# 1. Introduction

## 1.1. Overview of CUT&RUN

Chromatin immunoprecipitation (ChIP) and its variations are the main techniques to analyse transcription factor (TF) binding to DNA and histone modifications. However, ChIP and its sequencing version (ChIP-seq) present some drawbacks that result in poor resolution, high background (noise-to-signal ratios), or problems with antibody specificity.

In recent years, the laboratory of Professor Henikoff has developed and implemented CUT&RUN (Cleavage Under Targets and Release Using Nuclease; Kaya-Okur et al., 2019 and Meers et al., 2019, a technique for genome-wide profiling of TF binding sites, chromatin-bound complexes, and histone modifications, by using a micrococcal nuclease. This enzyme has both endo- and exo-nuclease activity that fragments the chromatin, generating precise protein-DNA footprints. Although this method still relies on antibody specificity, it allows for a reduction in the number of cell input material, reduces background noise, and the number of sequencing reads needed for the analysis.

Briefly, CUT&RUN is performed in situ on intact cells without cross-linking. An incubation with your antibody of interest (TF, associated complex or histone mark) is performed, followed by the addition of a micrococcal nuclease fused to Protein A (pA-MNase) and/or Protein G (pA/G-MNase) that is used to fragment the DNA surrounding the protein of interest. The fusion protein pA/G-MNase binds directly to the Fc region of your bound antibody to the target. With the addition of calcium, the DNA under your desired target is cleaved and released. Once released, they can freely diffuse outside the cell and be collected, extracted, and processed for next-generation sequencing (NGS).

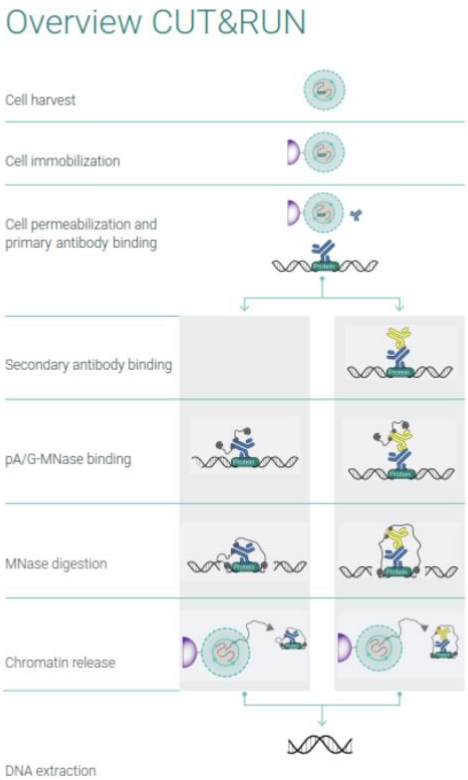



*Figure 1*. CUT&RUN overview from protocols.io (dx.doi.org/10.17504/protocols.io.bdwni7de)

## 1.2. Objectives

This tutorial is designed to process CUT&RUN data that has been generated following this bench protocol. In this tutorial, we will process data for CTCF and histone modifications from the human lymphoma K562 cell line from Meers et al., 2019.

For the data analysis, some steps are common to the ChIP-seq analysis with some specificities and important differences due to the different nature of the protocols. Although some companies offer the service of bioinformatic data analysis and some software are currently available (nf-core/cutandrun, basepair), we will follow a modified tutorial from Zheng Y et al., 2020 (Protocol.io).

### 1.3. CUT&RUN data processing and analysis outline

- Data pre-processing
    - FastQC for quality control assessment of the sequenced files
        - Installing FastQC
        - Running FastQC
        - Interpreting the FastQC results
        - Merging technical replicates or different sequence runs beforehand (Optional, decide if needed)

- Alignments
    - Alignment to the human reference genome (hg38)
    - Alignment to the spike-in genome (yeast/E. coli)
    - Report sequencing mapping summary
        - Sequencing depth CUT&RUN sample
        - Sequencing depth spike-in genome
        - Merge the alignment tables for hg38 and the spiked-in yeast
        - Plot sequencing and alignment results for comparison
    - Removing duplicates? (optional)
    - Assess mapped fragment size distribution
    - Replicate reproducibility assessment

- Alignment result filtering and file format conversion
    - Filtering mapped reads by the mapping quality filtering (optional)
    - File format conversion
    - Replicate reproducibility

- Spike-in calibration (optional)
    - Scaling factor

- Peak calling
    - SEACR
        - Summary of called peaks
        - Reproducibility of the peaks
        - Calculate the FRagment proportion in Peaks regions (FRiPs).
        - Visualisation of peak number, peak width, peak reproducibility and FRiPs

- Data visualisation
    - Browser display of normalised bedgraph files
    - Heatmap of specific regions
        - Heatmap over transcription units
        - Heatmap on CUT&RUN peaks

- Differential peak analysis
    - Generate the peak sample matrix
        - Generate a master peak list containing all the called peaks (in all replicates)
        - Get the fragment counts for each peak in the master peak list
        - Perform sequencing depth normalisation and differential enriched peaks detection

- Other ways to make the calculations
    - Data normalisation without spike-in DNA
    - Peak calling
    - Differential peak analysis
    - Pipelines

### 1.4. System requirements

Linux
Rstudio (optional)
R (versions>=3.6)
R libraries: dplyr, stringr, ggplot2, viridis, GenomicRanges, chromVAR, DESeq2, ggpubr, corrplot, ChIPseqSpikeInFree (optional)
FastQC (version >=0.11.9)
Bowtie2 (version >=2.3.4.3)
Samtools (version >=1.10)
Bedtools (version >=2.29.1)
Picard (version >=2.18.29)
SEACR (version >=1.3) | MACS2
deepTools (version >=2.0)

```
library(dplyr)

library(stringr)

library(ggplot2)

library(viridis)

library(GenomicRanges)

library(chromVAR) ## For FRiP analysis and differential analysis

library(DESeq2) ## For differential analysis section

library(ggpubr) ## For customizing figures

library(corrplot) ## For correlation plot
```

### 1.5. Data

In this practical, we will use data from Meers et al., 2019, available for downloading at Gene Expression Omnibus (GEO) or the European Nucleotide Archive (ENA). The corresponding SRA entries are as follows:

| experiment_type | protocol | cell type | enzyme/antibody | pA/pAG | treatment | url |
|---|---|---|---|---|---|---|
| CUT&RUN | Meers et al., 2019 | k562 | CTCF_millipore | pAG | hiCa-1min | SRR8581587 |
| CUT&RUN | Meers et al., 2019 | k562 | CTCF_millipore | pAG | | SRR8581588 |
| CUT&RUN | Meers et al., 2019 | k562 | CTCF_millipore | pAG | hiCa-27min_rep1 | SRR8581589 |
| CUT&RUN | Meers et al., 2019 | k562 | CTCF_millipore | pAG | hiCa-27min_rep2 | SRR8581590 |
| CUT&RUN | Meers et al., 2019 | k562 | CTCF_millipore | pAG | hiCa-3min | SRR8581591 |
| CUT&RUN | Meers et al., 2019 | k562 | CTCF_millipore | pAG | hiCa-9min | SRR8581592 |
| CUT&RUN | Meers et al., 2019 | k562 | CTCF_millipore | pA | hiCa-27min | SRR8581594 |
| CUT&RUN | Meers et al., 2019 | k562 | CTCF_millipore | pA | hiCa-3min | SRR8581595 |
| CUT&RUN | Meers et al., 2019 | k562 | CTCF_millipore | pA | hiCa-9min | SRR8581596 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27ac_abcam | pA | hiCa-1min | SRR8581593 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27ac_abcam | | hiCa-1min | SRR8581597 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27ac_abcam | | hiCa-20sec | SRR8581598 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27ac_abcam | | hiCa-27min_rep1 | SRR8581599 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27ac_abcam | | hiCa-3min | SRR8581600 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27ac_abcam | | hiCa-9min | SRR8581601 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27ac_abcam | | std-1min | SRR8581602 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27ac_abcam | | std-20sec | SRR8581603 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27ac_abcam | | std_3min | SRR8581604 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27ac_abcam | | std_9min | SRR8581605 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27ac_abcam | | gG_hi-Ca | SRR8581615 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27ac_abcam | pAG | hiCa_pellet_3min | SRR8581617 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos | pAG | hiCa_pellet_9min | SRR8581618 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos | pAG | hiCa_pellet_3min | SRR8581621 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos | pAG | hiCa_pellet_9min | SRR8581622 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos | pAG | hi-Ca_supn_3min | SRR8581623 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos | pAG | hi-Ca_supn_9min | SRR8581624 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos | pAG | hi-Ca_total_30min | SRR8581625 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos | pAG | std_total_30min | SRR8581626 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K4me2 | pA | hi-Ca_10min | SRR8634175 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K4me2 | pA | hi-Ca_2min | SRR8634176 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K4me2 | pA | hi-Ca_total_10min | SRR8634177 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K4me2 | pA | hi-Ca_total_2min | SRR8634178 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K4me2 | pA | std_10min | SRR8634179 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K4me2 | pA | std_2min | SRR8634180 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K4me2 | pA | std_total_10min | SRR8634181 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K4me2 | pA | std_total_2min | SRR8634182 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27ac_millipore | pAG | hi-Ca_1min | SRR8581606 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27ac_millipore | pAG | hi-Ca_27min | SRR8581607 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27ac_millipore | pAG | hi-Ca_27min | SRR8581608 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27ac_millipore | pAG | hi-Ca_3min | SRR8581609 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27ac_millipore | pAG | hi-Ca_9min | SRR8581610 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27ac_millipore | pA | hi-Ca_1min | SRR8581611 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27ac_millipore | pA | hi-Ca_27min | SRR8581612 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27ac_millipore | pA | hi-Ca_3min | SRR8581613 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27ac_millipore | pA | hi-Ca_9min | SRR8581614 |
| CUT&RUN | Meers et al., 2019 | k562 | IgG-Guinea pig anti-rabbit | | hi-Ca | SRR8581615 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos_cell_signalling_rabbit | pAG | hi-Ca_pellet_3min | SRR8581616 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos_cell_signalling_rabbit | pAG | hi-Ca__pellet_9min | SRR8581617 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos_cell_signalling_rabbit | pAG | hi-Ca_pellet_3min | SRR8581621 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos_cell_signalling_rabbit | pAG | hi-Ca__pellet_9min | SRR8581622 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos_cell_signalling_rabbit | pAG | hi-Ca_supn_3min | SRR8581623 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos_cell_signalling_rabbit | pAG | hi-Ca_supn_9min | SRR8581624 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos_cell_signalling_rabbit | pAG | hi-Ca_total_30min | SRR8581625 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos_cell_signalling_rabbit | pAG | hi-Ca_total_30min | SRR8581626 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos_abcam_mouse | pAG | hi-Ca_total_30min | SRR8581619 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos_abcam_mouse | pAG | hi-Ca_total_30min | SRR8581620 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos_abcam_mouse | pAG | hi-Ca_pellet_10min | SRR8634183 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos_abcam_mouse | pAG | hi-Ca_pellet_2min | SRR8634184 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos_abcam_mouse | pAG | std_supn_2min | SRR9073703 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos_abcam_mouse | pAG | std_supn10min | SRR9073704 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos_abcam_mouse | pAG | std_supn_2min | SRR9073705 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos_abcam_mouse | pAG | std_total_10min | SRR9073706 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos | pAG | hi-Ca_10min | SRR8634185 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos | pA | hi-Ca_pellet_10min | SRR8634186 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos_cell_signalling_rabbit | pAG | std_supn_2min | SRR9073707 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos_cell_signalling_rabbit | pAG | std_supn_10min | SRR9073708 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos_cell_signalling_rabbit | pAG | std_total_2min | SRR9073709 |
| CUT&RUN | Meers et al., 2019 | k562 | PolIISer5phos_cell_signalling_rabbit | pAG | std_total_10min | SRR9073710 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27me3_cell_signalling_rabbit | pA | 1to500_std_supn_30min | SRR9073700 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27me3_cell_signalling_rabbit | pAG | 1to1000_std_supn_30min | SRR9073701 |
| CUT&RUN | Meers et al., 2019 | k562 | H3K27me3_cell_signalling_rabbit | pAG | 1to8000_std_supn_30min | SRR9073702 |

Before starting, define your working directory by specifying your path:

```
##linux##

#yourPath="/path/to/project/where/data/and/results/are/saved"
```

Taking H3K37me3 replicates as an example:

