ATAC-Seq: QC, peak calling, differential accessibility

AUTHOR
Florian Geier

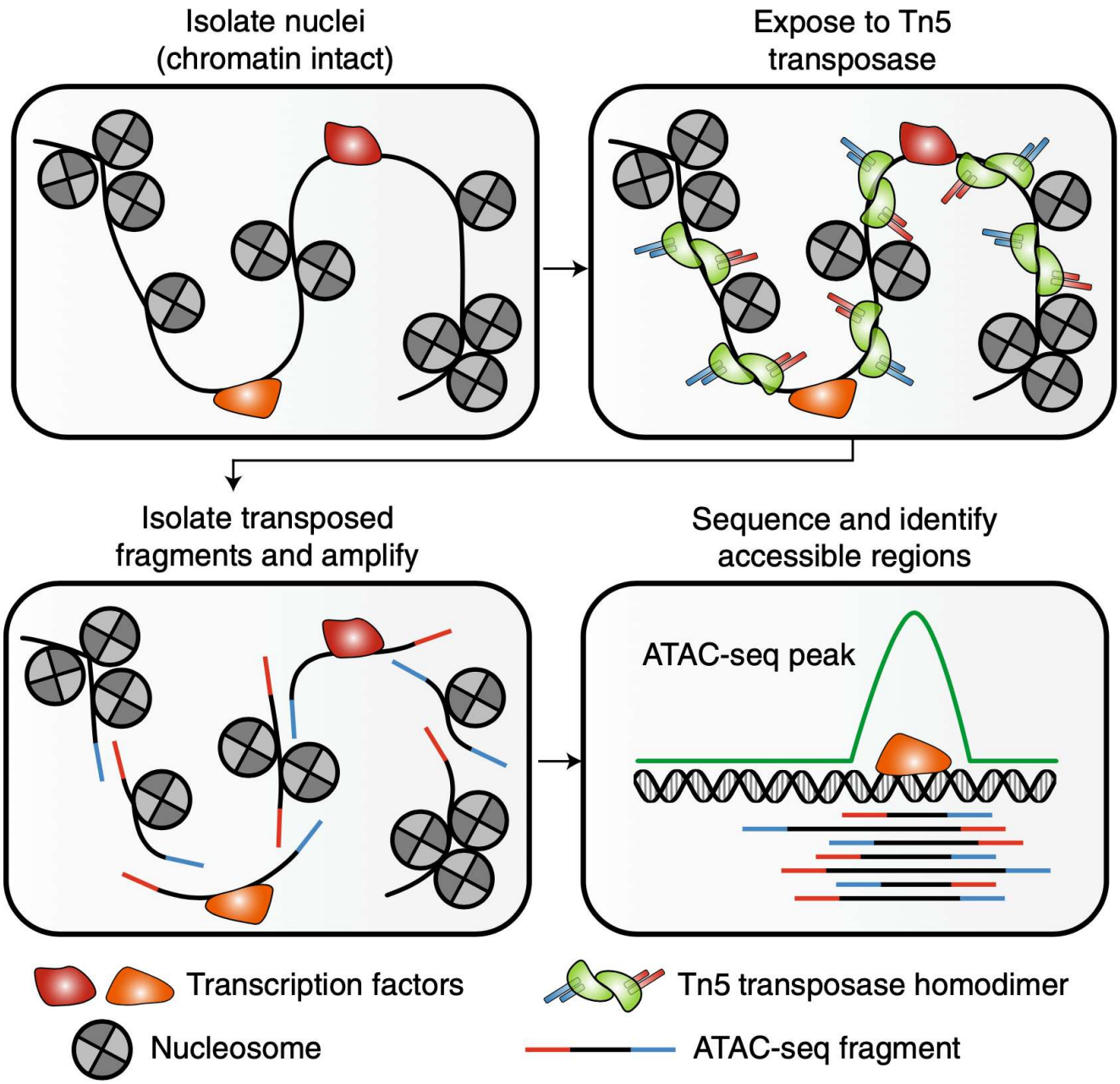
PUBLISHED
May 14, 2025

Overview

Goal

- Identify regions in the genome of open chromatin (“ATAC-Seq peaks”)
- Find subsets which are specifically open or closed in one of the experimental conditions or cell types
- Look at ATAC-Seq specific biases and learn how to deal with them

ATAC-Seq – Assay for Transposase-Accessible Chromatin using Sequencing



Source: Grandi et al. 2022

- The ATAC-Seq assay was developed around 2013 in the Greenleaf lab and is based on two main observations:
    - a Tn5 transposase preloaded with sequencing adapters can be used to simultaneously fragment and tag genomic DNA for high-throughput sequencing library construction
    - Tn5 efficiently inserts into nucleosome-free regions
- The assays identifies regions of open chromatin (also called accessible regions). Accessibility is a hallmark of active gene regulatory elements such as enhancers, promoters and insulators.
- As ATAC-Seq does not focus on specific epigenetic marks, such as H3K27ac in ChIP-Seq, it is a more unbiased assay to study epigenetic regulation.
- In high resolution, ATAC-Seq data also enables the study of nucleosome positioning and transcription factor footprints.
- Due to the simplified library construction, bulk ATAC-Seq can be performed with low amounts of starting material (DNA from 500 - 75'000 cells). Its even possible to perform ATAC-Seq of single

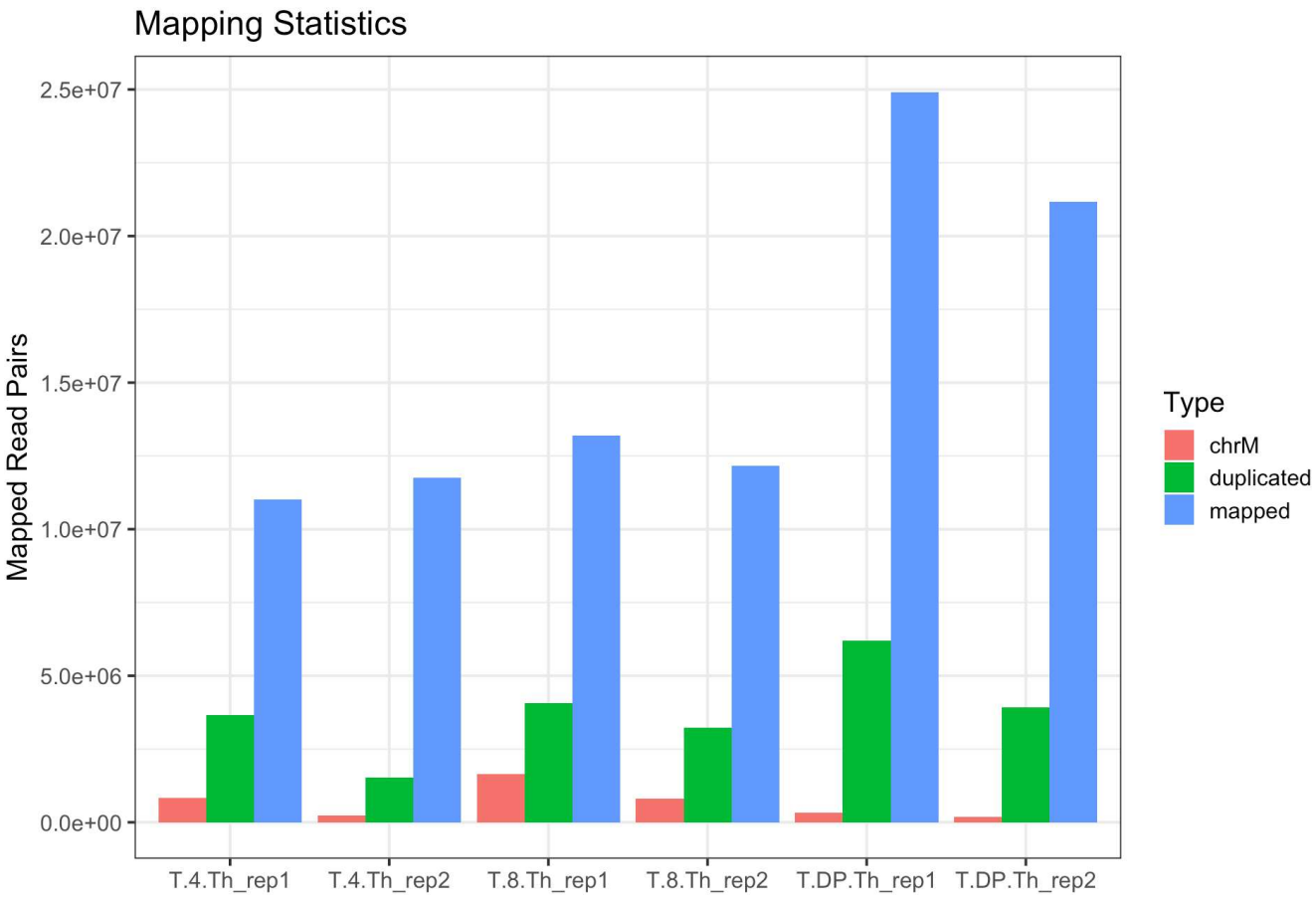
nuclei.

Example from the ImmGen ATAC-Seq data

- The immunological genome project (ImmGen) is an international research collaborative for the study of genome activity and regulation in the immune system of mouse.
- It offers several large scale data sets among which is chromatin accessibility on 86 primary immune cell types used to map out the atlas of cis-regulatory elements of the immune system (Yoshida et al. 2019).
- Raw data is deposited in GEO under GSE100738.
- We use a data subset on T cell differentiation from double-positive thymocytes to either CD4+ or CD8+ T cells.

Data Preprocessing Outside of R

- Paired-end reads were mapped with bowtie2 to the mouse genome (UCSC version mm10).
- The output was sorted and indexed with samtools.
- Duplicated reads were marked with picard.



Data analysis

- The data analysis workflow we discuss today consists of the following steps:
  - Counting reads in genomic windows
  - Perform quality control such as measuring enrichment and GC bias
  - Define accessible windows ("peak calling")
  - Identify differentially accessibility windows while accounting for bias
  - Summarize and annotate differential regions
- In R, both peak calling (based on a genome-wide windowing approach) and differential accessibility analysis are possible via the Bioconductor *csaw* package. For an extended documentation look at csawBook.
- The required R packages are:

```
# plotting
library(RColorBrewer)
library(tidyverse)
library(ggrepel)
library(ggrastr)
options(ggrastr.default.dpi=150)
library(patchwork)

# data import and structures
library(SummarizedExperiment)                # SummarizedExperiment
library(rtracklayer)                         # import/export BED files

# differential accessibility
library(csaw)                                # windowing approach
library(edgeR)                               # DGEList

# annotation
library(org.Mm.eg.db)                        # ID conversion
library(BSgenome.Mmusculus.UCSC.mm10)        # genome sequence
library(TxDb.Mmusculus.UCSC.mm10.knownGene)  # gene and transcript coordinates

# coverage tracks
library(Gviz)                                # plot coverage tracks
```

Counting reads in genomic windows

- All of the quality control and analysis we discuss focus on counts within small windows covering the entire genome. We get these counts using the `windowCounts` function from the *csaw* package.
- Its advisable to remove from the analysis some regions in the genome, typically centromere/telomere regions (not mappable), additionally dense repeat regions or other "read-attracting regions" which show a condition-independent over-representation of reads. There is a

blacklist of such regions for the mouse genome (mm10 version), available from the ENCODE project at https://github.com/Boyle-Lab/Blacklist/tree/master/lists.

- Since the `windowCounts` function from the *csaw* package discards only reads "completely contained within the supplied black-listed ranges" we have to extend them by the read size on both ends. As some of the extended regions exceed chromosome boundaries we need to additionally trim them to fit with chromosome sizes (`seqlengths`).

- We combine the extended black-listed regions with the mitochondrial (`chrM`) genome coordinates (taken from the *BSgenome.Mmusculus.UCSC.mm10* package).

```
blacklist.gr <- import("data/ATACSeq/mm10-blacklist.v2.bed.gz",
                       seqinfo=seqinfo(BSgenome.Mmusculus.UCSC.mm10))
# extension
readLength <- 38
blacklist.gr <- trim(resize(blacklist.gr,width=width(blacklist.gr) + 2*readLength, fix="center
# add mitochondrial genome to black list
chrM.gr <- GRanges("chrM",
                   IRanges(1,seqlengths(BSgenome.Mmusculus.UCSC.mm10)["chrM"]),
                   name="Mitochondrial genome")
blacklist.gr <- c(blacklist.gr, chrM.gr)
```

- Next we load information about samples and corresponding BAM files as given in the `alignments.tsv` file. Note that BAM files are available on sciCORE NextCloud. Link and password are shared via ADAM. **You only need to download the BAM files, if your want to re-do the window counting step. For all other steps two smaller `SummarizedExperiment` objects are also shared via NextCloud that contain all necessary data.**

```
# table with link to BAM files and sample annotation
alignments <- read.table("data/ATACSeq/alignments.tsv", header=TRUE, stringsAsFactors=FALSE)

# BAM files
bamFiles <- setNames(alignments$FileName, alignments$ExternalSampleName)
```

- Now we are in place for counting and just have to set a couple of parameters in the `readParam` and `windowCounts` function:

  - We treat the paired-end data as single-end (`pe="none"`). The reason is that the two ends of a single fragment are created by two "independent" Tn5 transposase cuts and we therefore treat them as two events.

  - We will remove PCR duplicates as marked by Picard (`dedup=TRUE`). Given that duplication levels are quite different, we want to level out any effect due to differences in library complexity.

  - We ignore (=do not count any reads) in black-listed regions.

  - We additionally focus only on standard chromosomes (parameter `restrict`). Unplaced or unlocalized scaffolds are not covered by the blacklist and might contain additional problematic regions.

