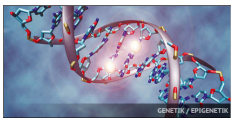
Epigenetic Data Processing Course, WS 2019/20
Prof. Dr. Jörn Walter
Abdulrahman Salhab, Michael Scherer, Kathrin Kattler
Department of Genetic/Epigenetics & Saarland University



UNIVERSITÄT DES SAARLANDES

RNA-seq

1 Get to know the system

See general task sheet

2 Running the RNA-seq pipeline

1. **Organize your raw data.** `grape-nf`[1] is not dependent on were the files are located, but it is still good practice to keep an organized workspace:
   a) create a `fastq` folder in the data folder
   b) as group 1|2 you will analyze all liver|kidney samples
   c) create symbolic links to the relevant files in `/vol/data/fastq/RNA-seq`

2. **Generate the sample sheet.** There are many ways to generate text files on the command line. One opinion is that the more you can document as commands and the less you actually have to edit the actual file, the better. Feel free to generate the file however you want, but here is one alternative:
   a) list the full path to your files (e.g. `ls $PWD/data/fastq/*`, see list of bash variables [2]).
   b) We will now use the beginning of each of the files as IDs. `sed`[3] is a convenient tool to make regex[4] searches on the command line:

```
ls $PWD/data/fastq/* \
    | sed -re "s/.*\/(.*)_(.*).fastq.gz/\1_\2/g"
```

   c) Successfully extracting the id, we can now build up the file by extending the command:

```
ls $PWD/data/fastq/* \
        | sed -re "s/.*\/(.*)_(.*).fastq.gz/\1_\2\t\0/g"
```

   Make sure the fourth and fifth columns are correct and redirect the output to a file. Keep organized!

   (Again, just one way of many).

3. **Start the pipeline.** `grape-nf` is based on a pipeline engine/language called Nextflow[5]. As described on the `grape-nf` site it can be easily installed. We will come back to the details, but in order to get the pipeline started, we will do it slightly different:
   a) Create a work folder for this step and enter it
   b) Load the Conda environment with the necessary executables:

```
source /vol/data/conda/base/bin/activate /vol/data/conda/envs/core
```

   c) Test if you have access to `nextflow`:

```
which nextflow
```

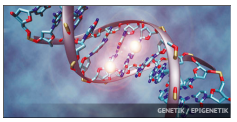
   d) In order to make full use of the system we need a configuration file:

```
cp /vol/data/pipelines/nextflow/nextflow.config .
```

   Inspect the file and make sure there are no surprises.

   e) We also need reference files. Today we will bypass index generation to speed things up somewhat.

[1] https://github.com/guigolab/grape-nf
[2] https://www.gnu.org/software/bash/manual/html_node/Bash-Variables.html
[3] http://www.grymoire.com/Unix/Sed.html
[4] https://regex101.com/
[5] https://www.nextflow.io/




```
    --genome /vol/data/reference/mm10/rna-seq/mm10_no_alt_analysis_set_ENCODE.fasta
--annotation /vol/data/reference/mm10/rna-seq/gencode.vM2.annotation.gtf
--genomeIndex /vol/data/reference/mm10/rna-seq/STAR_index
```

f) To execute the pipeline you need to complete this command:

```
NXF_HOME=/vol/data/pipelines/nextflow nextflow run grape-nf \
        -with-singularity \
        --with-ref-prefix - \
        --steps mapping,bigwig,quantification \
        --genomeIndex ...
```

Here, **NXF_HOME** is most important to include. Secondly, make sure you refer to the sample sheet you generated.

g) Create another window in your screen session and check that everything is running with `htop` (`u` to only view your jobs)

3 Quality Control, step 1: FastQC

After starting the pipeline, we will now inspect data quality of the input sequence data. We will use two tools for this purpose (`fastqc` and `multiqc`).

1. Create a new conda environment at `/vol/<yourdisk>/<yourgroup>/conda/`.
2. Install `fastqc` and `multiqc`.
3. Execute `fastqc` for the samples you should analyze and store the results in a separate folder (e.g. called `qc`).
4. Execute `multiqc` in the created folder and explore the results. Do you observe issues specific to RNA-seq? What are the read lengths for the samples?