H3K27me3_s1= SRR9073700
H3K27me3_s2= SRR9073701
H3K27me3_s3= SRR9073702
IgG= SRR8581615

```
##linux##

wget -O $yourPath/data/h3k27me3_rep1/SRR9073700_1.fastq.gz ftp://ftp.sra.ebi.ac.uk/vol1/fastq/
SRR907/000/SRR9073700/SRR9073700_1.fastq.gz

wget -O $yourPath/data/h3k27me3_ rep1/SRR9073700_2.fastq.gz ftp://ftp.sra.ebi.ac.uk/vol1/fastq
/SRR907/000/SRR9073700/SRR9073700_2.fastq.gz

wget -O $yourPath/data/h3k27me3  rep2/SRR9073701 1.fastq.gz ftp://ftp.sra.ebi.ac.uk/vol1/fastq
/SRR907/001/SRR9073701/SRR9073701_1.fastq.gz

wget -O $yourPath/data/h3k27me3_ rep2/SRR9073701_2.fastq.gz ftp://ftp.sra.ebi.ac.uk/vol1/fastq
/SRR907/001/SRR9073701/SRR9073701_2.fastq.gz

wget -O $yourPath/data/h3k27me3  rep3/SRR9073701 1.fastq.gz ftp://ftp.sra.ebi.ac.uk/vol1/fastq
/SRR907/002/SRR9073702/SRR9073702_1.fastq.gz

wget -O $yourPath/data/h3k27me3_ rep3/SRR9073701_2.fastq.gz ftp://ftp.sra.ebi.ac.uk/vol1/fastq
/SRR907/002/SRR9073702/SRR9073702_2.fastq.gz

wget -O $yourPath/data/IgG_ rep1/SRR8581615_1.fastq.gz ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR8
58/005/SRR8581615/SRR8581615_1.fastq.gz

wget -O $yourPath/data/IgG  rep1/SRR8581615 2.fastq.gz ftp://ftp.sra.ebi.ac.uk/vol1/fastq/SRR8
58/005/SRR8581615/SRR8581615_2.fastq.gz
```

## 2. Data pre-processing **(Practical 1)**

### 2.1. FastQC for quality control assessment of the sequenced files

It is highly recommended to check the sequencing runs' quality regardless of whether the researcher generates their own data or downloads it from a public repository.

#### 2.1.1. Installing FastQC

```
# Getting and installing FastQC. In this course, FastQC has been pre-installed for you.

# To obtain FastQC, download it directly from the Babraham Institute source:

##linux##

mkdir -p $yourPath/software/

wget -P $yourPath/software https://www.bioinformatics.babraham.ac.uk/projects/fastqc/fastqc_v0
.11.9.zip

cd $yourPath/software/

unzip fastqc_v0.11.9.zip
```

#### 2.1.2. Running FastQC

```
# Now, it is time to check the QC of your samples. By using the next command, FastQC will QC c
heck all of the fastq.gz files stored in the specified     directory

mkdir -p ${yourPath}/fastqFileQC/${Name}

cd ${yourPath}/fastqFileQC/
```

```
$yourPath/tools/FastQC/fastqc -o ${yourPath}/fastqFileQC/$[Name} -f fastq ${yourPath}/data/h3k
27me3_rep1/*.fastq.gz

$yourPath/tools/FastQC/fastqc -o ${yourPath}/fastqFileQC/$[Name} -f fastq ${yourPath}/data/h3k
27me3_rep2/*.fastq.gz

$yourPath/tools/FastQC/fastqc -o ${yourPath}/fastqFileQC/$[Name} -f fastq ${yourPath}/data/h3k
27me3_rep3/*.fastq.gz

$yourPath/tools/FastQC/fastqc -o ${yourPath}/fastqFileQC/$[Name} -f fastq ${yourPath}/IgG rep1
/*.fastq.gz

# Depending on the specificities of your computer and the size of the files you want to QC, it
might be advisable to run QC separately (calling one file at a time)
```

### 2.1.3. Interpreting the FastQC results

Before interpreting your results, it is recommended to check the FastQC guidelines, for example, reports.

This is a normal "per base sequence content" when dealing with CUT&RUN data. This does not mean that your sequencing run failed.

It will not affect the alignment of the reference genomes. It is not recommended to trim these base pairs out or trim the adaptors (the "overrepresented sequences" flag is due to the Illumina Multiplexing PCR primer) recommended for other NGS techniques.

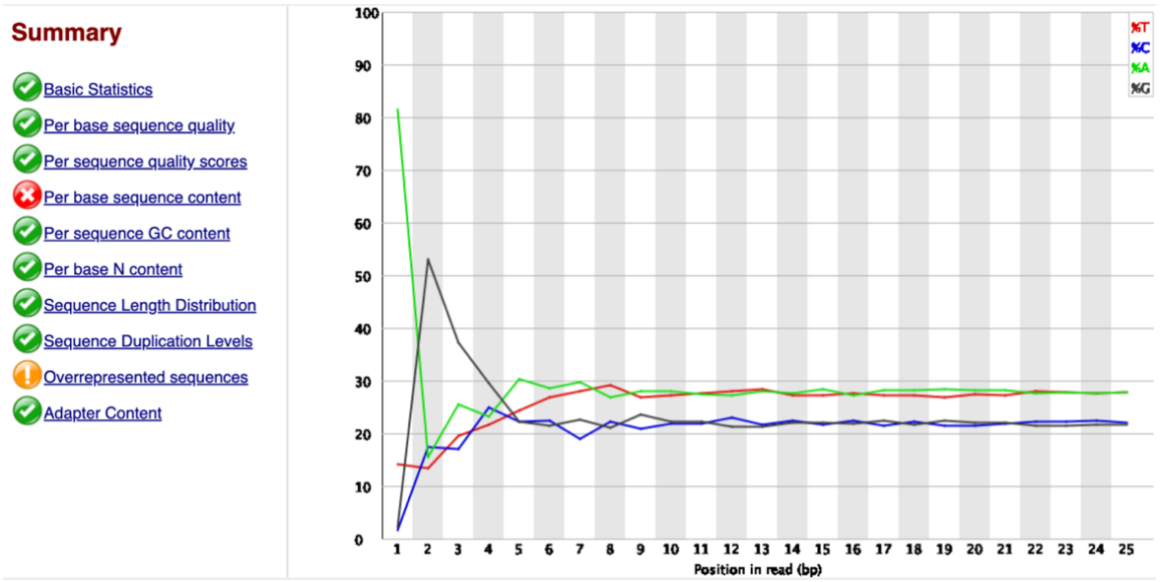



*Figure 2*. FastQC summary report.

### 2.1.4 Merging technical replicates or different sequence runs beforehand (Optional, decide if needed)

You have the option to merge different technical replicates or different sequence runs beforehand. Although this won't be the case in our tutorial, you can use the cat command.

```
##linux##

Name="H3K27me3_"
```

```
mkdir -p ${yourPath}/fastq_merged

cat ${yourPath}/${Name}/*_R1_*.fastq.gz >${yourPath}/fastq_merged/${Name}_R1.fastq.gz

cat ${yourPath}/${Name}/*_R2_*.fastq.gz >${yourPath}/fastq_merged/${Name}_R2.fastq.gz
```

## 3. Alignments **(Practical 2)**

CUT&RUN libraries include PCR primer barcoding for sequencing. Typically, CUT&RUN libraries are sequenced paired-end for a length of 25 bp (25x25 PE Illumina sequencing) and pool different libraries on the same sequencing run.

Generally, 5 million paired-end reads should be enough to provide high-quality profiling of abundant chromatin features, such as histone modifications, provided a specific and high-yield antibody. This needs to be adjusted depending on antibody quality (increasing the number of reads for generating robust chromatin profiles), the abundance of the features analysed (abundant features require fewer reads), etc.

### 3.1. Alignment to the human reference genome (hg38)

We will use Bowtie2 to align the reads with the human hg38 reference genome. More details on Bowtie2 parameters can be found here.

```
##linux##

cores=8 ## You need to specify the number of cores you will use. This will highly depend on yo
ur computer or server's capabilities.

ref="/path/to/bowtie2Index/hg38" ## we have pre-built the bowtie2Index for your


mkdir -p ${yourPath}/alignment/sam/bowtie2_summary

mkdir -p ${yourPath}/alignment/bam

mkdir -p ${yourPath}/alignment/bed

mkdir -p ${yourPath}/alignment/bedgraph


## We have built the bowtie2 reference genome hg38 index for you.

## You need to specify the path to your bowtie2-build: path/to/hg38/fasta/hg38.fa /path/to/bow
tie2Index/hg38

bowtie2 --end-to-end --very-sensitive --no-mixed --no-discordant --phred33 -I 10 -X 700 -p ${c
ores} -x ${ref} -1 ${yourPath}/fastq/${Name} R1.fastq.gz -2 ${yourPath}/fastq/${Name} R2.fastq
.gz -S ${yourPath}/alignment/sam/${Name} bowtie2.sam &> ${yourPath}/alignment/sam/bowtie2 summ
ary/${Name}_bowtie2.txt
```

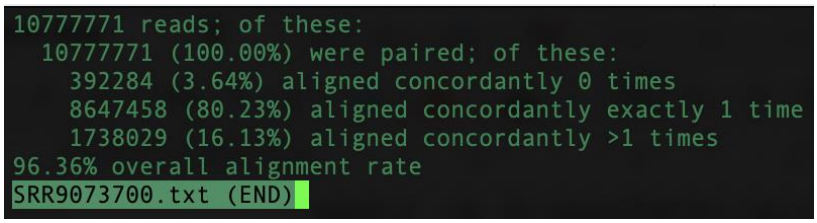
These Bowtie2 parameters will align paired-end reads with an insert length between 10 (`-I 10`) and 700 bp (`-X 700`). If you have performed a 25x25 PE sequencing, adapters will not be included in any read insert longer than 25 bp.

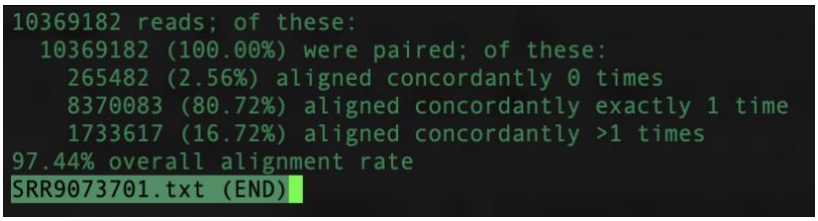
However, keep in mind that if you sequence longer reads, you will need to use Cutadapt or Trim Galore to remove adapter sequences and change the Bowtie2 argument to `--local` to remove any remaining adapter sequence at the 3' end of reads during the mapping process.

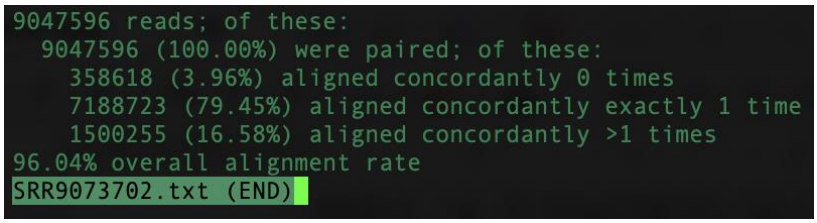
```
bowtie2 --local --very-sensitive --no-mixed --no-discordant --phred33 -I 10 -X 700 -p ${cores}
-x ${ref} -1 ${yourPath}/fastq/${Name} R1.fastq.gz -2 ${yourPath}/fastq/${Name} R2.fastq.gz -S
${yourPath}/alignment/sam/${Name}_bowtie2.sam &> ${yourPath}/alignment/sam/bowtie2_summary/${N
ame}_bowtie2.txt
```

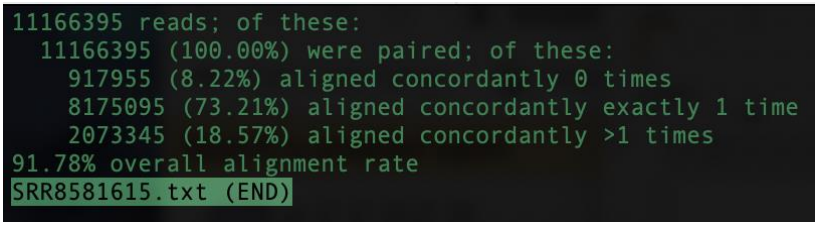
Results:

The generated ${Name}_bowtie2.txt file will contain the summary results and will look similar to this:




The first line shows the number of total reads and the sequenced depth in total number of paired reads (SRR9073700= 10,777,771; SRR9073701= 10,369,182; SRR9073702= 9,047,596).
In our case, 100% of the reads in the three samples are paired reads.