- Within `windowCount` we define the window width as 150 bases, roughly the size of a nucleosome. There is a trade-off between spatial resolution (as controlled by window size and window shift) and library size/complexity (=the number of reads observed in individual windows) which determine the power to detect differential accessibility. To identify ATAC-Seq peaks (not footprints!) the size of the nucleosome seems a reasonable scale. However, small library sizes might call for larger windows e.g. 500 bases.
- Additionally we do not perform any extension of reads (`ext=0`). The 5' end of reads, which is counted, reflects the position of Tn5 transposase cut on the scale of our window size.
- We don't filter windows by their count size (`filter=0`). This is only done to get an idea of the un-covered fraction of the mappable genome.
- By setting the `bin=TRUE` we make sure that the windows do not overlap (`step == width`).
- The `BPPARAM` argument is used to set up parallelization of calculations (6 processors here).

```
param <- readParam(pe="none", dedup=TRUE, discard=blacklist.gr,
                   restrict=standardChromosomes(BSgenome.Mmusculus.UCSC.mm10))

windows <- windowCounts(bamFiles, ext=0, width=150,
                        bin=TRUE, filter=0, param=param,
                        BPPARAM=MulticoreParam(6))

# Add sample annotation
colData(windows) <- cbind(colData(windows), alignments)
```

- Finally, we also tabulate the frequency of guanine and cytosine nucleotides ("GC content") of each window. For the purpose of excluding assembly gaps we also tabulate the frequency of the "N" nucleotide.

```
seqs <- getSeq(BSgenome.Mmusculus.UCSC.mm10, rowRanges(windows))
rowData(windows) <- letterFrequency(seqs, letters=c('GC', 'N'), as.prob=TRUE) # G or C, N freq
```

- We remove assembly gaps (windows where more than 2/3 of the bases are N's) and also exclude windows partially overlapping blacklisted regions or chromosome ends i.e. that are not of length 150 bases.

```
is.agap <- rowData(windows)[,'N'] > 2/3 # i.e. more than 100 of 150 bases
table(is.agap)/nrow(windows)
```

```
is.agap
     FALSE       TRUE
0.97139194 0.02860806
```

```
is.smaller <- width(rowRanges(windows)) != 150
table(is.smaller)
```

```
is.smaller
   FALSE     TRUE
```

```
18170240        22
```

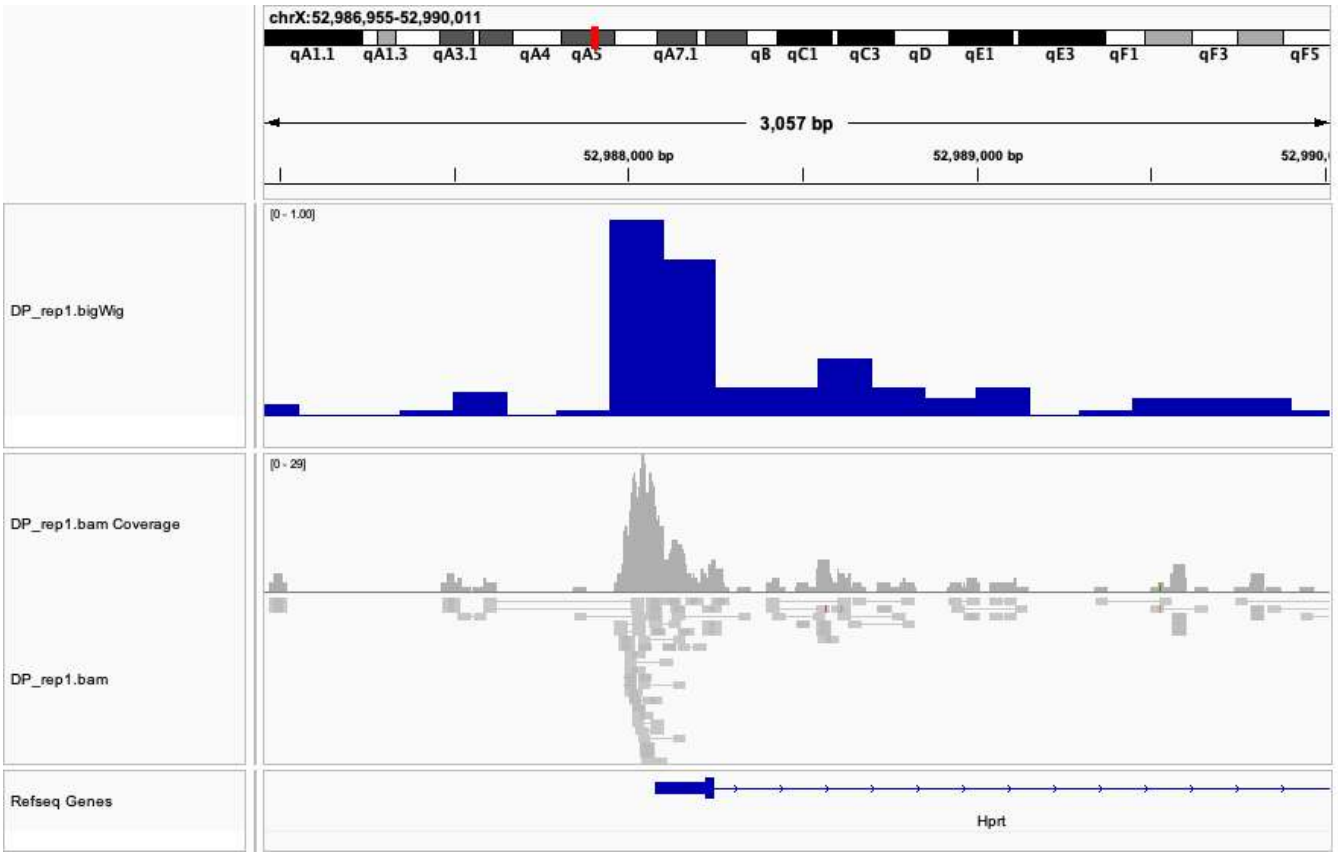
```
windows <- windows[!is.agap & !is.smaller,]
```

```
saveRDS(windows, file="data/ATACSeq/csaw_window_counts_rmDup.rds")
```

Export coverage tracks for IGV

- The window counts can be displayed as coverage tracks in genome browsers such as the Integrative Genomics Viewer (IGV).
- Here we normalize the counts by library size and use the *rtracklayer* package to export the coverage per window in bigWig format.

```
gr <- rowRanges(windows)

CPM <-  csaw::calculateCPM(windows, log=FALSE)

for (cn in colnames(CPM)) {
  gr$score <- CPM[,cn]
  rtracklayer::export.bw(gr, sprintf('bw/%s.bigWig', cn),
                         format = "bigWig")
}
```



Comparison of BAM and bigWig coverage for DP replicate one at the Hprt locus, a housekeeping gene.

Quality control

- Quality control steps start already upstream of read alignment, e.g. using tools such as fastQC. Here we assume all necessary steps (e.g. removing adapter sequences or low quality samples) have been taken and we can focus on analysis specific quality control figures. Important quality controls include assessment of:
    - Library complexity
    - Read enrichment in open vs. closed chromatin
    - GC bias

Library complexity

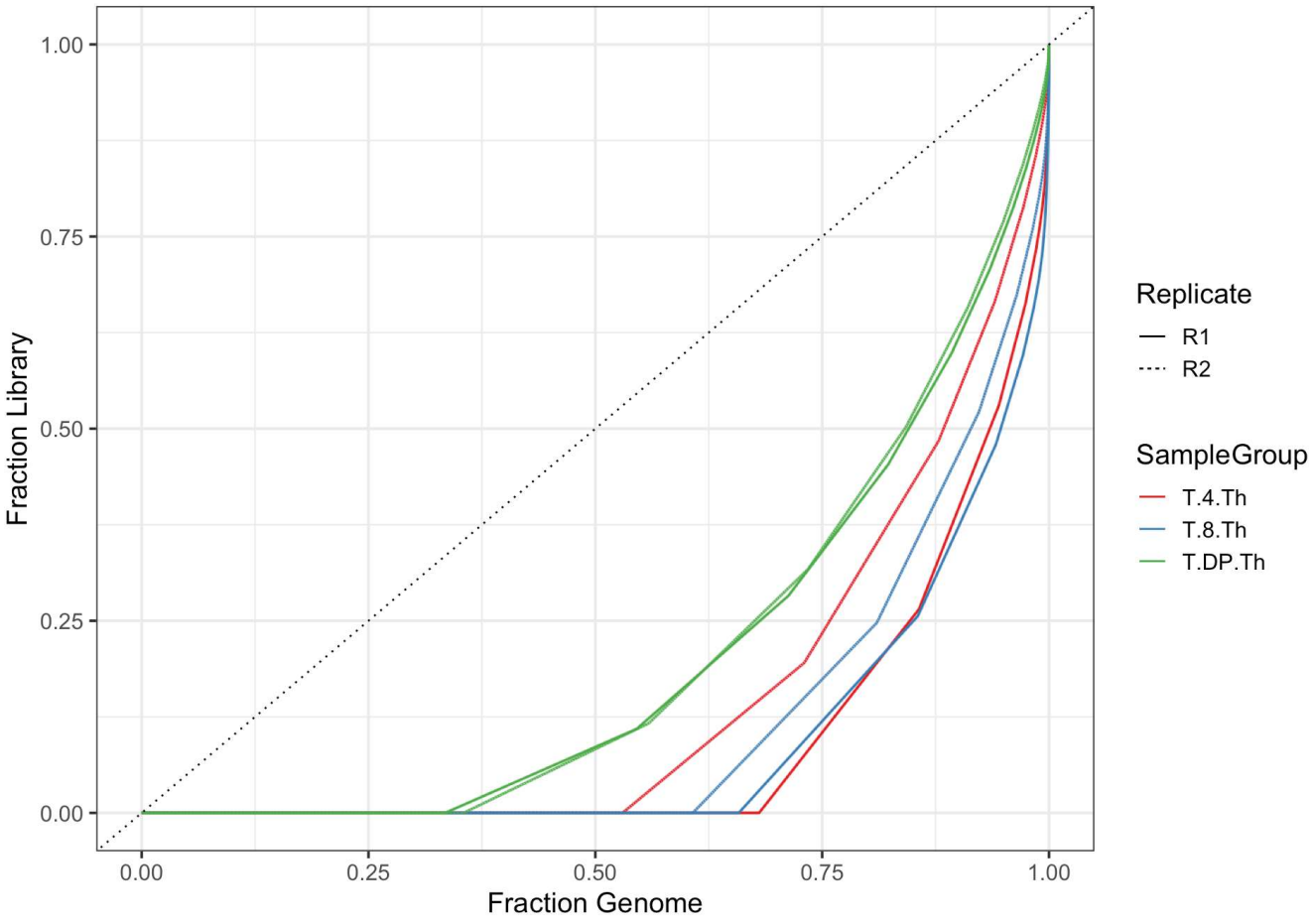
- When ATAC-Seq is performed from low amounts of input material, the coverage of the genome is limited and when combined with PCR amplification can lead to a low complexity of the final library despite high sequencing depth. This can be assessed by the PCR duplication level or by scanning through the BAM coverage in IGV. In the latter case, low complexity libraries appear as piles of read pairs aligning to the same position. These artificial coverage peaks must be removed (through PCR de-duplication) prior to peak calling. In our case, samples don't suffer from low complexity.

Read enrichment plot

- A common source of variance in ATAC-Seq data is due to differences in the signal-to-noise ratio between samples. In other words, the fraction of mapped reads spend in open versus closed chromatin differs between samples.
- The technical source of this variance is often due to differences in:
    - the level of dechromatinized DNA of damaged/dead cells (=background signal)
    - the ratio of Tn5 to nuclei
    - permeabilization of the nuclei
    - tagmentation efficiency
- Enrichment bias is the ATAC-Seq analog of IP-efficiency differences among ChIP-Seq samples. Here we use the more general term "enrichment bias" because of its multiple levels of origin.
- Genome-wide enrichment is assessed using our window counts by:
    - ordering windows by increasing count number, build the cumulative sum and divide by the total.
    - plot this fraction against the fraction of the covered genome.

```
fracLib <- matrix(0, nrow = nrow(windows), ncol = ncol(windows),
                  dimnames = dimnames(windows))
for (cn in colnames(windows)) {
  counts <- as.numeric(assays(windows)$counts[,cn])
  counts <- sort(counts, decreasing=FALSE)
  fracLib[,cn] <- cumsum(counts)/sum(counts)
}
fracGenome <- seq(0, 1, length.out=nrow(windows)) # since all windows are of the same size
```

```
data <- tibble(x=fracGenome) |>
  bind_cols(as_tibble(fracLib)) |>
  pivot_longer(-x) |>
  mutate(SampleGroup=str_replace(name, "_rep\\d+$", ""),
         Replicate=str_replace(name, ".*_rep(\\d+)$", "R\\1"))

ggplot(data = data,
       mapping = aes(x=x, y=value, colour=SampleGroup, linetype=Replicate)) +
  rasterize( geom_line() ) +
  geom_abline(slope=1, intercept=0, linetype="dotted", colour="black") +
  scale_color_brewer(palette = "Set1") +
  labs(x="Fraction Genome", y="Fraction Library") +
  theme_bw(base_size=15)
```



- Some notes on the enrichment figure:
  - the bisecting line corresponds to a uniform distribution of reads across windows i.e. when there is no enrichment.
  - enrichment strength is measured by the deviation from the bisecting line. A strong enrichment of reads in a small number of windows shows up as a sudden steep rise of the line.
  - the percentage of the uncovered (but mappable) genome corresponds to the point where the line starts to rise from 0 library fraction.
  - low sequencing depth or low library complexity results in an intial discreteness of the line (many windows with 1, 2, 3, etc. reads)

- Despite the overall low sequencing depth, the figure tells us that DP samples are less enriched compared to CD4 and CD8. This fact, together with a higher sequencing depth, leads to a higher coverage of the genome (about 65%), but a lower signal-to-noise ratio in DP.