4 Gather pipeline output

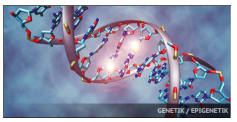
`grape-nf` is meant for large scale processing of data. One drawback with this is that the output folder is a bit convoluted. To our help, as described on the repository page, there is the pipelines.db file.

1. Browse the output folder. If nothing else was specified, the folder named `work` will contain most of the results.
2. Look at the files `pipeline.db` and `trace.txt`. Did the pipeline finish successfully? Or was there any problem?
3. Let's create a more organized folder structure with sample-wise folders with links to all output files. This is another task with multiple solutions, and below is a suggested start:

```
while read sample runid file type content readType readStrand; do
    outputFolder="The $sample sample has a file of type $type"
    echo $outputFolder
done < pipeline.db
```

Perhaps it can be modified to create folders and links...

4. You'll have to share your results with the other RNA-seq group. More specifically, the RSEM output files (`*.results`) need to be made available to the others. To keep consistent, we require the following folder structure:




```
/vol/data/share/RNA-seq
                    |
                    |__ rnaseq1
                    |     |__ RSEM
                    |
                    |__ rnaseq2
                    |     |__ RSEM
```

5 Quality Control, step 2: RNA-seQC

With the files a bit more easily accessible we will create a RNA-seq specific QC-report with a software called `RNA-seQC`.

1. Create a work folder
2. Create a sub folder and link the genomic bam files
3. Create a new conda environment and make sure you have `samtools`, `bwa`, and `RNA-seQC` installed.
4. You will need an index for the bam files. You have two possibilities: (i) Use the generated indeces (.bam.bai files) in the `work` folder or (ii) Use `samtools` together with a `for` loop or `xargs`.
5. You will need an additional reference file and a new output folder named `out`.

```
referenceAnnotation=/vol/data/reference/mm10/rna-seq/gencode.vM2.annotation.gtf
for bam in *.bam
do
 rnaseqc \
 $referenceAnnotation \
 $bam \
 out
done
```

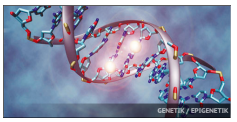
   The complete run will take a while... Create a new screen and continue with the next assignment?

6. After the run has completed, there will be sample specific files. We will focus on the `.tsv` files. Is the alignment efficiency similar across all the samples? Are there further statistics that you find interesting? It might be worthwhile to create a plot or two describing these statistics across the different samples (**Hint:** the `ggplot2` R package might help).

6 Exploratory Analysis using IGV

`IGV` lets you browse genomic tracks. Normally it goes smooth, but the connection to the cluster may hamper this. Thus, we recommend to install IGV locally (`https://software.broadinstitute.org/software/igv/download`) and transfer the data files from the cluster for visualization.

1. New work folder
2. data folder with big wig (`.bw`) files
3. Load the tracks, organize them and make it more visually appealing.
4. Save a session file for later
5. Find a favorite gene and make a screen shot.



7 Differential analysis with edgeR

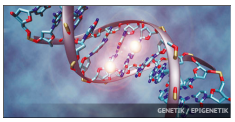
In our dataset, we would like to compare two tissues with one another to define tissue-specific expression. Your partner group processed the second half of the samples, so get in touch with them to use their datasets. We will contrast liver versus kidney tissues. For this, we will make use of the bioconductor package `edgeR`[6] and the companion package `txImport`[7].