The third line shows the reads aligned concordantly 0 times, it is the number of unmapped read pairs (SRR9073700= 392,284; SRR9073701= 265,482; SRR9073702= 358,618).

The sum of the fourth and fifth lines represents the number of reads that successfully map to the reference genome.

The final line shows the overall alignment rate to the reference genome (SRR9073700= 96.36%; SRR9073701= 97.44%; SRR9073702= 96.04%).

### 3.2. Alignment to the spike-in genome (yeast/E. coli)

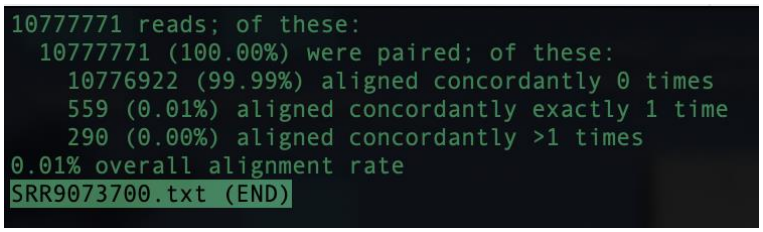
This step is optional but recommended and will depend on your experiment's type of spike-in calibration. It could be a yeast spike-in or the E. coli that is carried along with bacterially-produced pA-Tn5 protein (CUT&Tag protocols) that is non-specifically tagmented during the

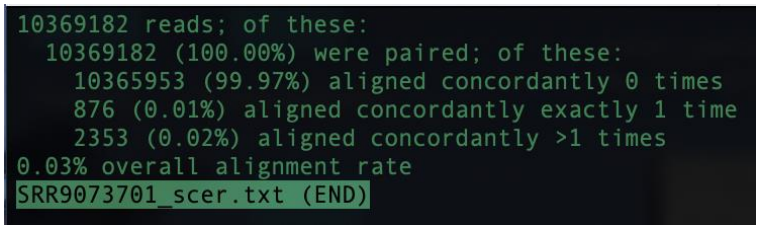
reaction. The fraction of total reads mapping to the spike-in genome will depend on the yield of epitope-targeted CUT&RUN, the number of cells used, and the abundance of that epitope in chromatin.

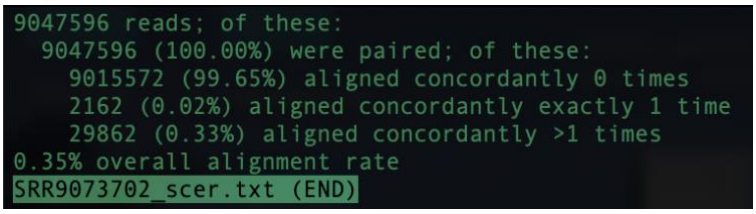
The purpose of the spike-in is to add a fixed amount of either yeast or E. coli. Spike-in reads will be used to normalise epitope abundance in a set of experiments.

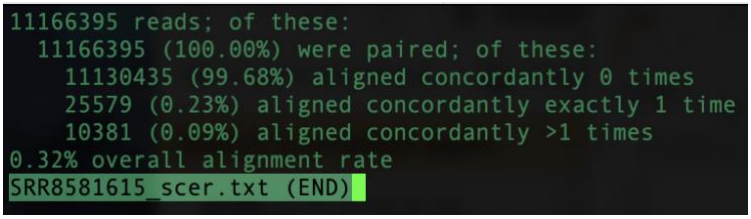
```
##linux##

spikeInRef="/path/to/bowtie2Index/Scer" #Scer=S. Cerevisiae

## bowtie2-build path/to/Ecoli/fasta/Ecoli.fa /path/to/bowtie2Index/Ecoli

bowtie2 --end-to-end --very-sensitive --no-mixed --no-discordant --no-overlap --no-dovetail --
phred33 -I 10 -X 700 -p ${cores} -x ${spikeInRef} -1 ${yourPath}/fastq/${Name} R1.fastq.gz -2
${yourPath}/fastq/${Name}_R2.fastq.gz -S $yourPath/alignment/sam/${Name}_bowtie2_spikeIn.sam &
> $yourPath/alignment/sam/bowtie2_summary/${Name}_bowtie2_spikeIn.txt
```

The `--no-overlap --no-dovetail` arguments are used to avoid possible cross-mapping between the experimental genome and that of the spike-in genome (yeast or carry-over E.coli in the case of CUT&Tag).
Results:




```
seqDepthDouble='samtools view -F 0x04 $yourPath/alignment/sam/${Name}_bowtie2_spikeIn.sam | wc
-l'

seqDepth=$((seqDepthDouble/2))
```

```
echo $seqDepth >$yourPath/alignment/sam/bowtie2_summary/${Name}_bowtie2_spikeIn.seqDepth
```

Results for `seqDepth`:

SRR9073700= `849`
SRR9073701= `3,229`
SRR9073702= `32,024`
IgG= `71,920`

### 3.3. Report sequencing mapping summary

It is required to assess and report the alignment efficiency by summarising some metrics on the raw reads and uniquely mapping reads.

In a successful experiment, you should expect alignment frequencies higher than 80% since CUT&RUN data has very low backgrounds. As few as 1 million mapped fragments are enough to give a robust profile for histone modifications in the human genome. However, if you are interested in profiling less-abundant transcription factors and/or chromatin proteins, getting robust profiles may require ten times more mapped fragments.

**Evaluate:** sequencing depth, alignment rate, number of mappable fragments, duplication rate, unique library size, and fragment size distribution.

#### 3.3.1.Sequencing depth CUT&RUN sample

```
##R##

## Path to the project and histone list

yourPath = "/path/to/your/files/"

sampleList = c("H3K27me3_rep1", "H3K27me3_rep2", "H3K27me3_rep3", "IgG")

histList = c("H3K27me3", "IgG")

## Collect the alignment results from the bowtie2 alignment summary files

alignResult = c()

for(hist in sampleList){

  alignRes = read.table(paste0(yourPath, hist, "/alignment/sam/bowtie2_summary", "_bowtie2.txt
"), header = FALSE, fill = TRUE)

  alignRate = substr(alignRes$V1[6], 1, nchar(as.character(alignRes$V1[6]))-1)

  histInfo = strsplit(hist, "_")[[1]]

  alignResult = data.frame(Histone = histInfo[1], Replicate = histInfo[2],

                           SequencingDepth = alignRes$V1[1] %>% as.character %>% as.numeric,

                           MappedFragNum_hg38 = alignRes$V1[4] %>% as.character %>% as.numeric
+ alignRes$V1[5] %>% as.character %>% as.numeric,

                           AlignmentRate_hg38 = alignRate %>% as.numeric)  %>% rbind(alignResul
t, .)

}

alignResult$Histone = factor(alignResult$Histone, levels = histList)

alignResult %>% mutate(AlignmentRate_hg38 = paste0(AlignmentRate_hg38, "%"))
```

The resulting table will look like this:

| | Histone | Replicate | SequencingDepth | MappedFragNum_hg38 | AlignmentRate_hg38 |
|---|---|---|---|---|---|
| 1 | H3K27me3 | rep1 | 10777771 | 10385487 | 96.36% |
| 2 | H3K27me3 | rep2 | 10369182 | 10103700 | 97.44% |
| 3 | H3K27me3 | rep3 | 9047596 | 8688978 | 96.04% |
| 4 | IgG | rep1 | 11166395 | 10248440 | 91.78% |

### 3.3.2. Sequencing depth spike-in genome

```
##R##
spikeAlign = c()

for(hist in sampleList){

  spikeRes = read.table(paste0(yourPath, hist, "/alignment/sam/bowtie2_summary", "_scer.txt"),
header = FALSE, fill = TRUE)

  alignRate = substr(spikeRes$V1[6], 1, nchar(as.character(spikeRes$V1[6]))-1)

  histInfo = strsplit(hist, "_")[[1]]

  spikeAlign = data.frame(Histone = histInfo[1], Replicate = histInfo[2],

                          SequencingDepth = spikeRes$V1[1] %>% as.character %>% as.numeric,

                          MappedFragNum_spikeIn = spikeRes$V1[4] %>% as.character %>% as.numer
ic + spikeRes$V1[5] %>% as.character %>% as.numeric,

                          AlignmentRate spikeIn = alignRate %>% as.numeric)  %>% rbind(spikeAl
ign, .)

}

spikeAlign$Histone = factor(spikeAlign$Histone, levels = histList)

spikeAlign %>% mutate(AlignmentRate_spikeIn = paste0(AlignmentRate_spikeIn, "%"))
```

The resulting table will look like this:

| | Histone | Replicate | SequencingDepth | MappedFragNum_spikeIn | AlignmentRate_spikeIn |
|---|---|---|---|---|---|
| 1 | H3K27me3 | rep1 | 10777771 | 849 | 0.01% |
| 2 | H3K27me3 | rep2 | 10369182 | 3229 | 0.03% |
| 3 | H3K27me3 | rep3 | 9047596 | 32024 | 0.35% |
| 4 | IgG | rep1 | 11166395 | 35960 | 0.32% |

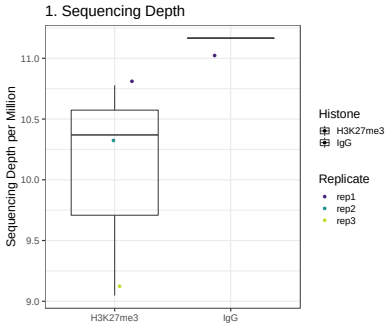
### 3.3.3. Merge the alignment tables for hg38 and the spiked-in yeast

```
##R##

alignSummary = left_join(alignResult, spikeAlign, by = c("Histone", "Replicate", "SequencingDe
pth")) %>%

  mutate(AlignmentRate_hg38 = paste0(AlignmentRate_hg38, "%"),

         AlignmentRate_spikeIn = paste0(AlignmentRate_spikeIn, "%"))

alignSummary
```

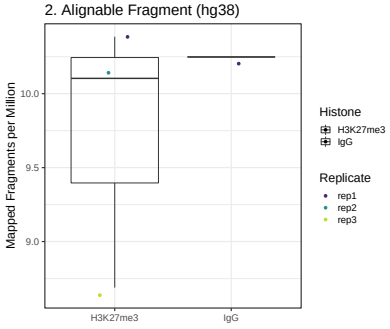
| | Histone | Replicate | SequencingDepth | MappedFragNum_hg38 | AlignmentRate_hg38 | MappedFragNum_spikeIn | AlignmentRate_spikeIn |
|---|---|---|---|---|---|---|---|
| 1 | H3K27me3 | rep1 | 10777771 | 10385487 | 96.36% | 849 | 0.01% |
| 2 | H3K27me3 | rep2 | 10369182 | 10103700 | 97.44% | 3229 | 0.03% |
| 3 | H3K27me3 | rep3 | 9047596 | 8688978 | 96.04% | 32024 | 0.35% |
| 4 | IgG | rep1 | 11166395 | 10248440 | 91.78% | 35960 | 0.32% |

### 3.3.4. Plot sequencing and alignment results for comparison **(Practical 3)**

```
##R##

## Generate sequencing depth boxplot

fig1 = alignResult %>% ggplot(aes(x = Histone, y = SequencingDepth/1000000, fill = Histone)) +

    geom_boxplot() +

    geom_jitter(aes(color = Replicate), position = position_jitter(0.15)) +

    scale_fill_viridis(discrete = TRUE, begin = 0.1, end = 0.9, option = "magma", alpha = 0.8)
+

    scale_color_viridis(discrete = TRUE, begin = 0.1, end = 0.9) +

    theme_bw(base_size = 18) +

    ylab("Sequencing Depth per Million") +

    xlab("") +

    ggtitle("1. Sequencing Depth")
```

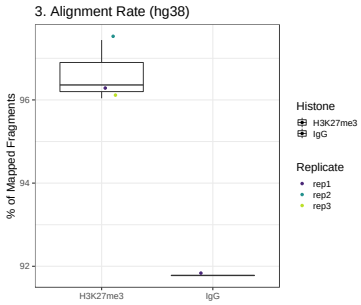



```
fig2 = alignResult %>% ggplot(aes(x = Histone, y = MappedFragNum_hg38/1000000, fill = Histone)
) +

    geom_boxplot() +

    geom_jitter(aes(color = Replicate), position = position_jitter(0.15)) +

    scale_fill_viridis(discrete = TRUE, begin = 0.1, end = 0.9, option = "magma", alpha = 0.8)
+

    scale_color_viridis(discrete = TRUE, begin = 0.1, end = 0.9) +

    theme_bw(base_size = 18) +

    ylab("Mapped Fragments per Million") +

    xlab("") +

    ggtitle("2. Alignable Fragment (hg38)")
```