Promoter accessibility plot

- To better understand the effect of enrichment bias, lets focus on gene promoters. Promoters of expressed genes are generally accessible (=open) and thus a site of read enrichment. By comparing open and closed promoters we can visualize the difference in foreground vs. background signal as seen above.

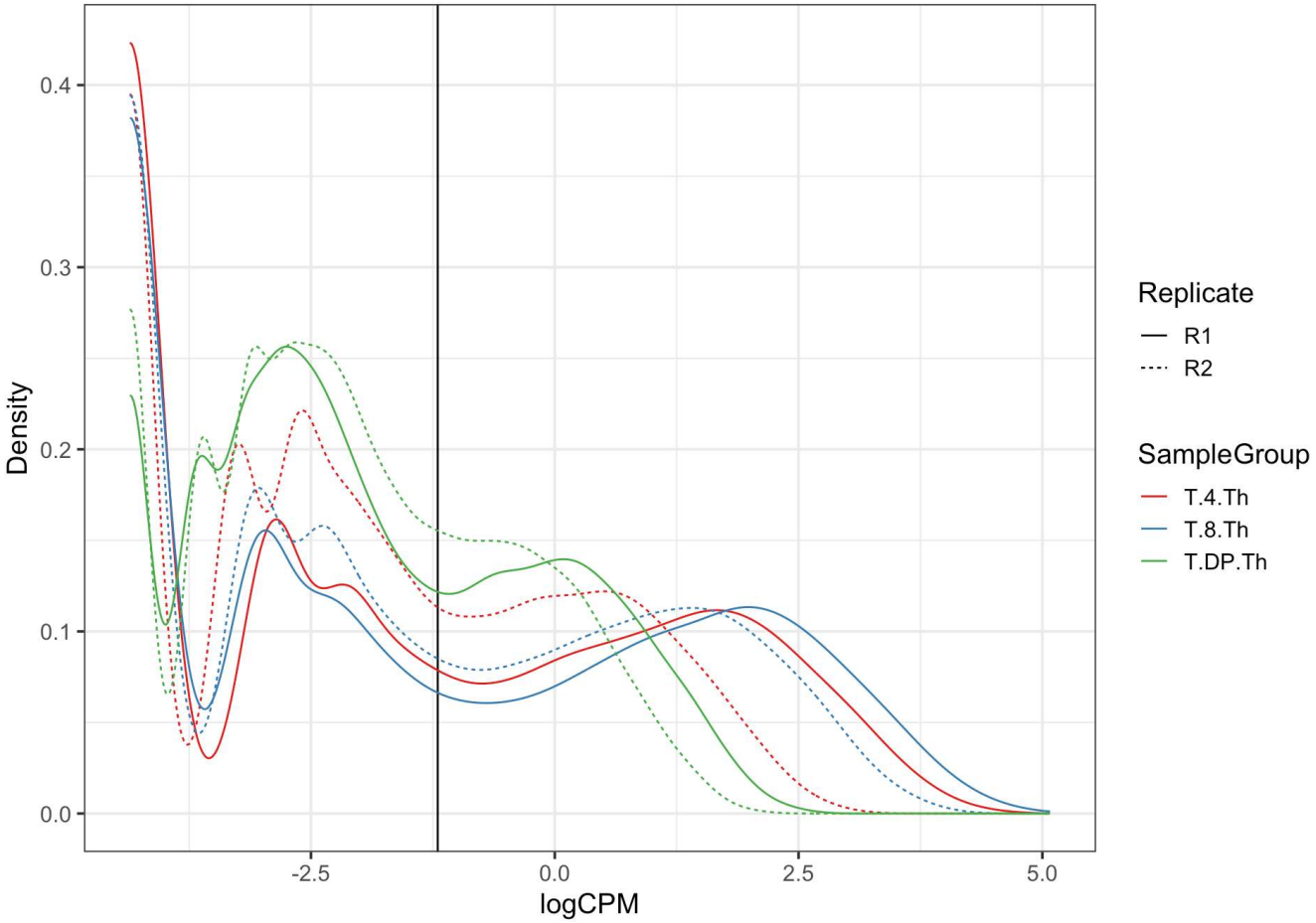
- Ideally we count the reads around transcription start sites (TSS) in use, which would require matching mRNA-Seq data. We will take a simpler approach and use windows overlapping with the most upstream TSS as defined in the "known gene" track from UCSC. Although we use the word "promoter" below, we mean TSS-overlapping windows in the following.

```
tss <- genes(TxDb.Mmusculus.UCSC.mm10.knownGene)
tss <- resize(tss, width=1, fix="start")
overlapTSS <- rowRanges(windows) %over% tss
table(overlapTSS)
```

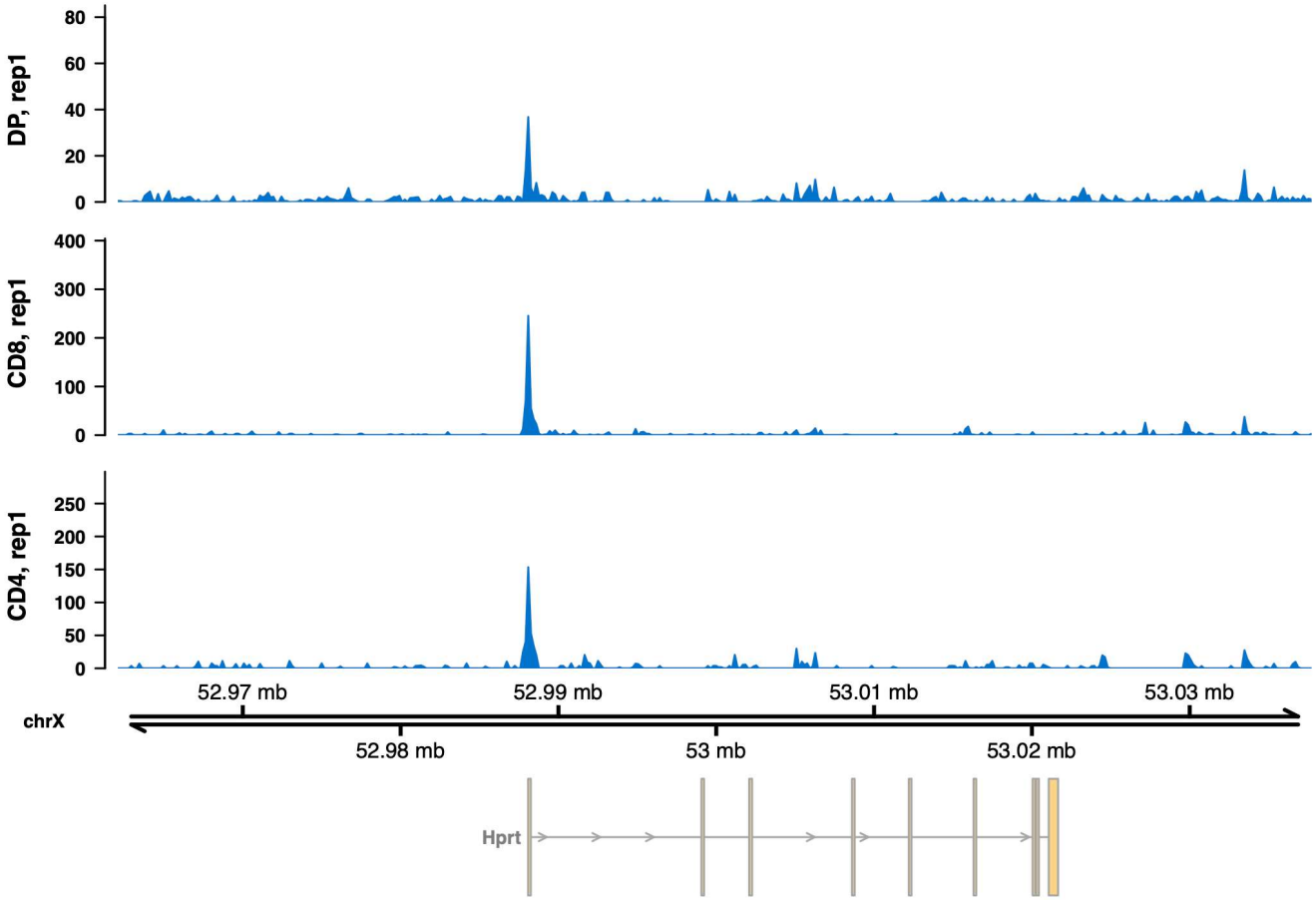
```
overlapTSS
   FALSE     TRUE
17626435    24010
```

```
rowData(windows)$overlapTSS <- overlapTSS
```

```
logCPM <- csaw::calculateCPM(windows, log=TRUE) |> # normalized by library size only
  as_tibble() |>
  filter(overlapTSS) |>
  pivot_longer(everything()) |>
  mutate(SampleGroup=str_replace(name, "_rep\\d+$", ""),
         Replicate=str_replace(name, ".*_rep(\\d+)$", "R\\1"))

ggplot(data=logCPM,
       aes(x=value, colour=SampleGroup, linetype=Replicate)) +
  geom_vline(xintercept=-1.2) + # threshold separating open and closed promoters
  labs(x="logCPM", y="Density") +
  scale_color_brewer(palette = "Set1") +
  stat_density(geom="line", position="identity") +
  theme_bw(base_size=15)
```



- The logCPM densities appear bimodal and can be split into average abundance for closed (left peaks) and open (right peaks) promoters using a threshold around logCPM = -1.2. The difference between average abundance of closed and open promoters can serve as a proxy for the signal-to-noise ratio of a sample. It is larger in CD4 and CD8 samples compared to DP samples.
- We can compare this finding with read coverage tracks of the data, either using IGV or the *Gviz* package (see code below).



Different dynamic ranges of the promotor signal at the Hprt gene (housekeeping gene).

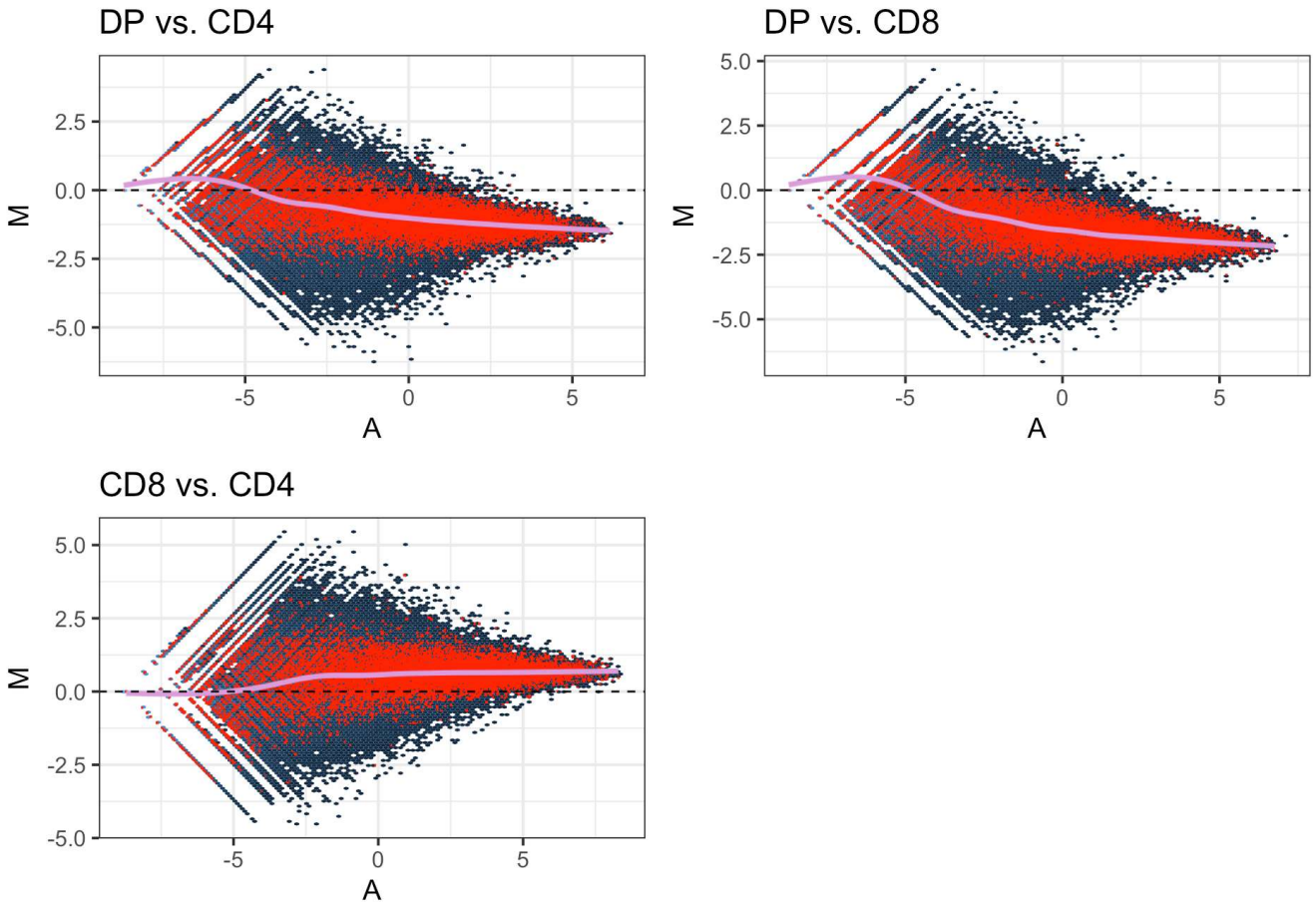
- Note the different y-scales on the figure! Each track is normalized to the number of mapped reads (excluding all regions mentioned above). DP samples show a smaller dynamic range in the coverage tracks due to lower enrichment. As we are looking at a housekeeping gene, we would expect similar accessibility levels across samples.