1. New work folder... and a results subfolder
2. For this we will need the `RSEM` output files
3. Although R packages can be installed inside R, the sanity of our conda environment is better preserved if we continue installing with conda. You need the packages mentioned above to begin with.
4. The edgeR user guide is your friend and in principle we will walk through the process outlined in the quick-start with some variation.
5. Load the data
   a) Create a data.frame with the following columns. One way is to create a sample sheet in excel and load this, but as we have informative file names, we can do this in `R` directly.
      - `genes` – path to gene results files (Hint: `list.files`)
      - `isoforms` – same as genes, but isoform results
      - `sampleID` – something that can be used to identify the samples (Hint: `stringi` or some other string manipulation package/function)
      - `timepoint` – The time point of the sample (numeric)
      - `replicate` – Does the file come from the first or second replicate
      - `group` – either the liver or the kidney samples

      For the downstream functions to work, make sure `genes` and `isoforms` are characters, not factors.
   b) Load data with `tximport`. RSEM reports some genes with zero length, this will cause an error. Exclude these.
6. Create a design matrix based on `group` and do the differential analysis with the quasi-likelihood F test method. Create a `data.frame` containing results for all genes. This will be your base comparison.
7. At the moment, you just have an ID for each gene. A first step to make the results more accessible is to annotate with the gene name and short description. For this, we will use the `biomaRt` package (==>install). This allows you to connect to Ensembl's BioMart[8], which contains a wealth of information.
   a) Create an annotation table:

```
biomartConnection<-useMart(
    "ensembl",
    dataset="mmusculus_gene_ensembl"
)
geneAnnotation<-getBM(
    attributes=c(
        "ensembl_gene_id", "external_gene_name", "description"
    ),
    mart=biomartConnection
)
```

[6] https://bioconductor.org/packages/release/bioc/html/edgeR.html
[7] https://bioconductor.org/packages/release/bioc/html/tximport.html
[8] https://www.ensembl.org/biomart/martview/



This command is quite selective, but it is easy to see how this can be changed to get additional information. To get a feeling for what you have access to, list all attributes of the `biomartConnection`.

b) Combine `geneAnnotation` with your top table. (Hint: `merge` and `ensembl_gene_id`)

8. Write the annotated table to disk.
9. Create a bedgraph file[9] with $-log_{10}$ scaled FDR-values in the value position (add a small value (1e-10) if zero values are present). Write to disk with a suffix that is compatible with `IGV` (The header and track line are optional.)
10. Write a table with CPM values to disk. Make sure the column names are informative!
11. For the handover, you will have to summarize the CPM values in the two groups. (**Useful functions:** `apply`, `mean`). You should export all the genes using the following columns in a BED file `Chromosome, Start, End, -log10P, logFC, meanCPMKidney, meanCPMLiver`.

Keep this session running, we just need to make sure everything is ready for tomorrow before we continue.

8 Prepare handover

You now have all results needed for tomorrow.

1. Create a folder for your group in `/vol/data/share/RNA-seq/`.
2. Copy (**not** link) and organize the following files:
   - Bigwig files
   - Differential table as a BED file. The following columns should be included: `Chromosome, Start, End, -log10P, logFC, meanCPMKidney, meanCPMLiver, ID`

   How we envisage the folder structure:

```
/vol/data/share/RNA-seq
                        |
                        |__ rnaseq1
                        |     |__ DEGs
                        |     |__ signals
                        |
                        |__ rnaseq2
                              |__ DEGs
                              |__ signals
```

3. Prepare one slide explaining how you organized the files, the content, and how they can be used.
4. Make sure that only people of your group can modify the file, but all can read it (Hint: `chmod`)

9 Explore differential expression using IGV

1. Expand your IGV session with the differential results
2. Browse the top scoring genes. Are the results convincing? Screen shots for the presentation?