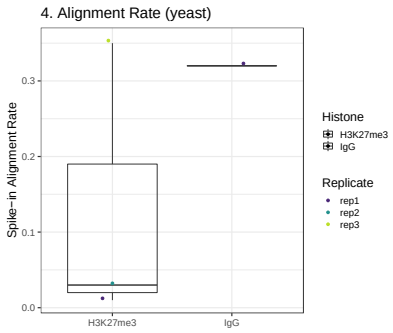



```
fig3 = alignResult %>% ggplot(aes(x = Histone, y = AlignmentRate_hg38, fill = Histone)) +
```

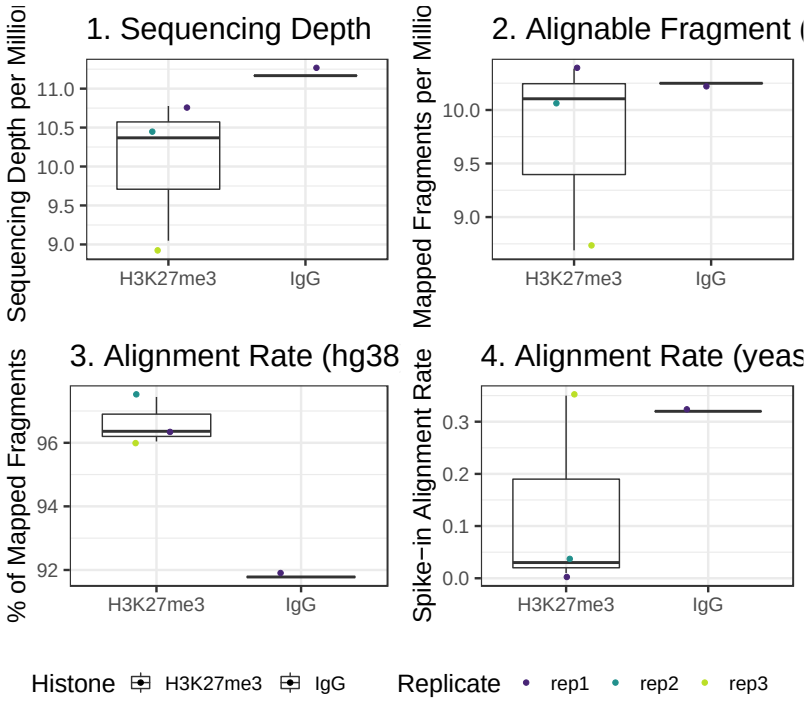
```
    geom_boxplot() +
    geom_jitter(aes(color = Replicate), position = position_jitter(0.15)) +
    scale_fill_viridis(discrete = TRUE, begin = 0.1, end = 0.9, option = "magma", alpha = 0.8)
+
    scale_color_viridis(discrete = TRUE, begin = 0.1, end = 0.9) +
    theme_bw(base_size = 18) +
    ylab("% of Mapped Fragments") +
    xlab("") +
    ggtitle("3. Alignment Rate (hg38)")
```




```
fig4 = spikeAlign %>% ggplot(aes(x = Histone, y = AlignmentRate_spikeIn, fill = Histone)) +
    geom_boxplot() +
    geom_jitter(aes(color = Replicate), position = position_jitter(0.15)) +
    scale_fill_viridis(discrete = TRUE, begin = 0.1, end = 0.9, option = "magma", alpha = 0.8)
+
    scale_color_viridis(discrete = TRUE, begin = 0.1, end = 0.9) +
    theme_bw(base_size = 18) +
    ylab("Spike-in Alignment Rate") +
    xlab("") +
    ggtitle("4. Alignment Rate (yeast)")
```




```
ggarrange(fig1, fig2, fig3, fig4, ncol = 2, nrow=2, common.legend = TRUE, legend="bottom")
```




For abundant targets such as H3K27me3 histone modification in humans, the expected yeast spike-in would be around ~0.01% to 10%. However, remember that lower input cell numbers and less abundant epitopes can increase the percentage of the spike-in reads (sometimes even as much as 70%). Likewise, IgG controls usually have higher spike-in reads than abundant epitopes such as histone modifications.

### 3.4. Removing duplicates? (optional)

Like in other NGS techniques, there is always a fraction of sequenced and mapped reads that are duplicated (i.e., they are the fruit of the same fragment that is amplified multiple times by PCR).
The researcher, based on the number of starting material and/or the selected number of PCR amplification cycles, should assess the complexity of the libraries and decide if a deduplication step is needed. However, having said so, in the case of CUT&RUN, there is some consensus on retaining the duplicate reads, as those may be due to the influence that chromatin conformation has on the nuclease cleavage. Those shorter fragments could be identical but originate from different cells rather than PCR amplification.

If the percentage of multi-mapping reads appears to be very high, the researchers can look into the level of library complexity by running some programs like Picard.

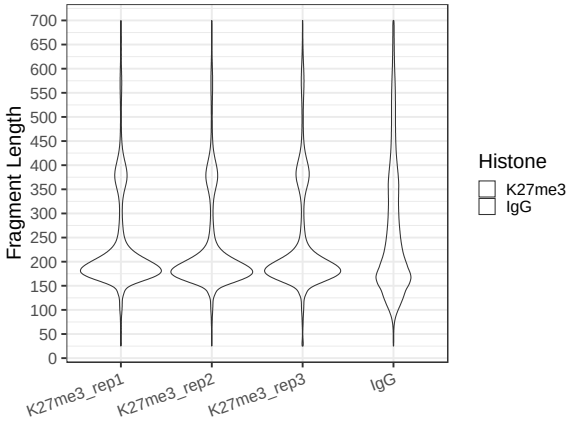
Then, it is suggested to call peaks in both libraries (duplicated and de-duplicated) and compare the number and nature of the peaks, motif enrichments, etc., before deciding what works best for your libraries.

### 3.5. Assess mapped fragment size distribution **(Practical 4)**

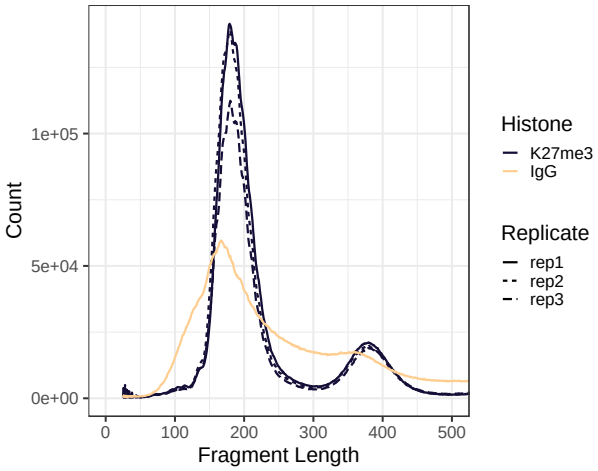
We expect fragments around nucleosomal length (~180 bp) or multiples of that length. TF CUT&RUN typically produce nucleosome-sized fragments plus shorter fragments (in variable numbers) from neighbouring nucleosomes and factor-bound sites.

```
##linux##

mkdir -p $yourPath/alignment/sam/fragmentLen

## Extract the 9th column from the alignment sam file, which is the fragment length

samtools view -F 0x04 $yourPath/alignment/sam/${Name} bowtie2.sam | awk -F'\t' 'function abs(x
){return ((x < 0.0) ? -x : x)} {print abs($9)}' | sort | uniq -c | awk -v OFS="\t" '{print $2,
$1/2}' >$yourPath/alignment/sam/fragmentLen/${Name}_fragmentLen.txt
```

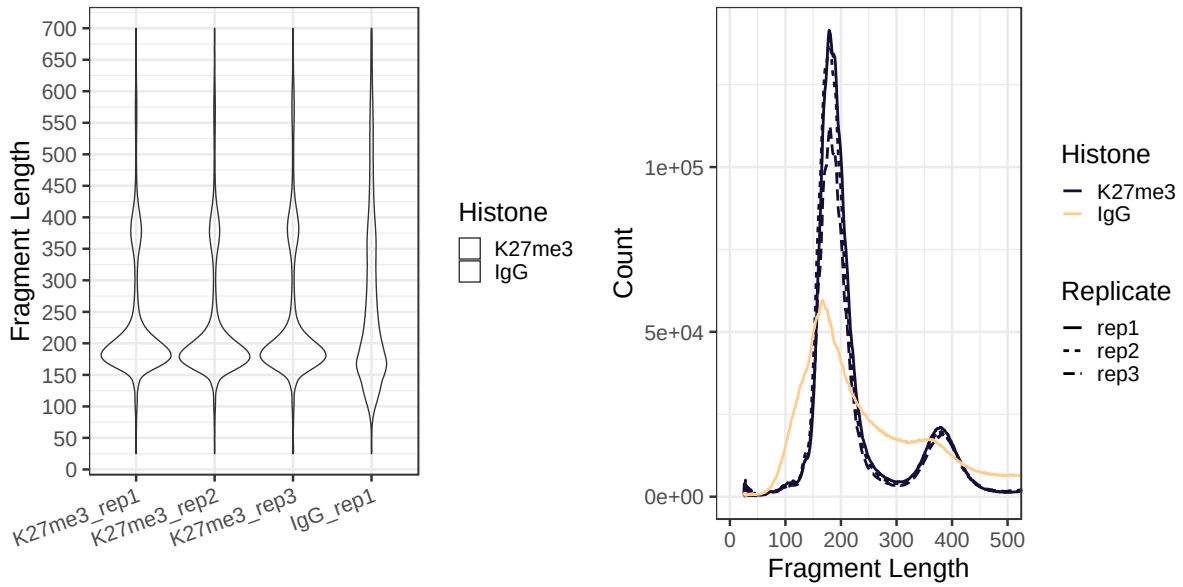
```
##R##

## Collect the fragment size information

yourPath = "/path/to/your/files/"

sampleList = c("H3K27me3_rep1", "H3K27me3_rep2", "H3K27me3_rep3", "IgG_rep1")

histList = c("H3K27me3", "IgG")


fragLen = c()

for(hist in sampleList){

  histInfo = strsplit(hist, "_")[[1]]

  fragLen = read.table(paste0(yourPath, hist, "/alignment/sam/fragmentLen/", " fragmentLen.txt
"), header = FALSE) %>% mutate(fragLen = V1 %>% as.numeric, fragCount = V2 %>% as.numeric, Wei
ght = as.numeric(V2)/sum(as.numeric(V2)), Histone = histInfo[1], Replicate = histInfo[2], samp
leInfo = hist) %>% rbind(fragLen, .)

}

fragLen$sampleInfo = factor(fragLen$sampleInfo, levels = sampleList)

fragLen$Histone = factor(fragLen$Histone, levels = histList)




## Generate the fragment size density plot (violin plot)

fig5 = fragLen %>% ggplot(aes(x = sampleInfo, y = fragLen, weight = Weight, fill = Histone)) +

    geom_violin(bw = 5) +

    scale_y_continuous(breaks = seq(0, 800, 50)) +

    scale_fill_viridis(discrete = TRUE, begin = 0.1, end = 0.9, option = "magma", alpha = 0.8)
+

    scale_color_viridis(discrete = TRUE, begin = 0.1, end = 0.9) +

    theme_bw(base_size = 20) +

    ggpubr::rotate_x_text(angle = 20) +

    ylab("Fragment Length") +

    xlab("")
```




```
fig6 = fragLen %>% ggplot(aes(x = fragLen, y = fragCount, color = Histone, group = sampleInfo,
linetype = Replicate)) +
  geom_line(size = 1) +
  scale_color_viridis(discrete = TRUE, begin = 0.1, end = 0.9, option = "magma") +
  theme_bw(base_size = 20) +
  xlab("Fragment Length") +
  ylab("Count") +
  coord_cartesian(xlim = c(0, 500))
```




```
ggarrange(fig5, fig6, ncol = 2) # To plot both figures side by side
```




### 3.6. Replicate reproducibility assessment

Correlation analysis of mapped read counts across the genome to measure data reproducibility. We will first convert the data into fragmented bed files.

## 4. Alignment result filtering and file format conversion **(Practical 5)**

### 4.1. Filtering mapped reads by the mapping quality filtering (optional)

Depending on your data, your datasets will need some stringent filtering on alignment quality.
Bowtie2 quality score assessment is based on the following:

MAPQ(x) = -10 * log10log10(P(x is mapped wrongly)) = -10 * log10(p)log10(p)
which ranges from 0 to 37, 40 or 42.