MA plot

- When comparing abundances across samples, the consequence of enrichment bias can be further highlighted using mean-average (MA) plots which you already encountered in the mRNA-Seq and ChIP-Seq lectures. Here we additionally highlight promoter windows in red.

```
logCPM <-  csaw::calculateCPM(windows, log=TRUE)

ct <- split(colnames(logCPM), colData(windows)$CellType)

df <- vapply(ct, function(x) {
  rowMeans(logCPM[,x,drop=FALSE])
}, rep(0, nrow(windows))) |>
  as_tibble() |>
  mutate(overlapTSS = rowData(windows)$overlapTSS,
         GC = rowData(windows)[,'G|C'])

plotMA <- function(df, s1, s2) {
  ggplot(df,
         mapping = aes(x = .data[[s1]] + .data[[s2]],
                       y = .data[[s1]] - .data[[s2]])) +
```

```
    geom_hex(bins = 100, aes(fill=after_stat(density)^(1/16)), show.legend = FALSE) +
    rasterize( geom_point(data = subset(df, overlapTSS), col='red', pch='.') ) +
    rasterize( geom_smooth(data = subset(df, overlapTSS), col='plum', se = FALSE) ) +
    geom_hline(yintercept = 0, lty = 2) +
    labs(x = "A", y="M", title=sprintf("%s vs. %s", s1, s2)) +
    theme_bw(base_size = 15)
}

combi <- list(c('DP','CD4'), c('DP','CD8'), c('CD8','CD4'))

gl <- lapply(combi, function(x) plotMA(df, x[1], x[2]))

patchwork::wrap_plots(gl, ncol=2)
```



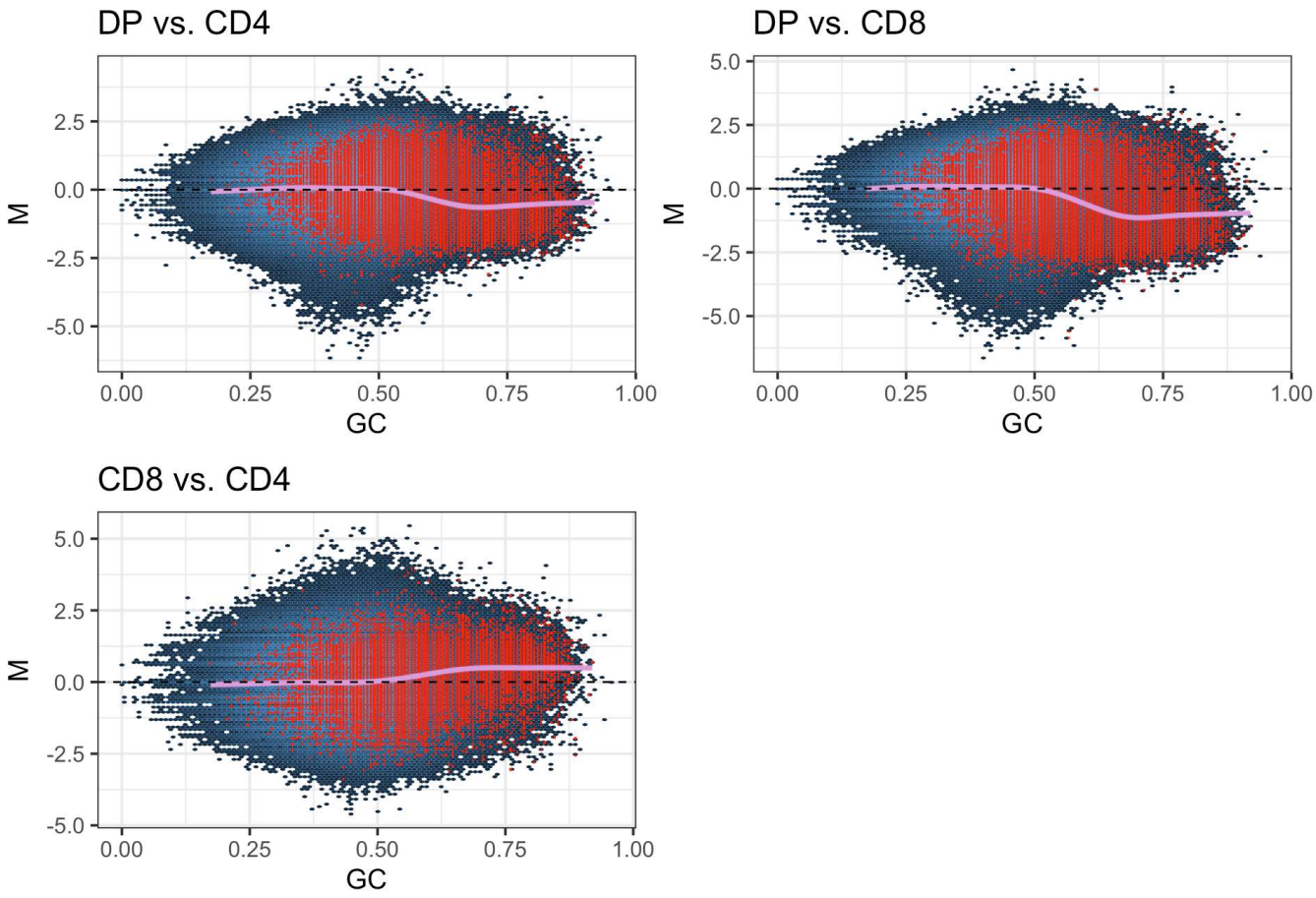
- The comparisons involving DP show a clear downward trend. Thus, open promoters appear systematically “more accessible” in CD4 or CD8 versus DP. We observe an opposite trend in CD8 versus CD4. Thus enrichment bias can be characterized by a systematic trend depending on the average abundance.

GC bias plot

- The GC content of a sequence influences the amplification rate during PCR. Subtle differences in GC composition between samples can get amplified during library preparation leading e.g. to an enrichment of GC-rich sequences and a suppression of AT-rich sequences.

- Similar to the MA-plot above we can highlight any systematic dependence of the log-fold-change (M values) on GC content in the figures below.

```
plotGC <- function(df, s1, s2) {
  ggplot(df,
         mapping = aes(x = GC,
                       y = .data[[s1]] - .data[[s2]])) +
    geom_hex(bins = 100, aes(fill=after_stat(density)^(1/16)), show.legend = FALSE) +
    rasterize( geom_point(data = subset(df, overlapTSS), col='red', pch='.') ) +
    rasterize( geom_smooth(data = subset(df, overlapTSS), col='plum', se = FALSE) ) +
    geom_hline(yintercept = 0, lty = 2) +
    labs(x = "GC", y="M", title=sprintf("%s vs. %s", s1, s2)) +
    theme_bw(base_size = 15)
}

gl <- lapply(combi, function(x) plotGC(df, x[1], x[2]))

patchwork::wrap_plots(gl, ncol=2)
```



- We observe similar trends as in the MA figures above. Since the majority of highly accessible promoters are also GC rich, the question arises if the the GC bias is due to enrichment bias?

Assessing GC bias in the presence of enrichment bias

ATAC-Seq libraries focus on promoter and other accessible regulatory regions which in mammalian genomes are often associated with higher GC (and CpG) content. Therefore we need to assess GC bias in the presence of an enrichment bias. One possibility is to split promoters into GC-low and GC-high, and check if their enrichment bias differs. Below we remove the enrichment bias by normalization and check if GC bias is also corrected. Alternatively, more elaborate normalization techniques such as *cqn* or *qsmooth* might be applied.

Filter windows (“peak calling”)

- We now aim at identifying windows with increased abundance compared with a global background. We will do so jointly across all samples as a means to remove uninteresting, lowly-covered windows. The main aim is to improve the detection of differentially accessible windows in the later analysis.
- Here we use a cutoff based on the average global abundance across windows and samples. A global background signal is estimated by counting reads in 10 kb windows. Large bins ensure reliable estimates of read density even if the genome coverage is low. Note, we use the same `param` object to synchronize both calls.

```
background <- windowCounts(bamFiles, bin=TRUE, width=10000L, param=param)
colData(background) <- cbind(colData(background), alignments)
saveRDS(background, file="data/ATACSeq/csaw_background_10k_rmDup.rds")
```

- Next we apply `filterWindowsGlobal` to compute enrichment values which measure the increase in the read count in each window over the expected count based on the background signal. The resulting filter statistic for each window is defined as the difference between the average window abundance (~ average log2CPM) and the median global background abundance. It is adjusted for the differences in widths between windows and bins.

```
filterStat <- filterWindowsGlobal(windows, background)
str(filterStat)
```

```
List of 3
 $ abundances     : num [1:17650445] -3.25 -2.93 -2.93 -2.84 -3.13 ...
 $ back.abundances: num [1:17650445] -2.79 -2.79 -2.79 -2.79 -2.79 ...
 $ filter         : num [1:17650445] -0.4581 -0.1439 -0.1395 -0.0541 -0.3466 ...
```

```
globalBG <- median( filterStat$back.abundances )
globalBG
```

```
[1] -2.78763
```

```
head(filterStat$abundances - globalBG)
```

```
[1] -0.45806515 -0.14385829 -0.13949719 -0.05407898 -0.34656929 -0.45806515
```

- To filter for high abundance windows we require at least 3-fold increase in enrichment over the background abundance. Note the filter statistics is on log2-scale. Our required 3-fold increase results in a similar threshold used above to split promoter abundances into open and closed.

```
keep <- filterStat$filter > log2(3)
sum(keep)
```

```
[1] 125837
```

```
mean(keep)
```

```
[1] 0.007129395
```

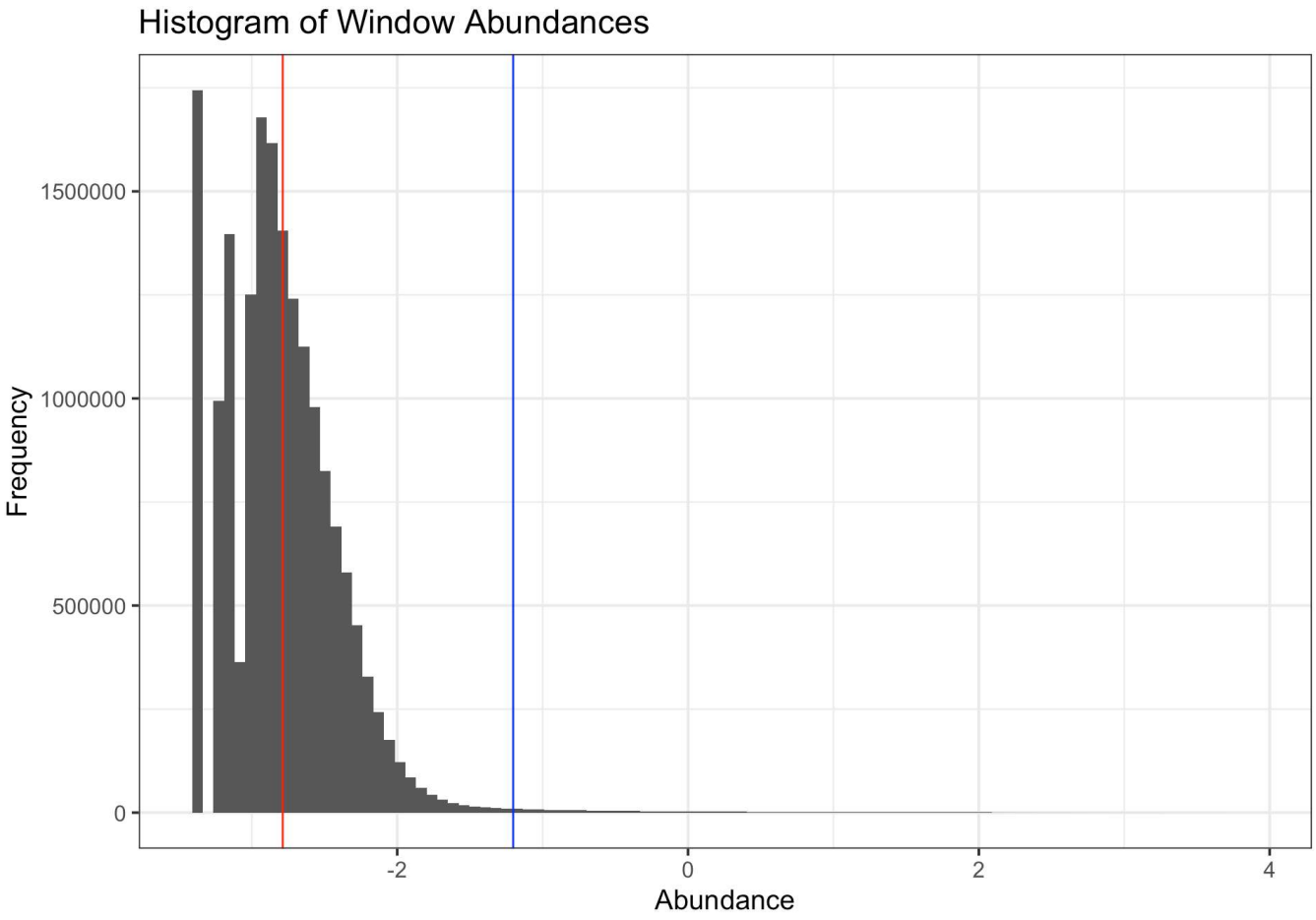
```
min(filterStat$abundances[keep])
```

```
[1] -1.202668
```

```
globalBG + log2(3)
```

```
[1] -1.202668
```

```
ggplot(data = tibble(x=filterStat$abundances),
       mapping = aes(x)) +
    geom_histogram(bins = 100) +
    geom_vline(xintercept = globalBG, color='red') +
    geom_vline(xintercept = globalBG + log2(3), color='blue') +
    labs(x="Abundance", y="Frequency", title="Histogram of Window Abundances") +
    theme_bw(base_size=15)
```



- We subset the `SummarizedExperiment` and export the filtered windows to BED format using the export functionality of *rtracklayer*:

```
rowData(windows)$enrichment <- filterStat$filter

filtered.windows <- windows[keep,]
```

```
rtracklayer::export(rowRanges(filtered.windows),
                    'data/ATACSeq/filtered_global.bed')
```

Identifying peaks relative to local background

We did not use the local filtering option of csaw (see csawBook: filtering) because ATAC-Seq peaks are broader and have more internal structure compared to e.g. ChIP-Seq of H3K4me3 or H3K27ac making the use of local backgrounds more error prone.