[9] https://genome.ucsc.edu/goldenPath/help/bedgraph.html

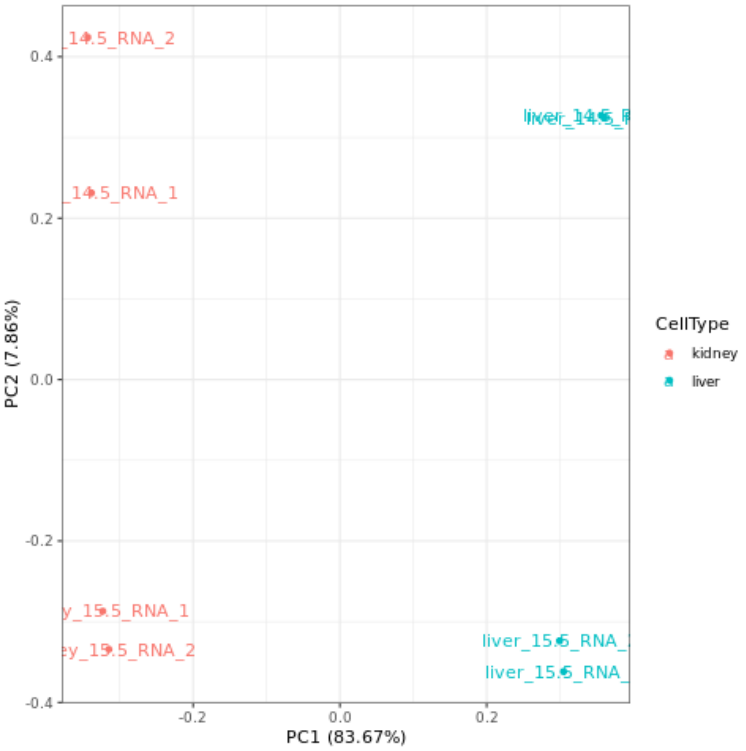


Figure 1: PCA plot generated with `prcomp` and `ggpfortify`

10 Exploratory Analysis using edgeR

10.1 Principal component analysis

This is probably something you would normally do before your differential analysis. Today's priority was making sure that the data was ready for tomorrow, but in general, this is a good thing to do early.

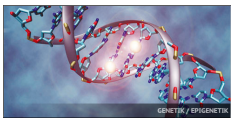
1. We will base this on the CPM values, but as the RNA-seq protocol contains some amplification steps, the strongly expressed genes are overrepresented. One way around this is doing a *log* transform ($y = log_2(x+1)$). Why do we use base 2 and 1 as pseudo count here (cf. bedgraph creation above)?
2. Calculate the PCA with `prcomp`.
3. Create a Scree plot and/or look at the summary of the object for the weight of the different components. (**Useful functions:** `ggfortify::autoplot`)

10.2 Heatmaps

With significant genes at hand, lets have a closer look. We will continue working with the *log* scaled CPM matrix we used for the PCA, but first we must reduce it.

1. Extract genes with $FDR <= 0.01$ and $|logFC| >= 2$ (Hint: `subset`)
2. How many genes do you have? (Is there something common between them?)
3. There are several packages for heatmaps, choose one, we recommend `ComplexHeatmap`[10]. Install...
4. Plot a heatmap of the significant genes.
5. To make it more readable, use the gene names instead of the Ensembl IDs

[10] `https://bioconductor.org/packages/release/bioc/html/ComplexHeatmap.html`



10.3 Enrichment

We are somewhat hampered by only working with a subset of chromosomes, but we can still try to get a feeling if there is a particular subset of genes among the most significant. Extract gene names that you visualized in the heatmap and choose the ones that have increased expression either in kidney or liver. Submit the two lists of genes to String[11].

You will have a chance to explore this further tomorrow.

10.4 Optional: Alternative models

Let us revisit section 7.6. We created the simplest model possible. Lets explore this further.

1. We have replicate information. Start by adding this to the model and see if there are genes connected to this factor.

```
#Model
~ group + replicate
```

   Did you get any significant genes? How does this compare to the PCA plot (see Figure 1)?

2. When looking to the PCA plot, notice how the time points influence the location in PCA space. Perhaps, there is something to this?

```
#Model
~ as.numeric(timePoint)
```

   Are the most significant genes the same?

   a) Plot log_{10}-scaled FDR values from the base comparison against those from this timePoint comparison
   b) Plot logFC from the two comparisons against each other. Color based on whether the gene is significant in either comparison (four classes).
   c) Extract the genes that are significant in different comparisons and plot them in a heatmap. Are there differences between the classes?

[11] https://string-db.org/cgi/input.pl?input_page_active_form=multiple_identifiers