To eliminate all alignment results below the minQualityScore threshold (set by the researcher), use `samtools view -q minQualityScore`.

```
##linux##

minQualityScore=2

samtools view -q $minQualityScore ${yourPath}/alignment/sam/${Name} bowtie2.sam >${yourPath}/a
lignment/sam/${Name}_bowtie2.qualityScore$minQualityScore.sam

# If you apply the filtering step, instead of ${Name} bowtie2.sam use in the next
steps ${Name}_bowtie2.qualityScore$minQualityScore.sam.
```

### 4.2. File format conversion

Filtering and file format conversion steps towards peak calling and visualisation.

```
##linux##

## Filter and keep the mapped read pairs

samtools view -bS -F 0x04 $yourPath/alignment/sam/${Name}_bowtie2.sam >$yourPath/alignment/bam
/${Name}_bowtie2.mapped.bam
```

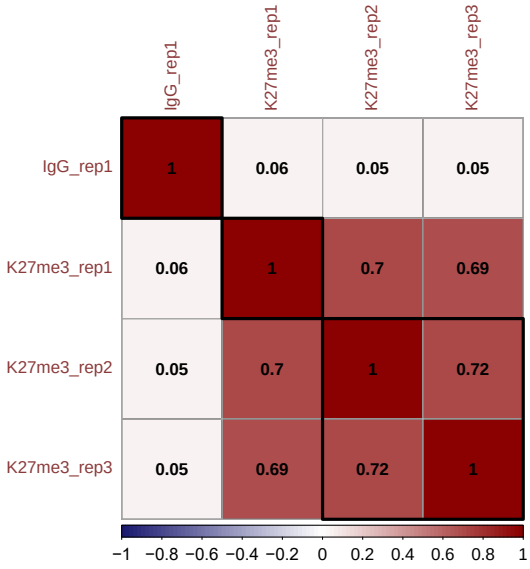
```
## Convert into bed file format

bedtools bamtobed -i $yourPath/alignment/bam/${Name}_bowtie2.mapped.bam -bedpe >$yourPath/alig
nment/bed/${Name}_bowtie2.bed


## Keep the read pairs from the same chromosome and fragment length less than 1000bp.

awk '$1==$4 && $6-$2 < 1000 {print $0}' $yourPath/alignment/bed/${Name} bowtie2.bed >$yourPath
/alignment/bed/${Name}_bowtie2.clean.bed


## Only extract the fragment related columns

cut -f 1,2,6 $yourPath/alignment/bed/${Name}_bowtie2.clean.bed | sort -k1,1 -k2,2n -k3,3n
>$yourPath/alignment/bed/${Name}_bowtie2.fragments.bed
```

### 4.3. Replicate reproducibility **(Practical 6)**

The genome is split into 500 bp bins to assess reproducibility between replicates and/or conditions. This step is followed by calculating the Pearson correlation of the log2-transformed values of read counts/bin between replicates.

```
##linux##
## We use the midpoint of each fragment to infer which 500bp bins this fragment belongs to.

binLen=500

awk -v w=$binLen '{print $1, int(($2 + $3)/(2*w))*w + w/2}'
$yourPath/alignment/bed/${Name} bowtie2.fragments.bed | sort -k1,1V -k2,2n | uniq -c | awk -v
OFS="\t" '{print $2, $3, $1}' |  sort -k1,1V -k2,2n
>$yourPath/alignment/bed/${Name}_bowtie2.fragmentsCount.bin$binLen.bed
```

```
##R##
reprod = c()
fragCount = NULL

for(hist in sampleList){

  if(is.null(fragCount)){

    fragCount = read.table(paste0(yourPath, "/alignment/bed/", hist, ".fragmentCount.bin500.be
d"), header = FALSE)

    colnames(fragCount) = c("chrom", "bin", hist)

  }else{

    fragCountTmp = read.table(paste0(yourPath, hist, ".fragmentCount.bin500.bed"), header = FA
LSE)

    colnames(fragCountTmp) = c("chrom", "bin", hist)

    fragCount = full_join(fragCount, fragCountTmp, by = c("chrom", "bin"))

  }

}

M = cor(fragCount %>% select(-c("chrom", "bin")) %>% log2(), use = "complete.obs")


corrplot(M, method = "color", outline = T, addgrid.col = "darkgray", order="hclust", addrect =
3, rect.col = "black", rect.lwd = 3,cl.pos = "b", tl.col = "indianred4", tl.cex = 1, cl.cex =
1, addCoef.col = "black", number.digits = 2, number.cex = 1, col = colorRampPalette(c("midnigh
tblue","white","darkred"))(100))
```




## 5. Spike-in calibration (optional) **(Practical 7)**

This step is highly recommended for yeast spike-ins but might be skipped if the carry-over E.coli (i.e. CUT&Tag experiments) is too low to apply.

The spike-in step is meant to calibrate the success of your sequencing experiment. This stems from the assumption that the ratio of fragments mapped to the yeast genome will be the same for all samples in different CUT&RUN experiments (the same amount of yeast is spiked in every sample). Then, there is no need to normalise between experiments or pA/G-MNase and yeast DNA batches.

A constant C is used to avoid small fractions in normalised data. This constant is an arbitrary multiplier; typically, a 10,000 value is applied.

We define a scaling factor S as:

```
S = C / (fragments mapped to E. coli genome)
```

Normalised coverage is then calculated as follows:

```
Normalized coverage = (primary_genome_coverage) * S
```

```
##linux##
if [[ "$seqDepth" -gt "1" ]]; then

    mkdir -p $yourPath/alignment/bedgraph

    scale_factor=`echo "10000 / $seqDepth" | bc -l`

    echo "Scaling factor for $Name is: $scale_factor!"

    bedtools genomecov -bg -scale $scale factor -i $yourPath/alignment/bed/${Name} bowtie2.fra
gments.bed -g $chromSize > $yourPath/alignment/bedgraph/${Name} bowtie2.fragments.normalized.b
edgraph

fi
```

### 5.1. Scaling factor

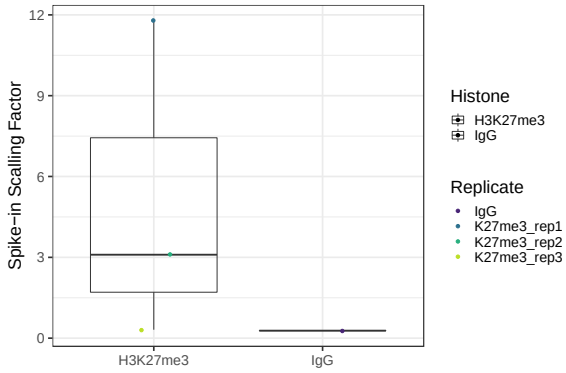
```
##R##
scaleFactor = c()
for(hist in sampleList){

  spikeDepth = read.table(paste0(yourPath, "/alignment/sam/bowtie2_summary/", hist, "_bowtie2
_spikeIn.seqDepth"), header = FALSE, fill = TRUE)$V1[1]

  histInfo = strsplit(hist, "_")[[1]]

  RepliInfo = RepliList[which(hist == sampleList)]

  scaleFactor = data.frame(scaleFactor = 10000/spikeDepth, Histone = histInfo[1], Replicate =
RepliInfo[1]) %>% rbind(scaleFactor, .))

}

scaleFactor$Histone = factor(scaleFactor$Histone, levels = histList)

scaleFactor %>% mutate(scaleFactor = paste0(scaleFactor, "%"))
```

The results will look like this:

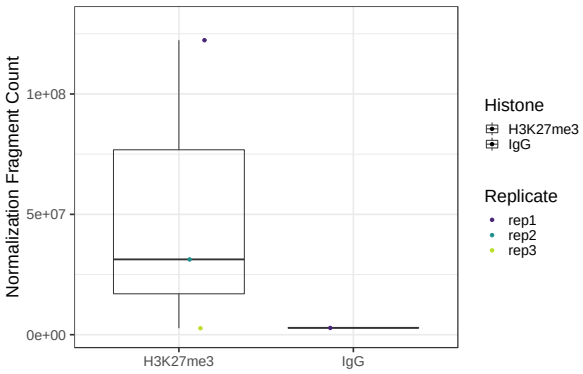
| | scaleFactor | Histone | Replicate |
|---|---|---|---|
| 1 | 11.7785630153121% | H3K27me3 | K27me3_rep1 |
| 2 | 3.09693403530505% | H3K27me3 | K27me3_rep2 |
| 3 | 0.312265800649513% | H3K27me3 | K27me3_rep3 |
| 4 | 0.278086763070078% | IgG | IgG |

```
##R##
## Generate sequencing depth boxplot

fig7 = scaleFactor %>% ggplot(aes(x = Histone, y = scaleFactor, fill = Histone)) +

    geom_boxplot() +

    geom_jitter(aes(color = Replicate), position = position_jitter(0.15)) +

    scale_fill_viridis(discrete = TRUE, begin = 0.1, end = 0.9, option = "magma", alpha = 0.8)
+

    scale_color_viridis(discrete = TRUE, begin = 0.1, end = 0.9) +

    theme_bw(base_size = 20) +

    ylab("Spike-in Scaling Factor") +

    xlab("")

normDepth = inner_join(scaleFactor, alignResult, by = c("Histone", "Replicate")) %>% mutate(no
rmDepth = MappedFragNum_hg38 * scaleFactor)
```

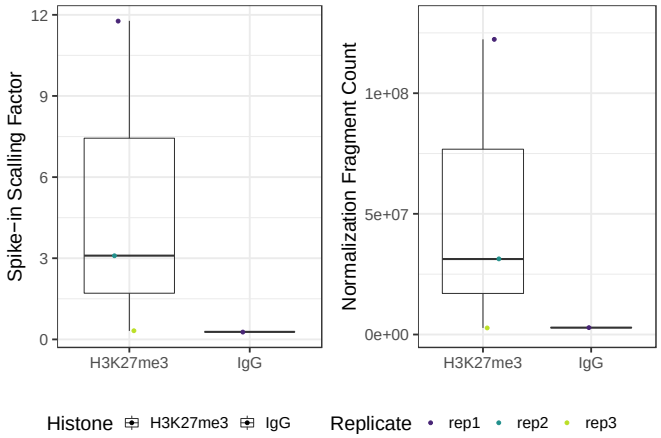
| | scaleFactor | Histone | Replicate | SequencingDepth | MappedFragNum_hg38 | AlignmentRate_hg38 | normDepth |
|---|---|---|---|---|---|---|---|
| 1 | 11.7785630 | H3K27me3 | rep1 | 10777771 | 10385487 | 96.36 | 122326113 |
| 2 | 3.0969340 | H3K27me3 | rep2 | 10369182 | 10103700 | 97.44 | 31290492 |
| 3 | 0.3122658 | H3K27me3 | rep3 | 9047596 | 8688978 | 96.04 | 2713271 |
| 4 | 0.2780868 | IgG | rep1 | 11166395 | 10248440 | 91.78 | 2849956 |




```
fig8 = normDepth %>% ggplot(aes(x = Histone, y = normDepth, fill = Histone)) +
    geom_boxplot() +
    geom_jitter(aes(color = Replicate), position = position_jitter(0.15)) +
    scale_fill_viridis(discrete = TRUE, begin = 0.1, end = 0.9, option = "magma", alpha = 0.8)
+
    scale_color_viridis(discrete = TRUE, begin = 0.1, end = 0.9) +
    theme_bw(base_size = 20) +
    ylab("Normalization Fragment Count") +
    xlab("") +
    coord_cartesian(ylim = c(1000000, 130000000))
```




```
ggarrange(fig7, fig8, ncol = 2, common.legend = TRUE, legend="bottom")
```



Note that the box plot is not visible in the IgG sample because there is just one replicate for this condition. The code is also meant to work for your multiple histone/TFs marks.

## 6. Peak calling **(Practical 8)**

Traditionally, people have used and use MACS2 for calling peaks as a legacy peak caller from ChIP-seq experiments. Remember that although MACS2 works well with multiple samples and controls, replicates, and Input backgrounds, it does not perform well with low backgrounds like those found in CUT&RUN and CUT&Tag experiments.