Sample group specific filtering/peak calling

Internally, the `filterWindowsGlobal` function will compare the sample-average abundance of a window with the average global background abundance. This average might be low if a peak appears only in a small subset of the samples. For large datasets with many sample groups, a group specific filtering with subsequent merging of the retained windows is required in order to capture group-specific peaks.

Differential accessibility analysis

- In analogy to differential expression analysis, differential accessibilty analysis will evaluate accessibility of windows across sample groups.

Normalization

- Up to now we normalized the data by library size only. As highlighted in the MA plots above, comparison between any cell type shows a trend of the log-fold-change on the average abundance.

Normalization by effective library size

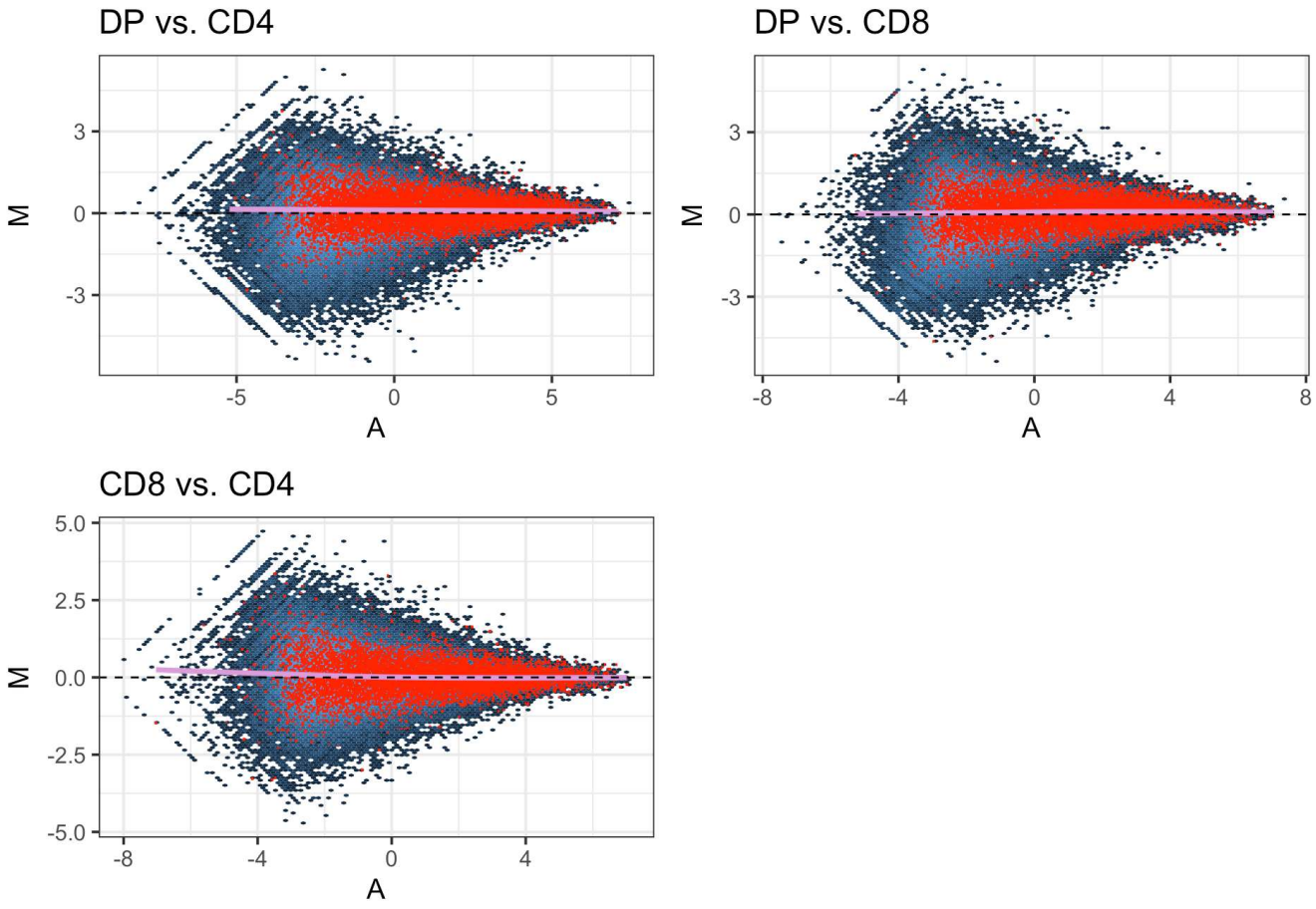
Normalization procedures known from mRNA-Seq differential expression analysis, such as TMM (`edgeR`) or median of ratio's (`DESeq2`), use an effective library size to compensate the effect of composition biases i.e. when few genes occupy a significant fraction of the library. Looking at MA-plots, the effect of scaling the library size leads to a global shift of the M values (i.e. the whole cloud of points moves up or down), allowing to either set the M values of low abundance/closed or the high abundance/open windows to zero on average. In other words, scaling of library sizes cannot normalize background and foreground windows independently.

- Offsets allow to normalize this type of trended biases across samples. They are window- and sample-specific normalization factors, similar to the (effective) library size which are just sample-specific normalization factors.
- For each sample, a trend curve is fitted to the log-counts against the log-average counts across all samples. The scaled, fitted value for each window and sample is then used as an offset in the generalized linear model underlying the `edgeR` fitting procedure to correct for the trend. For more details, have a look at csawBook: trended biases.

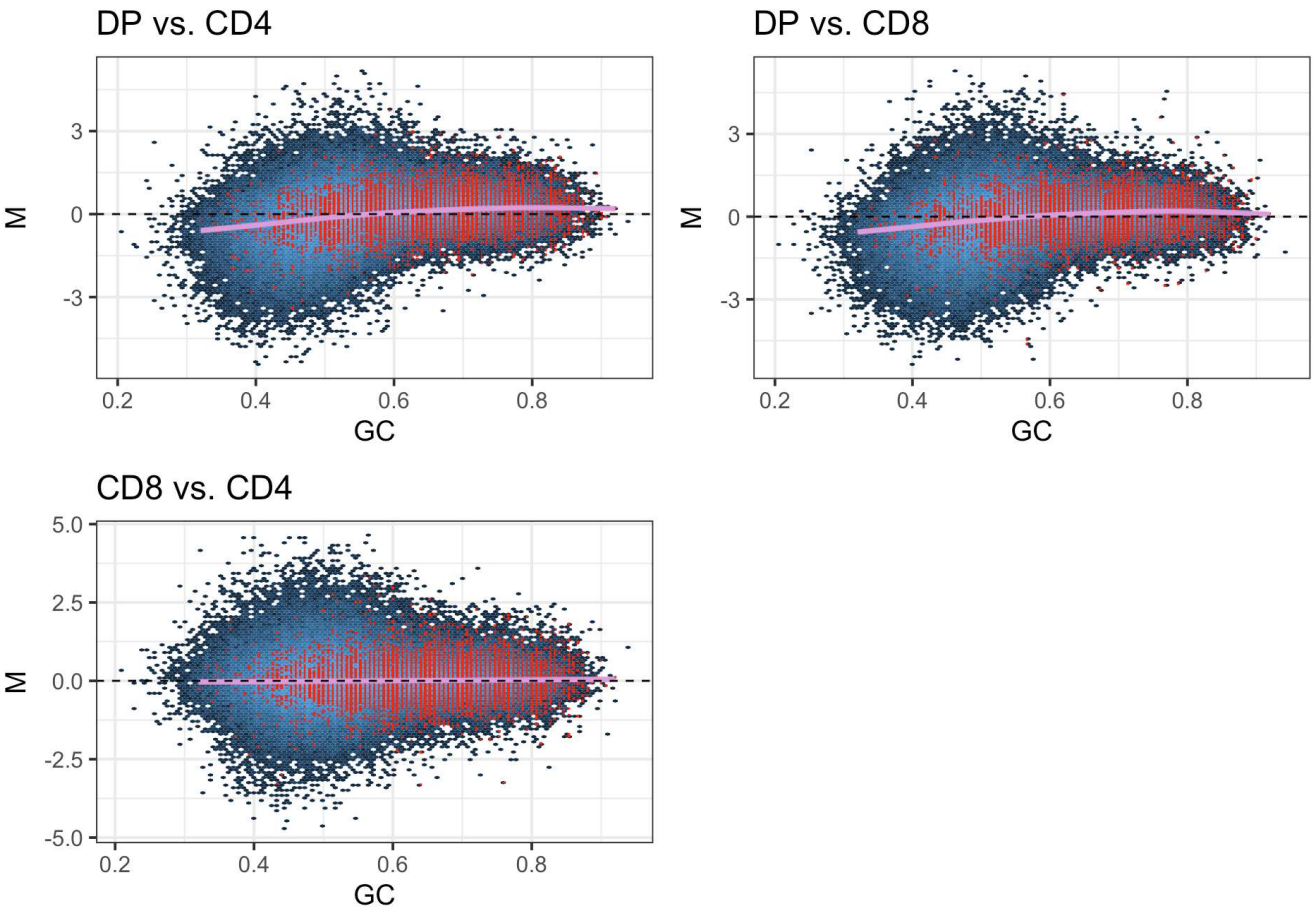
```
filtered.windows <- normOffsets(filtered.windows)
```

```
logCPM <- csaw::calculateCPM(filtered.windows, log=TRUE, use.offsets = TRUE)
```

```
ct <- split(colnames(logCPM), colData(filtered.windows)$CellType)

df <- vapply(ct, function(x) {
  rowMeans(logCPM[,x,drop=FALSE])
}, rep(0, nrow(filtered.windows))) |>
  as_tibble() |>
  mutate(overlapTSS = rowData(filtered.windows)$overlapTSS,
         GC = rowData(filtered.windows)[,'G|C'])

patchwork::wrap_plots(lapply(combi, function(x) plotMA(df, x[1], x[2])), ncol=2)
```




```
patchwork::wrap_plots(lapply(combi, function(x) plotGC(df, x[1], x[2])), ncol=2)
```



- As expected, any systematic trend between mean abundance and differential accessibility is now removed (MA figures). The correction also removed the GC bias trend for the most part highlighting the link between both.

Differential accessibility test

- We are now ready to identify regions of differential accessibility between cell types. We could use *DESeq2* package as you already did for mRNA-Seq analysis or we stay in the framework of *csaw* which is using the *edgeR* package for differential accessibility testing. The procedure consists of following steps:

1. Convert the normalized `SummarizedExperiment` to a `edgeR::DGEList` object. It is a simple extension of a list object with slots for:
   - `counts`: count matrix
   - `genes`: gene info or row data of `SummarizedExperiment`
   - `samples`: sample info; column data of `SummarizedExperiment`
   - `offset`: if given, stores the offsets used in the negative-binomial model

```
dgel <- csaw::asDGEList(filtered.windows,
                        genes = as.data.frame(rowData(filtered.windows)),
                        group = colData(filtered.windows)$CellType)
colnames(dgel) <- colnames(filtered.windows)
rownames(dgel) <- as.character(rowRanges(filtered.windows))
dgel
```

```
An object of class "DGEList"
$counts
                     T.DP.Th_rep1 T.DP.Th_rep2 T.4.Th_rep1 T.4.Th_rep2
chr1:3042901-3043050           12            8           3          13
chr1:3043051-3043200           12           19           5          16
chr1:3043201-3043350           28           30           9          23
chr1:4258201-4258350           20           18           2          10
chr1:4258351-4258500           20           20           4          11
                     T.8.Th_rep1 T.8.Th_rep2
chr1:3042901-3043050           5           9
chr1:3043051-3043200           5          12
chr1:3043201-3043350          10          20
chr1:4258201-4258350           4           4
chr1:4258351-4258500           5           7
125832 more rows ...

$samples
             group lib.size norm.factors
T.DP.Th_rep1    DP 34072462            1
T.DP.Th_rep2    DP 30895103            1
T.4.Th_rep1    CD4 11731713            1
T.4.Th_rep2    CD4 18099631            1
T.8.Th_rep1    CD8 13601670            1
T.8.Th_rep2    CD8 14474145            1

$genes
                          G.C N overlapTSS enrichment
chr1:3042901-3043050 0.4666667 0      FALSE   1.830482
chr1:3043051-3043200 0.4733333 0      FALSE   2.210324
chr1:3043201-3043350 0.3866667 0      FALSE   2.906892
chr1:4258201-4258350 0.6333333 0      FALSE   1.920591
chr1:4258351-4258500 0.6000000 0      FALSE   2.117133
125832 more rows ...

$offset
     T.DP.Th_rep1 T.DP.Th_rep2 T.4.Th_rep1 T.4.Th_rep2 T.8.Th_rep1
[1,]     16.96602     16.68032    16.64400    16.48163    17.03801
[2,]     16.91185     16.59110    16.68977    16.46407    17.10377
[3,]     16.83705     16.45899    16.76094    16.44986    17.19582
[4,]     16.95023     16.65472    16.65806    16.47464    17.05834
[5,]     16.92318     16.60906    16.67969    16.46706    17.09121
     T.8.Th_rep2
[1,]    16.68291
[2,]    16.73234
[3,]    16.79023
[4,]    16.69691
[5,]    16.72270
125832 more rows ...
```

2. Define a model matrix which assigns samples to groups of interest.

```
moma <- model.matrix(~ 0 + group, dgel$samples)
moma
```

```
             groupCD4 groupCD8 groupDP
T.DP.Th_rep1        0        0       1
T.DP.Th_rep2        0        0       1
T.4.Th_rep1         1        0       0
T.4.Th_rep2         1        0       0
T.8.Th_rep1         0        1       0
T.8.Th_rep2         0        1       0
attr(,"assign")
[1] 1 1 1
attr(,"contrasts")
attr(,"contrasts")$group
[1] "contr.treatment"
```

> **Dispersion estimation in edgeR**
>
> Older versions of edgeR (< 4.0) required a separate dispersion estimation step. This in now intregrated in the model fitting procedure in recent versions of edgeR 4.0.