A tool called SEACR (Sparse Enrichment Analysis for CUT&RUN) has been developed and described by Meers et al., 2019. This peak caller has been designed to perform well with low backgrounds as the sparse CUT&RUN data in which the background is dominated by zeroes (i.e. regions without rad coverage). It also has a web interface to perform the peak calling analysis. Keep in mind that it does not handle multiple samples, controls and replicates must be merged beforehand, and it is very sensitive to negative control signals.

### 6.1. SEACR

SEACR requires bedGRaph files from paired-end libraries as input and will define a peak as contiguous blocks of basepair coverage that do not overlap with blocks of background signal (IgG control dataset). It can be used to determine narrow peaks from factor binding sites or broader peaks from histone modifications.

**Important:** if you have normalised data (as normalised fragment counts from the yeast spike-in), use the normalisation option to `non`. Otherwise, use the option `norm`.

```
##linux##

seacr="/path/to/SEACR/SEACR_1.3.sh"

Control=$2

mkdir -p $yourPath/peakCalling/SEACR


bash $seacr $yourPath/alignment/bedgraph/${Name}_bowtie2.fragments.normalized.bedgraph \

     $yourPath/alignment/bedgraph/${Control}_bowtie2.fragments.normalized.bedgraph \

     non stringent $yourPath/peakCalling/SEACR/${Name}_seacr_control.peaks

# top peaks:

bash $seacr $yourPath/alignment/bedgraph/${Name}_bowtie2.fragments.normalized.bedgraph 0.01 no
n stringent $yourPath/peakCalling/SEACR/${Name}_seacr_top0.01.peaks

Examples of use:

          bash SEACR_1.3.sh target.bedgraph IgG.bedgraph norm stringent output

          Calls enriched regions in target data using normalised IgG control track with string
ent threshold

          bash SEACR_1.3.sh target.bedgraph IgG.bedgraph non relaxed output

          Calls enriched regions in target data using non-normalised IgG control track with re
laxed threshold

          bash SEACR_1.3.sh target.bedgraph 0.01 non stringent output

          Calls enriched regions in target data by selecting the top 1% of regions by the area
under the curve (AUC)
```

### 6.1.1. Summary of called peaks

```
##R##
peakN = c()

peakWidth = c()

peakType = c("control", "top0.01")

for(hist in sampleList){

  histInfo = strsplit(hist, "_")[[1]]

  if(histInfo[1] != "IgG"){

    for(type in peakType){

      peakInfo = read.table(paste0(yourPath, "/peakCalling/SEACR/", hist, "_seacr_", type, ".p
eaks.stringent.bed"), header = FALSE, fill = TRUE)  %>% mutate(width = abs(V3-V2))

      peakN = data.frame(peakN = nrow(peakInfo), peakType = type, Histone = histInfo[1], Repli
cate = histInfo[2]) %>% rbind(peakN, .)

      peakWidth = data.frame(width = peakInfo$width, peakType = type, Histone = histInfo[1], R
eplicate = histInfo[2])  %>% rbind(peakWidth, .)

    }

  }

}

peakN %>% select(Histone, Replicate, peakType, peakN)
```

| | Histone | Replicate | peakType | peakN |
|---|---|---|---|---|
| 1 | H3K27me3 | rep1 | control | 104528 |
| 2 | H3K27me3 | rep1 | top0.01 | 13540 |
| 3 | H3K27me3 | rep2 | control | 29371 |
| 4 | H3K27me3 | rep2 | top0.01 | 13337 |
| 5 | H3K27me3 | rep3 | control | 5948 |
| 6 | H3K27me3 | rep3 | top0.01 | 13426 |

### 6.1.2. Reproducibility of the peaks

A way to confirm the reproducibility of the peaks called across biological replicates is to use the top 1% peaks (ranked by the total signal in each block, higher signal values) as high-confidence peaks.

```
##R##

histL = c("H3K27me3")

repL = paste0("rep", 1:2)

peakType = c("control", "top0.01")

peakOverlap = c()

for(type in peakType){

  for(hist in histL){

    overlap.gr = GRanges()

    for(rep in repL){

      peakInfo = read.table(paste0(yourPath, "/peakCalling/SEACR/", hist, "_", rep, "_seacr_",
type, ".peaks.stringent.bed"), header = FALSE, fill = TRUE)

      peakInfo.gr = GRanges(peakInfo$V1, IRanges(start = peakInfo$V2, end = peakInfo$V3), stra
nd = "*")
```

```
      if(length(overlap.gr) >0){

        overlap.gr = overlap.gr[findOverlaps(overlap.gr, peakInfo.gr)@from]

      }else{

        overlap.gr = peakInfo.gr


      }

    }

    peakOverlap = data.frame(peakReprod = length(overlap.gr), Histone = hist, peakType = type)
%>% rbind(peakOverlap, .)

  }

}


peakReprod = left_join(peakN, peakOverlap, by = c("Histone", "peakType")) %>% mutate(peakRepro
dRate = peakReprod/peakN * 100)

peakReprod %>% select(Histone, Replicate, peakType, peakN, peakReprodNum = peakReprod, peakRep
rodRate)
```

| | Histone | Replicate | peakType | peakN | peakReprodNum | peakReprodRate |
|---|---|---|---|---|---|---|
| 1 | H3K27me3 | rep1 | control | 104528 | 38960 | 37.27231 |
| 2 | H3K27me3 | rep1 | top0.01 | 13540 | 11646 | 86.01182 |
| 3 | H3K27me3 | rep2 | control | 29371 | 38960 | 132.64785 |
| 4 | H3K27me3 | rep2 | top0.01 | 13337 | 11646 | 87.32099 |
| 5 | H3K27me3 | rep3 | control | 5948 | 38960 | 655.01009 |
| 6 | H3K27me3 | rep3 | top0.01 | 13426 | 11646 | 86.74214 |

The reproducibility number is sensitive to the total number of called peaks in each replicate, and it is calculated as:

```
[#peaks overlapping all biological replicates/ #peaks of rep1 or rep2 or rep3] * 100
```

### 6.1.3. Calculate the FRagment proportion in Peaks regions (FRiPs) **(Practical 9)**

As a measure of signal-to-noise, compute the fraction of reads in peaks (FRiPs) and compare it to IgG control FRiPs. Remember that although sequencing depths in CUT&RUN experiments are typically low, 1-5 million paired-end reads, the low background translates into high FRiP scores.

```
##R##
library(chromVAR)

bamDir = paste0(yourPath, "/alignment/bam")

inPeakData = c()

## overlap with bam file to get count

for(hist in histL){

  for(rep in repL){

    peakRes = read.table(paste0(yourPath, "/peakCalling/SEACR/", hist, " ", rep, " seacr contr
ol.peaks.stringent.bed"), header = FALSE, fill = TRUE)

    peak.gr = GRanges(seqnames = peakRes$V1, IRanges(start = peakRes$V2, end = peakRes$V3), st
rand = "*")

    bamFile = paste0(bamDir, "/", hist, "_", rep, "_bowtie2.mapped.bam")
```

```
    fragment_counts <- getCounts(bamFile, peak.gr, paired = TRUE, by_rg = FALSE, format = "bam
")

    inPeakN = counts(fragment_counts)[,1] %>% sum

    inPeakData = rbind(inPeakData, data.frame(inPeakN = inPeakN, Histone = hist, Replicate = r
ep))

  }

}


frip = left_join(inPeakData, alignResult, by = c("Histone", "Replicate")) %>% mutate(frip = in
PeakN/MappedFragNum_hg38 * 100)

frip %>% select(Histone, Replicate, SequencingDepth, MappedFragNum hg38, AlignmentRate hg38, F
ragInPeakNum = inPeakN, FRiPs = frip)
```

Replicate 1 vs Replicate 2:

| | Histone | Replicate | SequencingDepth | MappedFragNum_hg38 | AlignmentRate_hg38 | FragInPeakNum | FRiPs |
|---|---|---|---|---|---|---|---|
| 1 | H3K27me3 | rep1 | 10777771 | 10385487 | 96.36 | 6923176 | 66.66203 |
| 2 | H3K27me3 | rep2 | 10369182 | 10103700 | 97.44 | 5011246 | 49.59813 |

Replicate 1 vs Replicate 3:

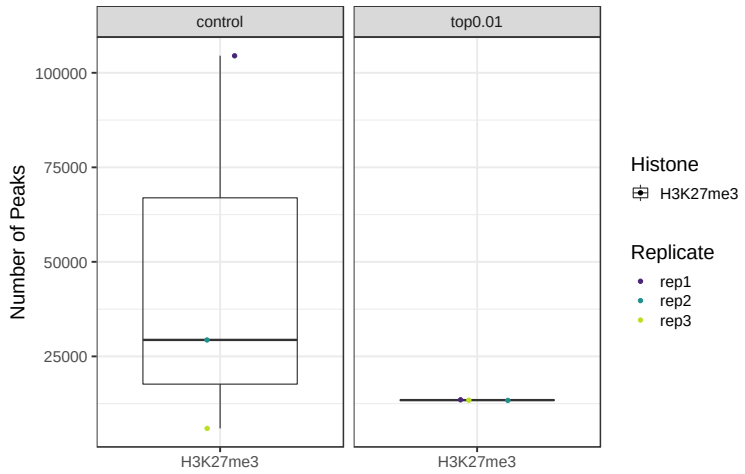
| | Histone | Replicate | SequencingDepth | MappedFragNum_hg38 | AlignmentRate_hg38 | FragInPeakNum | FRiPs |
|---|---|---|---|---|---|---|---|
| 1 | H3K27me3 | rep1 | 10777771 | 10385487 | 96.36 | 6923176 | 66.66203 |
| 2 | H3K27me3 | rep2 | 10369182 | 10103700 | 97.44 | 4196943 | 41.53867 |

Replicate 2 vs Replicate 3:

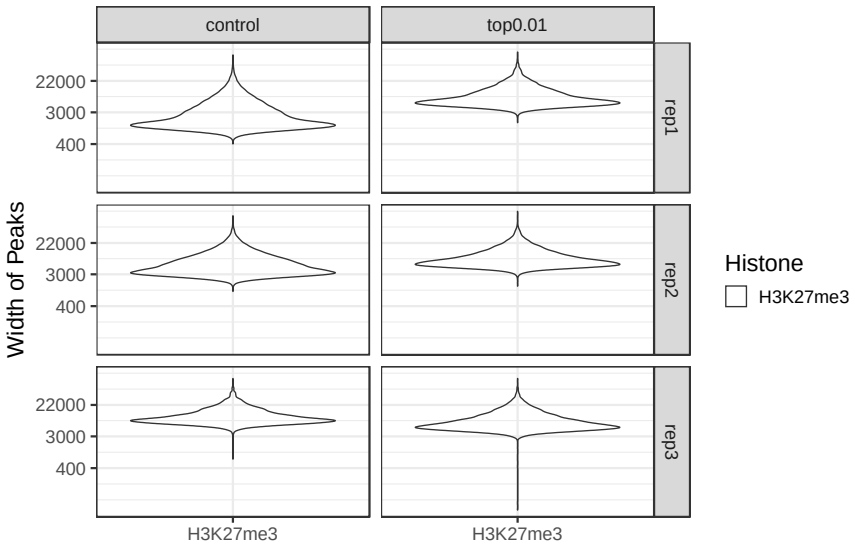
| | Histone | Replicate | SequencingDepth | MappedFragNum_hg38 | AlignmentRate_hg38 | FragInPeakNum | FRiPs |
|---|---|---|---|---|---|---|---|
| 1 | H3K27me3 | rep1 | 10777771 | 10385487 | 96.36 | 6867802 | 66.12884 |
| 2 | H3K27me3 | rep2 | 10369182 | 10103700 | 97.44 | 4196943 | 41.53867 |

### 6.1.4.Visualisation of peak number, peak width, peak reproducibility and FRiPs

```
fig9 = peakN %>% ggplot(aes(x = Histone, y = peakN, fill = Histone)) +

    geom_boxplot() +

    geom_jitter(aes(color = Replicate), position = position_jitter(0.15)) +

    facet_grid(~peakType) +

    scale_fill_viridis(discrete = TRUE, begin = 0.1, end = 0.55, option = "magma", alpha = 0.8
) +

    scale_color_viridis(discrete = TRUE, begin = 0.1, end = 0.9) +

    theme_bw(base_size = 18) +

    ylab("Number of Peaks") +

    xlab("")
```