3. Run the test (`glmQLFit`, `glmQLFtest`) and correct for multiple testing (`decideTests`, `topTags`). Here we focus on the CD4 vs DP comparison:

```
fit <- glmQLFit(dgel, moma)

contr <- makeContrasts(groupCD4 - groupDP, levels = colnames(moma))

res <- glmQLFTest(fit, contrast = contr)

table(decideTests(res, p.value = 0.05, lfc = 1.5))
```

```
   -1      0      1
 2621 117065   6151
```

```
tt <- topTags(res, n=Inf, sort.by="none")$table |>
    as_tibble() |>
    dplyr::select(logFC,logCPM,F,PValue,FDR) |>
    mutate(Windows = rownames(dgel), .before=1) |>
    arrange(FDR) |>
    head()
```

- After testing for differential accessibility we store the results in the row data of the `filtered.windows` object:

```
rowData(filtered.windows) <- cbind(rowData(filtered.windows), tt)

saveRDS(filtered.windows, file='data/ATACSeq/csaw_filtered_window_counts_fitted.rds')
```

Join neighboring windows and compute their significance

- The differential analysis results are still on the window level and are not taking into account if differential windows form larger peak regions. The aggregation into larger regions is done in two steps:

1. by merging neighboring windows

2. summarizing the differential results for those merged windows

- Neighboring windows are joined into larger regions by calling `mergeWindows`. It will apply a single linkage algorithm to join neighboring windows with less than `tol` distance apart. An optional `max.width` parameter would keep regions at a maximal width in order to avoid very large regions.

```
merged <- mergeWindows(rowRanges(filtered.windows), tol=150L)
str(merged,1)
```

```
List of 2
 $ ids    : int [1:125837] 1 1 1 2 2 3 3 3 3 4 ...
 $ regions:Formal class 'GRanges' [package "GenomicRanges"] with 7 slots
```

```
merged$region
```

```
GRanges object with 48744 ranges and 0 metadata columns:
          seqnames          ranges strand
             <Rle>       <IRanges>  <Rle>
      [1]     chr1 3042901-3043350      *
      [2]     chr1 4258201-4258500      *
      [3]     chr1 4496251-4496850      *
      [4]     chr1 4771051-4771200      *
      [5]     chr1 4780201-4780350      *
      ...      ...             ...    ...
  [48740]     chrY   897451-897600      *
  [48741]     chrY 1010251-1010700      *
  [48742]     chrY 1245451-1245900      *
  [48743]     chrY 2224951-2225100      *
  [48744]     chrY 2225401-2225550      *
  -------
  seqinfo: 22 sequences from an unspecified genome
```

```
summary(width(merged$region))
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  150.0   150.0   300.0   399.9   450.0  3900.0
```

- We annotate the peak regions using the `detailRanges` function of `csaw` to extract information about promoters and genes from annotation packages. Peak regions will be assign to features in the neighborhood of 10kb, promoters are defined as 1.5kb upstream and 500 bases downstream of a TSS.

```
anno <- detailRanges(merged$region,
                     txdb=TxDb.Mmusculus.UCSC.mm10.knownGene,
                     orgdb=org.Mm.eg.db, promoter=c(1500, 500), dist=10*1e3L)

merged$region$overlap <- anno$overlap
merged$region$left <- anno$left
merged$region$right <- anno$right
merged$region
```

```
GRanges object with 48744 ranges and 3 metadata columns:
          seqnames          ranges strand |      overlap          left
             <Rle>       <IRanges>  <Rle> |  <character>   <character>
      [1]     chr1 3042901-3043350      * |
      [2]     chr1 4258201-4258500      * |      Rp1:-:I
      [3]     chr1 4496251-4496850      * |   Sox17:-:PE   Sox17:-:309
      [4]     chr1 4771051-4771200      * |
      [5]     chr1 4780201-4780350      * |   Mrpl15:-:I Mrpl15:-:2553
      ...      ...             ...    ... .          ...           ...
  [48740]     chrY   897451-897600      * |    Kdm5d:+:P
  [48741]     chrY 1010251-1010700      * | Eif2s3y:+:PE
  [48742]     chrY 1245451-1245900      * |     Uty:-:PE     Uty:-:232
  [48743]     chrY 2224951-2225100      * |
  [48744]     chrY 2225401-2225550      * |
                 right
            <character>
      [1]
      [2]    Rp1:-:3027
      [3]
      [4] Mrpl15:-:2006
      [5]  Mrpl15:-:871
      ...           ...
  [48740]   Kdm5d:+:188
  [48741] Eif2s3y:+:560
  [48742]
  [48743]
  [48744]
  -------
  seqinfo: 22 sequences from an unspecified genome
```

- P-values are computed for the merged regions from the individual p-values of the underlying windows. Here its important to control the desired FDR at the region level instead of the window level. At the same time we would like to keep some resolution of the windowing approach i.e. indicate if merged neighboring windows show the same or opposite changes in accessibility. The `combineTests` function will summarize the test statistics on the region level, for details check csawBook: correction for multiple testing:

```
region.stats <- combineTests(merged$id, tt) |>
    as_tibble() |>
    mutate(rep.logCPM = tt[rep.test,'logCPM'],
           Contrast = colnames(contr))

region.stats |>
    head()
```

```
# A tibble: 6 × 10
  num.tests num.up.logFC num.down.logFC   PValue     FDR direction rep.test
      <int>        <int>          <int>    <dbl>   <dbl> <chr>        <int>
1         3            0              0  1.85e-1 4.00e-1 down             3
2         2            0              2  3.42e-2 1.38e-1 down             5
3         4            1              0  5.65e-6 2.16e-4 up               7
4         1            0              0  2.86e-1 5.15e-1 down            10
5         1            0              0  3.99e-1 6.19e-1 up              11
```

```
6         4            0              0  6.07e-1 7.78e-1 up              13
# ℹ 3 more variables: rep.logFC <dbl>, rep.logCPM <dbl>, Contrast <chr>
```

- Here every row is a merged peak region with the summary statistics based on the underlying window tests.
- We merge the region statistics with the region annotation and save the final results as a table (gzipped plain text file) and also export it to BED format:

```
peaks <- merged$region

mcols(peaks) <- c(mcols(peaks), region.stats)

peaks$name <- ifelse(peaks$rep.logFC > 1.5 & peaks$FDR < 0.05, 'CD4',
                     ifelse(peaks$rep.logFC < (-1.5) & peaks$FDR < 0.05, 'DP', 'Common'))

write.table(as.data.frame(peaks), file=gzfile('data/ATACSeq/peaks_CD4_vs_DP.tsv.gz'),
            row.names=FALSE, sep='\t')
rtracklayer::export(peaks, 'data/ATACSeq/peaks_CD4_vs_DP.bed')
```

Inspecting the results

- How many common and cell type specific peaks are there and how many are of type “mixed”?

```
table(peaks$name, peaks$direction)
```

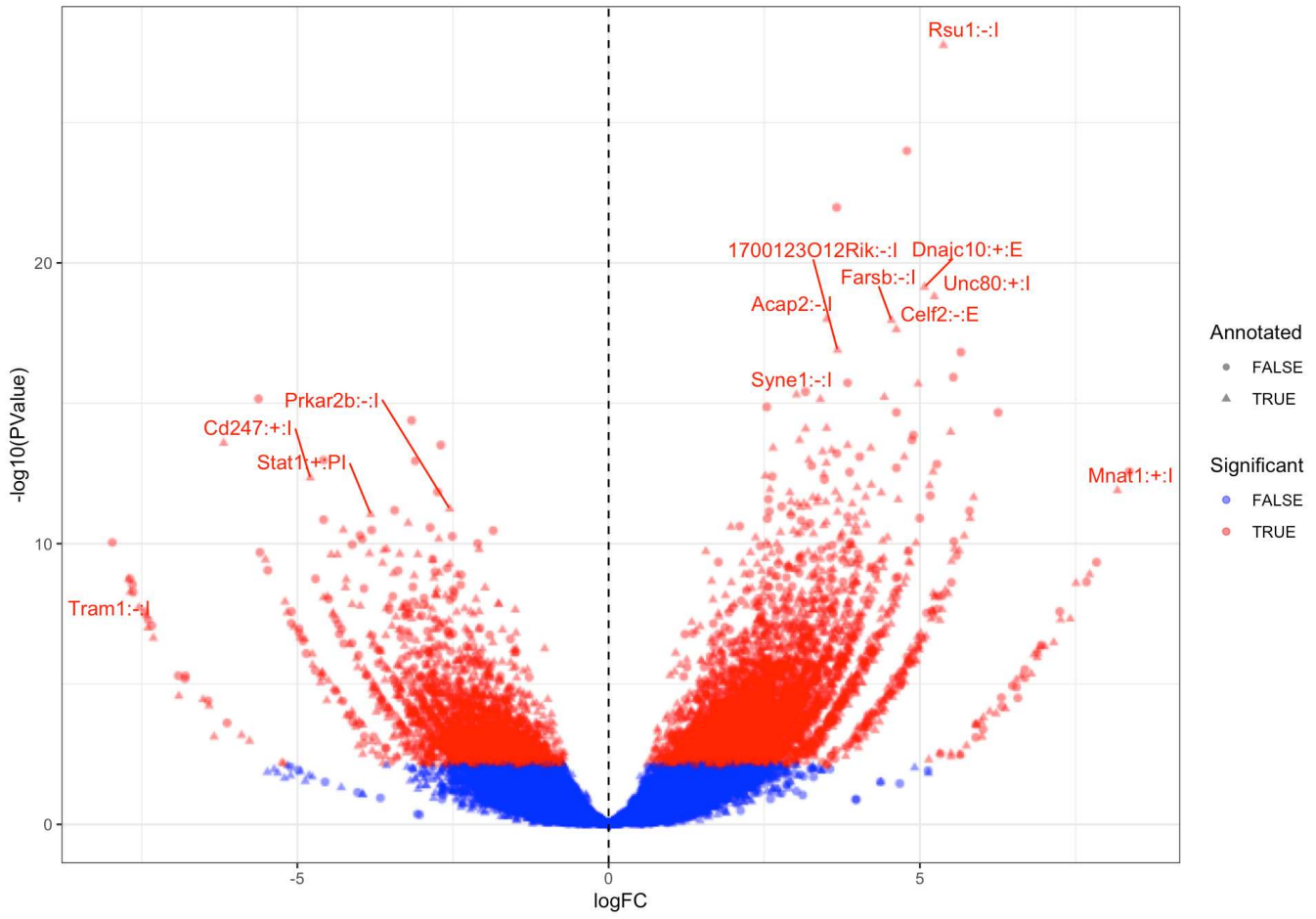
```
         down mixed  none    up
 CD4        0     2     0  4524
 Common 17393  4710     1 20118
 DP      1994     2     0     0
```

- Many mixed, cell type specific peaks would indicate a too coarse resolution of the regions. Our resolution looks good.
- Plot the merged test results as a Volcano plot:

```
data <- mcols(peaks) |>
    as_tibble() |>
    mutate(logFC = rep.logFC,
           Significant = FDR < 0.05,
           Annotated = overlap != "")

ggplot(data = data,
       mapping = aes(x = logFC,
                     y = -log10(PValue),
                     color = Significant,
                     shape = Annotated,
                     label = overlap)) +
    rasterize( geom_point(alpha=0.5) ) +
    geom_text_repel(data = data |> filter(Annotated & Significant & abs(logFC) > 1.5),
                    show.legend = FALSE) +
    geom_vline(xintercept = 0, lty = 2) +
```

```
    scale_color_manual(values=c(`TRUE`='red', `FALSE`='blue')) +
    theme_bw()
```

```
Warning: ggrepel: 4210 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```



- Check a few regulated sites in IGV and have a look at positive controls such as Runx3 and Cd8 for CD8, Gata3 and Zbtb7b (Thpok) for CD4, Ccr7 (for exit of CD4/CD8 from thymus) etc.

Outlook

- The results of ATAC-Seq analysis usually don't stand alone. Downstream of differential peak calling and annotation is often some type of overlap and/or filtering by additional data e.g. mRNA-Seq or ChIP-Seq. Further annotation of peaks by transcription factor binding sites followed by some type of enrichment test is often used to identify regulators driving changes in accessibility and gene expression. This will be a focus of the following lecture!

Appendix: plotting coverage tracks with Gviz

- Using the *Gviz* package we can plot specific regions to check the ATAC-Seq profile.
- Assuming you have access to the BAM files, the following code will produce the coverage tracks from the example above (housekeeping gene Hprt):

```
library(Gviz)

chrom <- "chrX"
afrom <- 52977084 - 15000L
ato <- 53002718 + 35000L

T.DP.Th_rep1 <- AlignmentsTrack(bamFiles["T.DP.Th_rep1"], type="coverage",
                           col="dodgerblue3", fill="dodgerblue3",
                           name="DP, rep1",
                           transformation=function(x) x/colData(windows)["T.DP.Th_rep1","total

T.4.Th_rep1 <- AlignmentsTrack(bamFiles["T.4.Th_rep1"], type="coverage",
                          col="dodgerblue3", fill="dodgerblue3",
                          name="CD4, rep1",
                          transformation=function(x) x/colData(windows)["T.4.Th_rep1","totals"]

T.8.Th_rep1 <- AlignmentsTrack(bamFiles["T.8.Th_rep1"], type="coverage",
                          col="dodgerblue3", fill="dodgerblue3",
                          name="CD8, rep1",
                          transformation=function(x) x/colData(windows)["T.8.Th_rep1","totals"]

txTrack <- UcscTrack(genome = "mm10", chromosome = chrom, track="NCBI RefSeq",
                     table = "ncbiRefSeqCurated", from = afrom, to =ato,
                     trackType = "GeneRegionTrack",
                     rstarts = "exonStarts", rends = "exonEnds",
                     gene = "name2", symbol = "name2",
                     transcript = "name", strand = "strand",
                     fontcolor = "black", name = "",
                     transcriptAnnotation = "symbol", showId=TRUE)

axisTrack <- GenomeAxisTrack(name=chrom, rotation.title=0, col="black", fontcolor="black")

plotTracks(list(T.DP.Th_rep1, T.8.Th_rep1, T.4.Th_rep1, axisTrack, txTrack),  chromosome=chrom
           fontcolor.title="black", background.title="transparent", showTitle=TRUE, col.axis="
           sizes=c(1.5,1.5,1.5,0.5,1))
```

- For more details on Gviz have a look at the user guide.

Exercise

1. Repeat the differential accessibility analysis contrasting CD8 with DP and save the region-level result in a table.

```
contr <- makeContrasts(groupCD8 - groupDP, levels = colnames(moma))
res <- glmQLFTest(fit, contrast = contr)

tt <- topTags(res, n=Inf, sort.by="none")$table[,c('logFC','logCPM','F','PValue','FDR')]

region.stats <- combineTests(merged$id, tt) |>
    as_tibble() |>
    mutate(rep.logCPM = tt[rep.test,'logCPM'],
```

```
            Contrast = colnames(contr))

peaks <- merged$region
mcols(peaks) <- c(mcols(peaks), region.stats)
peaks$name <- ifelse(peaks$rep.logFC > 1.5 & peaks$FDR < 0.05, 'CD8',
                     ifelse(peaks$rep.logFC < (-1.5) & peaks$FDR < 0.05, 'DP', 'Common'))

write.table(as.data.frame(peaks), file=gzfile('data/ATACSeq/peaks_CD8_vs_DP.tsv.gz'),
            row.names=FALSE, sep='\t')
```

2. Use the promotor counts provided on the course website and repeat the differential analysis. How many differentially accessible promoters (FDR < 0.05 and |logFC| > 1.5) do you get for each of the three comparison CD4 vs DP, CD8 vs DP and CD4 vs CD8?

```
proms <- readRDS('data/ATACSeq/csaw_promoter_counts_rmDup.rds')

proms <- csaw::normOffsets(proms)

dgel <- csaw::asDGEList(proms,
                        genes = as.data.frame(rowData(proms)),
                        group = colData(proms)$CellType)
colnames(dgel) <- colnames(proms)
rownames(dgel) <- as.character(rowRanges(proms)$gene_id)
dgel
```

```
An object of class "DGEList"
$counts
          T.DP.Th_rep1 T.DP.Th_rep2 T.4.Th_rep1 T.4.Th_rep2 T.8.Th_rep1
100009600           58           67          54           34           68
100009609           20           20           8            7            2
100009614           40           41           2           23           23
100009664           46           28           8           12            5
100012              17           29           6            8            8
          T.8.Th_rep2
100009600          50
100009609           8
100009614          11
100009664          16
100012             12
23657 more rows ...

$samples
             group lib.size norm.factors
T.DP.Th_rep1    DP 34072462            1
T.DP.Th_rep2    DP 30895103            1
T.4.Th_rep1    CD4 11731713            1
T.4.Th_rep2    CD4 18099631            1
T.8.Th_rep1    CD8 13601670            1
T.8.Th_rep2    CD8 14474145            1

$genes
            gene_id        symbol       G.C
100009600 100009600         Zglp1 0.5400000
100009609 100009609       Vmn2r65 0.4090909
```

```
100009614 100009614    Krtap12-22 0.4868182
100009664 100009664 F630206G17Rik 0.4613636
100012       100012          Oog3 0.4268182
23657 more rows ...

$offset
          T.DP.Th_rep1 T.DP.Th_rep2 T.4.Th_rep1 T.4.Th_rep2 T.8.Th_rep1
100009600     17.06941     16.85727    16.56514    16.49751    16.89501
100009609     17.37378     17.41372    16.16485    16.85313    16.17944
100009614     17.48577     17.37434    16.21388    16.74257    16.26985
100009664     17.48542     17.41407    16.18688    16.78727    16.20258
100012        17.42502     17.41981    16.17216    16.83042    16.17511
          T.8.Th_rep2
100009600    16.60856
100009609    16.50797
100009614    16.40648
100009664    16.41667
100012       16.47037
23657 more rows ...
```

```
moma <- model.matrix(~ 0 + group, dgel$samples)
moma
```

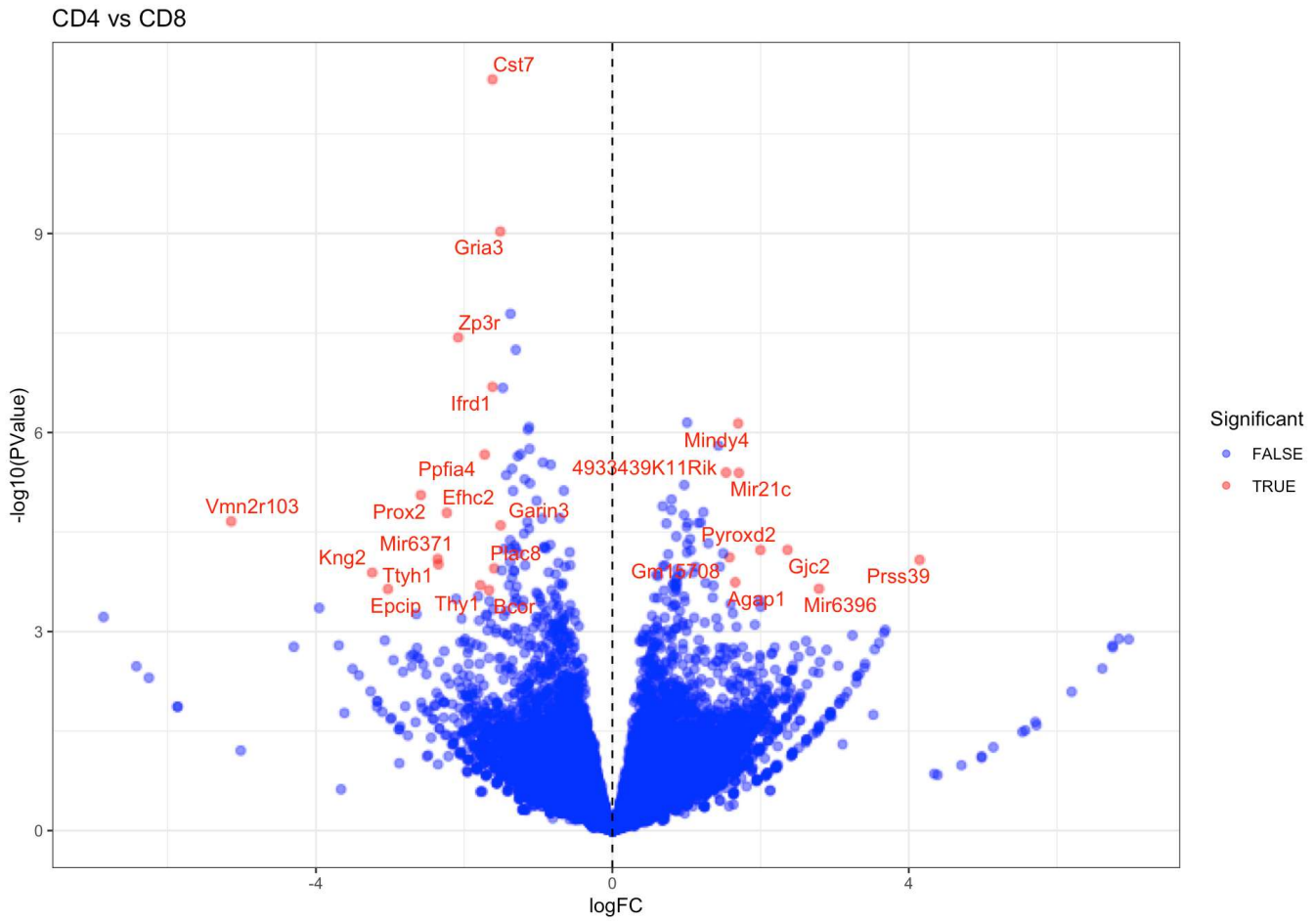
```
             groupCD4 groupCD8 groupDP
T.DP.Th_rep1        0        0       1
T.DP.Th_rep2        0        0       1
T.4.Th_rep1         1        0       0
T.4.Th_rep2         1        0       0
T.8.Th_rep1         0        1       0
T.8.Th_rep2         0        1       0
attr(,"assign")
[1] 1 1 1
attr(,"contrasts")
attr(,"contrasts")$group
[1] "contr.treatment"
```

```
fit <- glmQLFit(dgel, moma)

contr <- makeContrasts(groupCD4 - groupDP,
                       groupCD8 - groupDP,
                       groupCD4 - groupCD8,
                       levels = colnames(moma))

test <- sapply(colnames(contr),
               function(cn) glmQLFTest(fit, contrast = contr[,cn]),
               simplify = FALSE)

sapply(test, function(x) table(decideTests(x, p.value = 0.05, lfc = 1.5)))
```

```
   groupCD4 - groupDP groupCD8 - groupDP groupCD4 - groupCD8
-1                 21                111                  16
0               23602              23421               23637
1                  39                130                   9
```

3. Plot the results for the CD4 vs CD8 condition as a Volcano plot. How can you optimize the analysis by filtering out lowly expressed promoters? Hint: check edgeR User Guide, section 2.7 Filtering.

```
data <- topTags(test[["groupCD4 - groupCD8"]],
                sort="none", n=nrow(dgel))$table |>
    as_tibble() |>
    mutate(Significant = FDR < 0.05 & abs(logFC) > 1.5)

ggplot(data = data,
       mapping = aes(x = logFC,
                     y = -log10(PValue),
                     color = Significant,
                     label = symbol)) +
    rasterize( geom_point(alpha=0.5) ) +
    geom_text_repel(data = data |> filter(Significant),
                    show.legend = FALSE) +
    geom_vline(xintercept = 0, lty = 2) +
    scale_color_manual(values=c(`TRUE`='red', `FALSE`='blue')) +
    labs(title='CD4 vs CD8') +
    theme_bw()
```



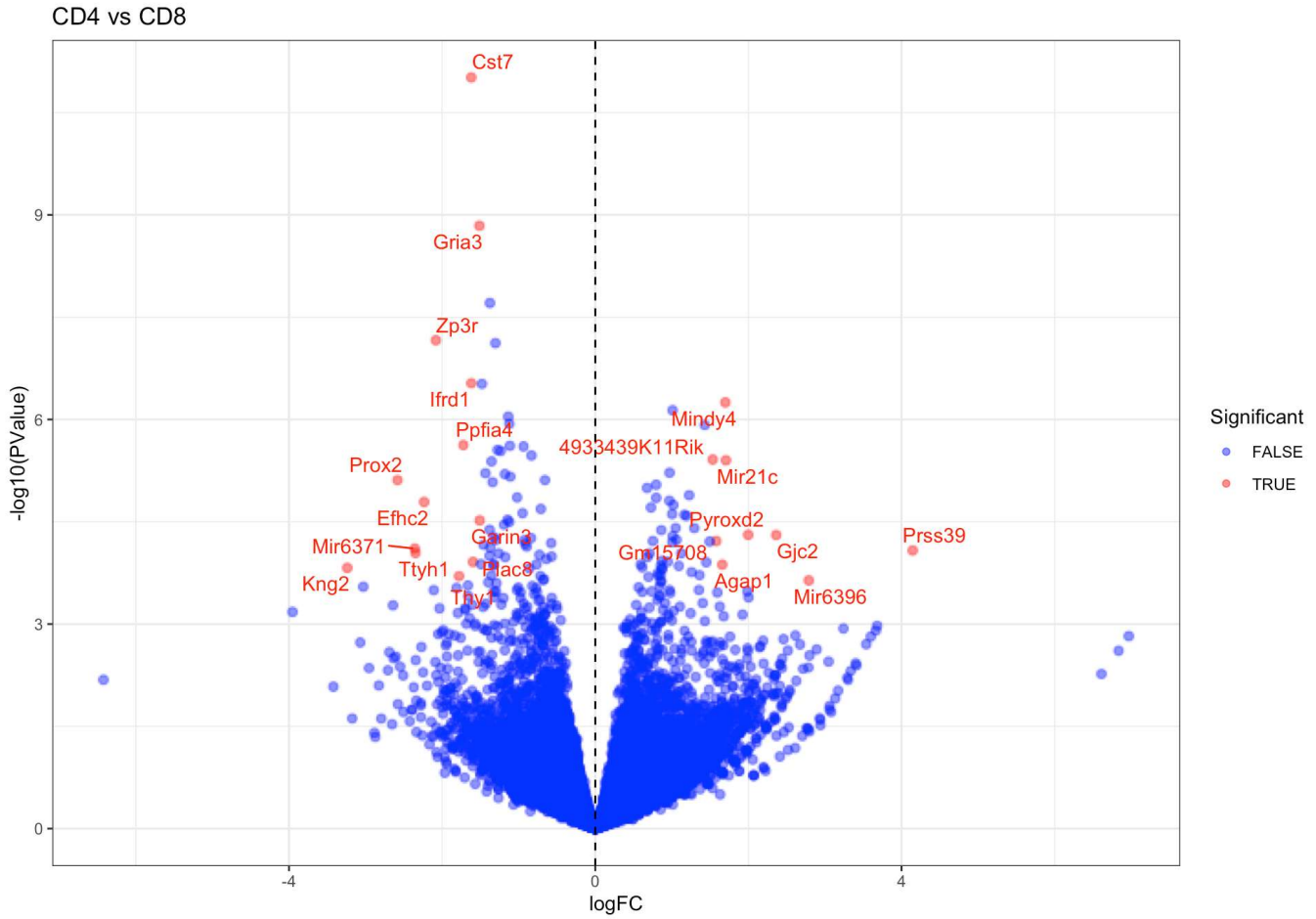