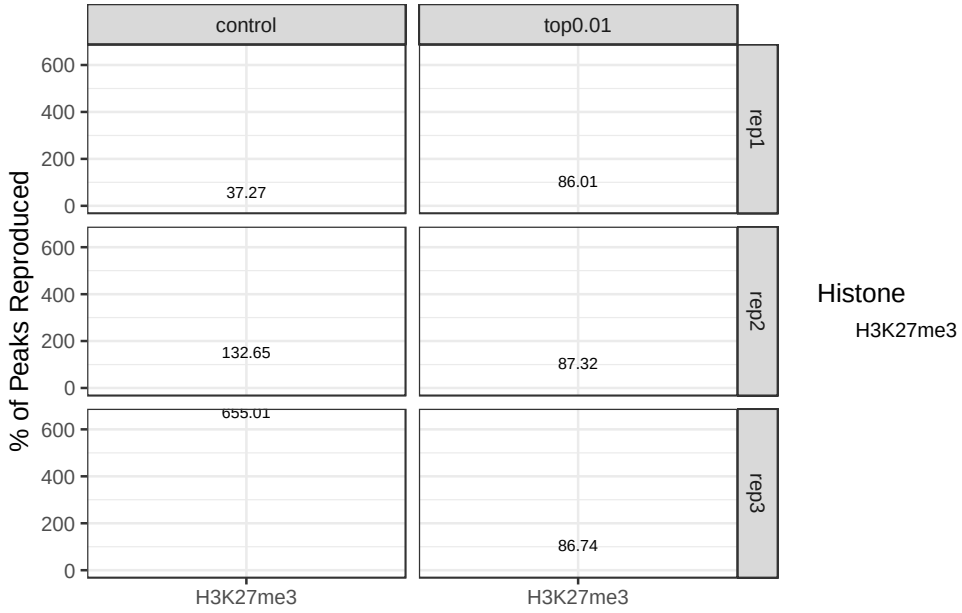



```
fig10 = peakWidth %>% ggplot(aes(x = Histone, y = width, fill = Histone)) +
    geom_violin() +
    facet_grid(Replicate~peakType) +
    scale_fill_viridis(discrete = TRUE, begin = 0.1, end = 0.55, option = "magma", alpha = 0.8
) +
    scale_color_viridis(discrete = TRUE, begin = 0.1, end = 0.9) +
    scale_y_continuous(trans = "log", breaks = c(400, 3000, 22000)) +
    theme_bw(base_size = 18) +
    ylab("Width of Peaks") +
    xlab("")
```

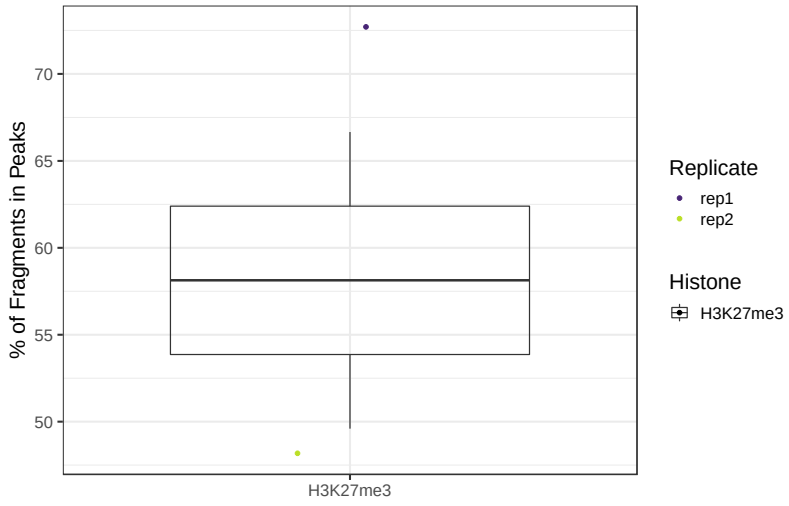



```
fig11 = peakReprod %>% ggplot(aes(x = Histone, y = peakReprodRate, fill = Histone, label = rou
nd(peakReprodRate, 2))) +
    geom_bar(stat = "identity") +
    geom_text(vjust = 0.1) +
    facet_grid(Replicate~peakType) +
    scale_fill_viridis(discrete = TRUE, begin = 0.1, end = 0.55, option = "magma", alpha = 0.8
) +
```

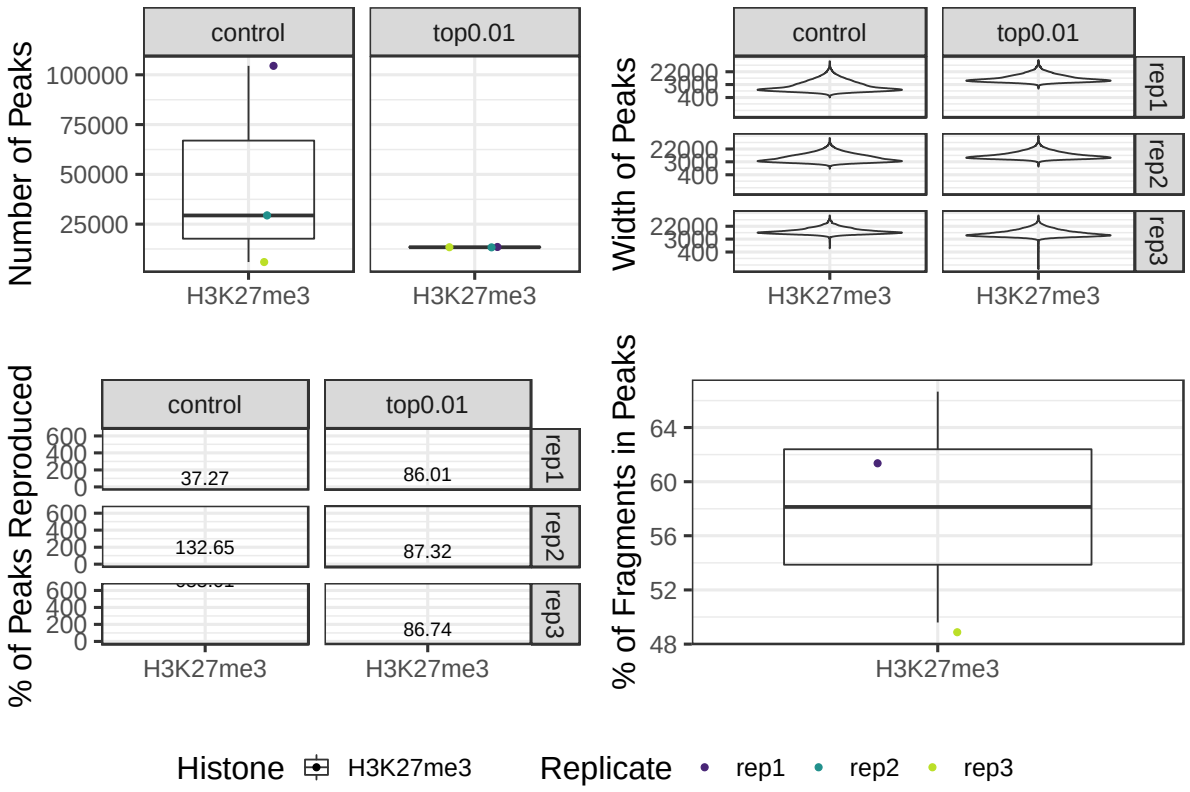
```
    scale_color_viridis(discrete = TRUE, begin = 0.1, end = 0.9) +
    theme_bw(base_size = 18) +
    ylab("% of Peaks Reproduced") +
    xlab("")
```




```
fig12 = frip %>% ggplot(aes(x = Histone, y = frip, fill = Histone, label = round(frip, 2))) +
    geom_boxplot() +
    geom_jitter(aes(color = Replicate), position = position_jitter(0.15)) +
    scale_fill_viridis(discrete = TRUE, begin = 0.1, end = 0.55, option = "magma", alpha = 0.8
) +
    scale_color_viridis(discrete = TRUE, begin = 0.1, end = 0.9) +
    theme_bw(base_size = 18) +
    ylab("% of Fragments in Peaks") +
    xlab("")
```



```
ggarrange(fig9, fig10, fig11, fig12, ncol = 2, nrow=2, common.legend = TRUE, legend="bottom")
```




## 7. Data visualisation **(Practical 10)**

Use a genome browser for chromatin landscape visualisation using the Integrative Genomic Viewer (IGV) locally (local desktop version) or on the web. Or UCSC Genome Browser to add supplementary datasets to complement your information.

### 7.1. Browser display of normalised bedgraph files

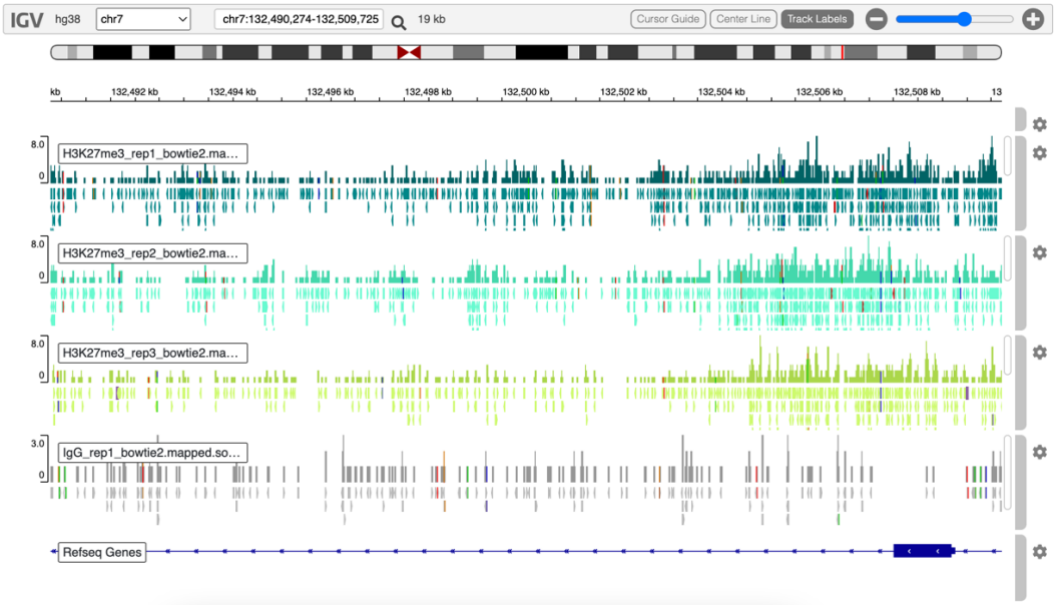



*Figure 3*. The chr7:132,490,274-132,509,725 region shows H3K27me3 profile.

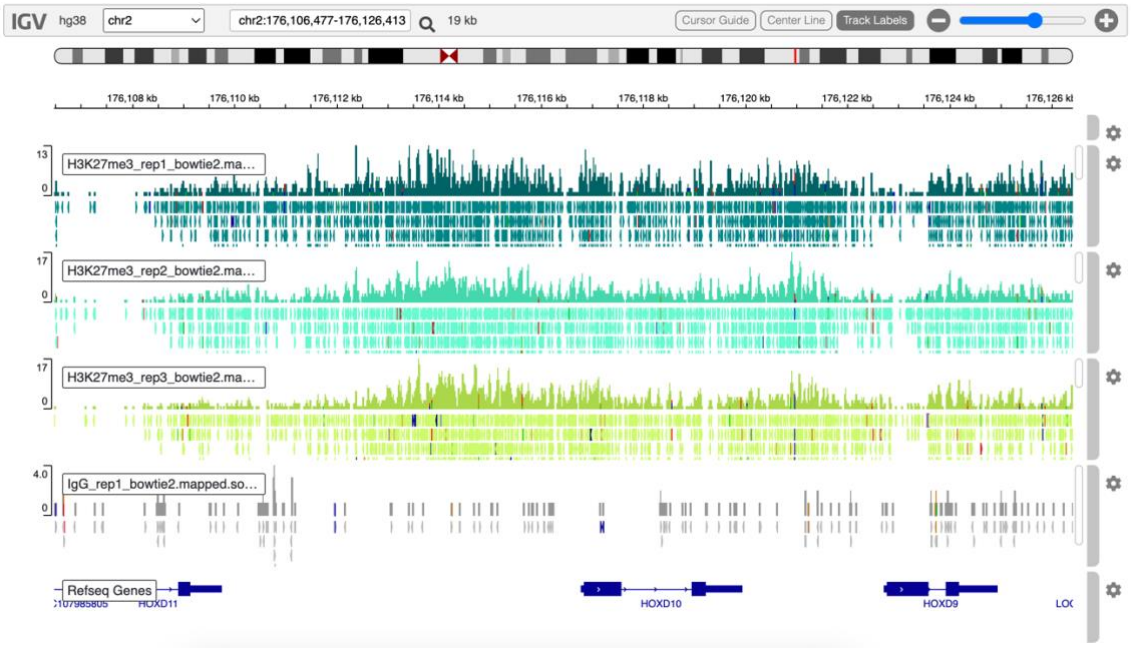



*Figure 4*. Part of the HOXD cluster showing its H3K27me3 profile

### 7.2. Heatmap of specific regions **(Practical 11)**

Check chromatin features at a list of annotated sites, such as gene promoters. For this, we will use deepTools and its `computeMatrix` and `plotHeatmap` functions.

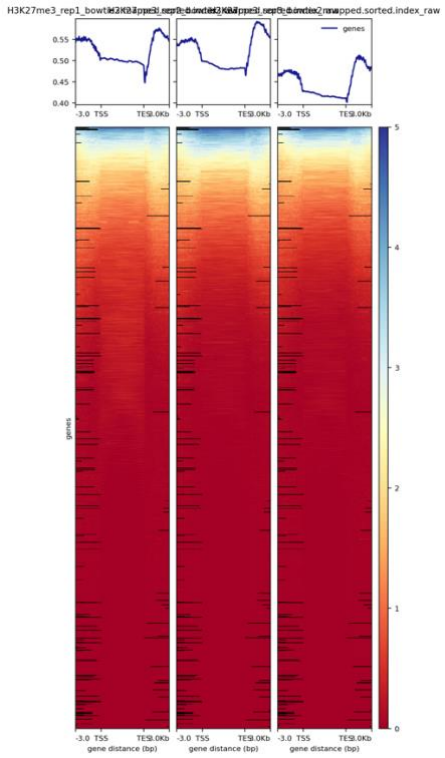
```
##linux##

mkdir -p $yourPath/alignment/bigwig

samtools sort -o $yourPath/alignment/bam/${Name}.sorted.bam $yourPath/alignment/bam/${Name}_bo
wtie2.mapped.bam

samtools index $yourPath/alignment/bam/${Name}.sorted.bam

bamCoverage -b $yourPath/${Name}.sorted.bam -o $yourPath/${Name}_raw.bw
```

#### 7.2.1.Heatmap over transcription units

Get promoters at UCSC browser.

```
##linux##

cores=8

computeMatrix scale-regions -S $yourPath/alignment/bigwig/H3K27me3_rep1_raw.bw \
                               $yourPath/alignment/bigwig/H3K27me3_rep2_raw.bw \
                               $yourPath/alignment/bigwig/H3K27me3_rep3_raw.bw \
                              -R $yourPath/data/hg38_gene/promoters\
                              --beforeRegionStartLength 3000 \
                              --regionBodyLength 5000 \
                              --afterRegionStartLength 3000 \
                              --skipZeros -o $yourPath/data/hg38_gene/matrix_gene.mat.gz -p $c
ores
```

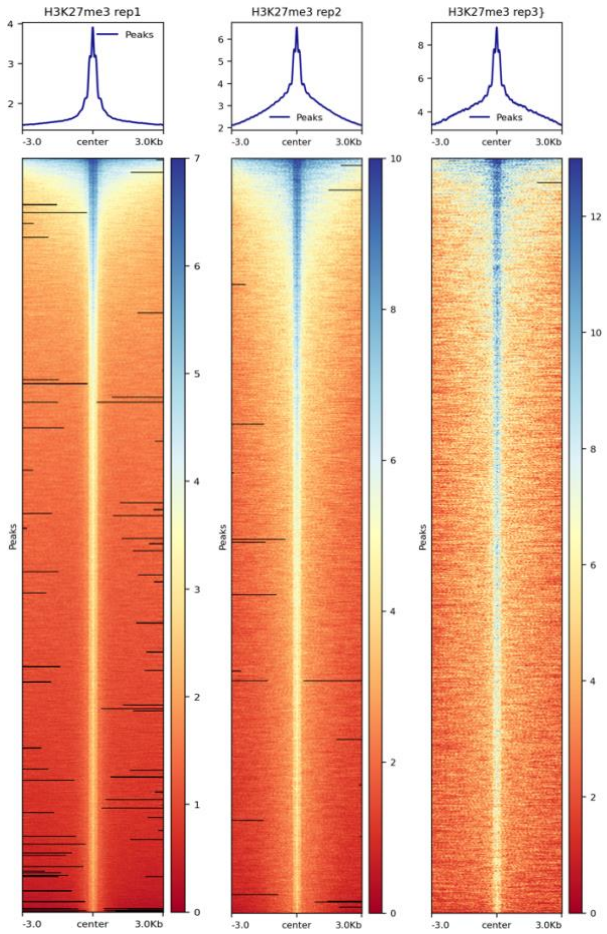
```
plotHeatmap -m $yourPath/data/hg38_gene/matrix_gene.mat.gz -out $yourPath/data/hg38_gene/Histo
ne_gene.png --sortUsing sum
```




### 7.2.2. Heatmap on CUT&RUN peaks **(Practical 12)**

In this section, we will extract the information from column 6 at the SEACR output. This column includes an entry for the localisation of the region with the maximum signal (in the form chr:start-end). We will use the signal block midpoint from the SEACR output to align signals in the heatmaps.

```
##linux##

awk '{split($6, summit, ":"); split(summit[2], region, "-"); print summit[1]"\t"region[1]"\t"r
egion[2]}' $yourPath/${Name}_${repName}_seacr_control.peaks.stringent.bed >$yourPath/${Name}_$
{repName}_seacr_control.peaks.summitRegion.bed


computeMatrix reference-point -S $yourPath/alignment/bigwig/${Name}_${repName}_raw.bw \

              -R $yourPath/${Name}_${repName}_seacr_control.peaks.summitRegion.bed \

              --skipZeros -o $yourPath/${Name}_${repName}_SEACR.mat.gz -p $cores -a 3000 -b 30
00 --referencePoint center


plotHeatmap -m $yourPath/${Name}_SEACR.mat.gz -out $yourPath/${Name}_SEACR_heatmap.png --sortU
sing sum --startLabel "Peak Start" -\

-endLabel "Peak End" --xAxisLabel "" --regionsLabel "Peaks" --samplesLabel "${Name} ${repName}
"
```




## 8. Differential peak analysis **(Practical 13)**

We will use DESeq2 to compare sequencing assays and their changes across experimental conditions. DESeq2 is a method for differential analysis of count data that focuses on a more quantitative analysis of the strength of the difference rather than the existence of differential expression. The method relies on shrinking dispersion estimation and on fold changes to improve the stability/interpretation of estimates

### 8.1. Generate the peak sample matrix

Generally, the differential test compares two or more conditions (i.e. control versus treated). The complete DESeq2 tutorial can be found here.

#### 8.1.1. Generate a master peak list containing all the called peaks (in all replicates)

```
##R##
mPeak = GRanges()

## Overlap with bam file to get count

for(hist in histL){

  for(rep in repL){

    peakRes = read.table(paste0(yourPath, "/peakCalling/SEACR/", hist, "_", rep, "_seacr_contr
ol.peaks.stringent.bed"), header = FALSE, fill = TRUE)

    mPeak = GRanges(seqnames = peakRes$V1, IRanges(start = peakRes$V2, end = peakRes$V3), stra
nd = "*") %>% append(mPeak, .)
```

```
  }
}
masterPeak = reduce(mPeak)
```

### 8.1.2. Get the fragment counts for each peak in the master peak list

```
##R##
library(DESeq2)
bamDir = paste0(yourPath, "/alignment/bam/")
countMat = matrix(NA, length(masterPeak), length(histL)*length(repL))
## Overlap with bam file to get count
i = 1
for(hist in histL){
  for(rep in repL){

    bamFile = paste0(bamDir, "/", hist, "_", rep, "_bowtie2.mapped.bam")
    fragment_counts <- getCounts(bamFile, masterPeak, paired = TRUE, by_rg = FALSE, format = "
bam")
    countMat[, i] = counts(fragment_counts)[,1]
     i = i + 1
  }
}
colnames(countMat) = paste(rep(histL, 2), rep(repL, each = 1), sep = "_")
```

### 8.1.3. Perform sequencing depth normalisation and differential enriched peaks detection

```
##R##
selectR = which(rowSums(countMat) > 5) ## remove low count genes
dataS = countMat[selectR,]
condition = factor(rep(histL, each = length(repL)))
dds = DESeqDataSetFromMatrix(countData = dataS,
                             colData = DataFrame(condition),
                             design = ~ condition)# here should be 1 if replicates, condition
if different histones
DDS = DESeq(dds)
normDDS = counts(DDS, normalized = TRUE) ## normalization with respect to the sequencing depth
colnames(normDDS) = paste0(colnames(normDDS), "_norm")
res = results(DDS, independentFiltering = FALSE, altHypothesis = "greaterAbs")

countMatDiff = cbind(dataS, normDDS, res)
head(countMatDiff)
```

```
DataFrame with 6 rows and 10 columns
  H3K27me3_rep1 H3K27me3_rep2 H3K27me3_rep1_norm H3K27me3_rep2_norm  baseMean log2FoldChange     lfcSE      stat      pvalue        padj
      <numeric>     <numeric>          <numeric>          <numeric> <numeric>      <numeric> <numeric> <numeric>   <numeric>   <numeric>
1            15            15            14.2678            15.76972  15.01879        3.90870  0.514198   7.60154 2.92621e-14 4.52663e-14
2            40            28            38.0476            29.43682  33.74220        5.07648  0.355510  14.27945 2.93933e-46 7.67597e-46
3            21            16            19.9750            16.82104  18.39801        4.20148  0.467809   8.98117 2.67895e-19 4.76985e-19
4            36            41            34.2428            43.10391  38.67337        5.27327  0.333992  15.78860 3.72779e-56 1.06551e-55
5            12             6            11.4143             6.30789   8.86108        3.14748  0.671505   4.68720 2.76962e-06 3.00255e-06
6            14            10            13.3167            10.51315  11.91490        3.57470  0.575835   6.20784 5.37165e-10 6.89055e-10
```

*DESeq2:* The input matrix needs to have unnormalised counts or estimated counts of sequencing reads. It is meant to compare different conditions (i.e. different histone marks), but in this example, `design` had a single variable (H3K27me3) with all samples having the same value, and it had to be set at ~1. The DESeq2 model corrects internally for library size.

How to interpret countMatDiff results:

- The first four columns = raw read counts after filtering peak regions with low counts.
- The following four columns = normalised read counts. Library size differences are removed.
- The remaining columns = differential detection results.

## 9. Other ways to make the calculations

### 9.1. Data normalisation without spike-in DNA

ChIPseqSpikeInFree is a method to determine scaling factors across different conditions/treatments in ChIP-seq experiments.
This method does not include spike-in chromatin or peak detection steps to show global changes in histone modification profiles.

### 9.2. Peak calling

The most common alternative to SEACR is the use of MACS2. MACS2 allows for peak calling with or without replicates and with or without control IgG.

```
##linux##
Name="H3K27me3"

controlName="IgG"


mkdir -p $yourPath/peakCalling

macs2 callpeak -t ${yourPath}/alignment/bam/${Name}_rep1_bowtie2.mapped.bam \

      -c ${yourPath}/alignment/bam/${controlName}_rep1_bowtie2.mapped.bam \

      -g hs -f BAMPE -n macs2_peak_q0.1 --outdir $yourPath/peakCalling/MACS2 -q 0.1 --keep-dup
all 2>${yourPath}/peakCalling/MACS2/macs2Peak_summary.txt
```

Other alternatives to SEACR and MACS2 are dPeak and MOSAiCs.

### 9.3. Differential peak analysis

Other alternatives to DESeq2 are Limma and edgeR. Limma is an R package designed to analyse microarray data by applying linear models to differential expression analysis. Apart from analysing comparisons between many RNA targets simultaneously, it can be used to study differential fragment enrichment within peak regions. edgeR was designed to perform differential expression analysis of RNA-seq data with biological replicates. It includes a set of

statistical methods based on negative binomial distributions. Apart from RNA-seq analysis, it can be applied to differential signal analysis of other types of genomic data with read counts, such as ChIP-seq, ATAC-seq, CAGE-seq, SAGE-seq or Bisulfite-seq.

### 9.4. Pipelines for CUT&RUN data analysis

Alternatively, there are other pipelines to analyse CUT&RUN data, such as nf-core/cutandrun (from The Francis Crick Institute, adapted from Henikoff's pipeline) or the CUT&RUNTools more oriented toward the identification of chromatin-associated protein binding and genomic footprinting from antibody-targeted CUT&RUN.