```
## gene filtering based on minimum counts across experimental conditions
ok <- filterByExpr(dgel, design=moma)
table(ok)
```

```
ok
FALSE  TRUE
 1357 22305
```

```
dgel <- dgel[ok,,keep.lib.sizes=FALSE]

fit <- glmQLFit(dgel, moma)

test <- sapply(colnames(contr),
               function(cn) glmQLFTest(fit, contrast = contr[,cn]),
               simplify = FALSE)

sapply(test, function(x) table(decideTests(x, p.value = 0.05, lfc = 1.5)))
```

```
   groupCD4 - groupDP groupCD8 - groupDP groupCD4 - groupCD8
-1                 20                119                  13
0               22249              22057               22283
1                  36                129                   9
```

```
data <- topTags(test[["groupCD4 - groupCD8"]],
                sort="none", n=nrow(dgel))$table |>
    as_tibble() |>
    mutate(Significant = FDR < 0.05 & abs(logFC) > 1.5)

ggplot(data = data,
       mapping = aes(x = logFC,
                     y = -log10(PValue),
                     color = Significant,
                     label = symbol)) +
    rasterize( geom_point(alpha=0.5) ) +
    geom_text_repel(data = data |> filter(Significant),
                    show.legend = FALSE) +
    geom_vline(xintercept = 0, lty = 2) +
    scale_color_manual(values=c(`TRUE`='red', `FALSE`='blue')) +
    labs(title='CD4 vs CD8') +
    theme_bw()
```



4. Repeat the differential promoter analysis, this time only correcting for library composition differences. I.e. instead of using gene- and sample-wise offsets, apply normalization factors to account for library composition biases only (Check: edgeR User Guide, section 2.8 Normalization). How many differentially accessible promoters do you get now? Check the MA plot for the CD4 vs DP condition. Does it look suspicious to you?

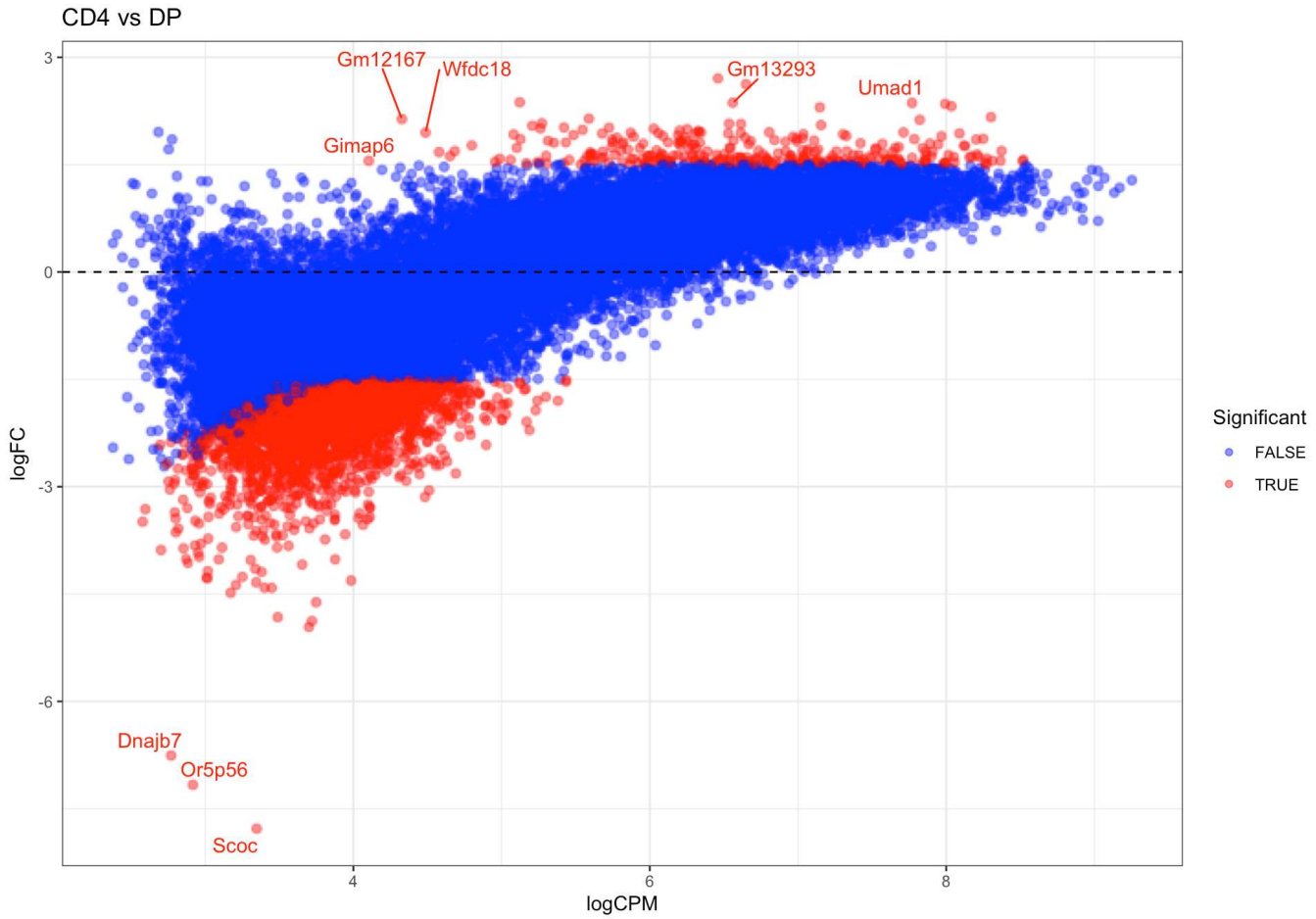
```
dgel2 <- dgel
dgel2$offset <- NULL

dgel2 <- calcNormFactors(dgel2)

fit2 <- glmQLFit(dgel2, moma)

test <- sapply(colnames(contr),
               function(cn) glmQLFTest(fit2, contrast = contr[,cn]),
               simplify = FALSE)

sapply(test, function(x) table(decideTests(x, p.value = 0.05, lfc = 1.5)))
```

```
   groupCD4 - groupDP groupCD8 - groupDP groupCD4 - groupCD8
-1                146               2789                 11
0               22095              19221              22284
1                  64                295                 10
```

```
data <- topTags(test[["groupCD8 - groupDP"]],
                sort="none", n=nrow(dgel))$table |>
    as_tibble() |>
```

```
    mutate(Significant = FDR < 0.05 & abs(logFC) > 1.5)

ggplot(data = data,
       mapping = aes(x = logCPM,
                     y = logFC,
                     color = Significant,
                     label = symbol)) +
    rasterize( geom_point(alpha=0.5) ) +
    geom_text_repel(data = data |> filter(Significant),
                    show.legend = FALSE) +
    geom_hline(yintercept = 0, lty = 2) +
    scale_color_manual(values=c(`TRUE`='red', `FALSE`='blue')) +
    labs(title='CD4 vs DP') +
    theme_bw()
```

```
Warning: ggrepel: 3076 unlabeled data points (too many overlaps). Consider
increasing max.overlaps
```



References

1. Buenrostro JD, Giresi PG, Zaba LC, Chang HY, Greenleaf WJ (2013) Transposition of native chromatin for fast and sensitive epigenomic profiling of open chromatin, DNA-binding proteins and nucleosome position. Nature Methods 10, 1213-1218
2. Corces MR et al. (2017) An improved ATAC-seq protocol reduces background and enables interrogation of frozen tissues. Nature Methods 14, 959–962

3. Grandi FC et al. (2022) Chromatin accessibility profiling by ATAC-seq. Nat Protoc 17, 1518–1552.
4. Yoshida H et al. (2019) The cis-Regulatory Atlas of the Mouse Immune System. Cell. 176(4), 897-912
5. Lun AT and Smyth GK (2016) csaw: a Bioconductor package for differential binding analysis of ChIP-seq data using sliding windows. NAR 44(5):e45
6. Lun AT (2023) The csaw book. https://bioconductor.org/books/release/csawBook/
7. Chen Y et al. (2025) edgeR v4: powerful differential analysis of sequencing data with expanded functionality and improved support for small counts and larger datasets. NAR 53(2):gkaf018

Session info

```
R version 4.5.0 Patched (2025-04-21 r88169)
Platform: x86_64-apple-darwin20
Running under: macOS Sequoia 15.4.1

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.5-x86_64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.5-x86_64/Resources/lib/libRlapack.dylib;
LAPACK version 3.12.1

locale:
[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Europe/Zurich
tzcode source: internal

attached base packages:
[1] grid      stats4    stats     graphics  grDevices utils     datasets
[8] methods   base

other attached packages:
 [1] Gviz_1.52.0
 [2] TxDb.Mmusculus.UCSC.mm10.knownGene_3.10.0
 [3] GenomicFeatures_1.60.0
 [4] BSgenome.Mmusculus.UCSC.mm10_1.4.3
 [5] BSgenome_1.76.0
 [6] BiocIO_1.18.0
 [7] Biostrings_2.76.0
 [8] XVector_0.48.0
 [9] org.Mm.eg.db_3.21.0
[10] AnnotationDbi_1.70.0
[11] edgeR_4.6.1
[12] limma_3.64.0
[13] csaw_1.42.0
[14] rtracklayer_1.68.0
[15] SummarizedExperiment_1.38.1
[16] Biobase_2.68.0
[17] GenomicRanges_1.60.0
[18] GenomeInfoDb_1.44.0
[19] IRanges_2.42.0
[20] S4Vectors_0.46.0
[21] BiocGenerics_0.54.0
```

```
[22] generics_0.1.3
[23] MatrixGenerics_1.20.0
[24] matrixStats_1.5.0
[25] patchwork_1.3.0
[26] ggrastr_1.0.2
[27] ggrepel_0.9.6
[28] RColorBrewer_1.1-3
[29] BiocStyle_2.36.0
[30] lubridate_1.9.4
[31] forcats_1.0.0
[32] stringr_1.5.1
[33] dplyr_1.1.4
[34] purrr_1.0.4
[35] readr_2.1.5
[36] tidyr_1.3.1
[37] tibble_3.2.1
[38] ggplot2_3.5.2
[39] tidyverse_2.0.0
[40] knitr_1.50

loaded via a namespace (and not attached):
  [1] rstudioapi_0.17.1        jsonlite_2.0.0
  [3] magrittr_2.0.3           ggbeeswarm_0.7.2
  [5] farver_2.1.2             rmarkdown_2.29
  [7] vctrs_0.6.5              Cairo_1.6-2
  [9] memoise_2.0.1            Rsamtools_2.24.0
 [11] RCurl_1.98-1.17          base64enc_0.1-3
 [13] htmltools_0.5.8.1        S4Arrays_1.8.0
 [15] progress_1.2.3           curl_6.2.2
 [17] SparseArray_1.8.0        Formula_1.2-5
 [19] htmlwidgets_1.6.4        httr2_1.1.2
 [21] cachem_1.1.0             GenomicAlignments_1.44.0
 [23] lifecycle_1.0.4          pkgconfig_2.0.3
 [25] Matrix_1.7-3             R6_2.6.1
 [27] fastmap_1.2.0            GenomeInfoDbData_1.2.14
 [29] digest_0.6.37            colorspace_2.1-1
 [31] Hmisc_5.2-3              RSQLite_2.3.11
 [33] filelock_1.0.3           labeling_0.4.3
 [35] timechange_0.3.0         mgcv_1.9-3
 [37] httr_1.4.7               abind_1.4-8
 [39] compiler_4.5.0           bit64_4.6.0-1
 [41] withr_3.0.2              backports_1.5.0
 [43] htmlTable_2.4.3          BiocParallel_1.42.0
 [45] DBI_1.2.3                hexbin_1.28.5
 [47] biomaRt_2.64.0           rappdirs_0.3.3
 [49] DelayedArray_0.34.1      rjson_0.2.23
 [51] tools_4.5.0              vipor_0.4.7
 [53] foreign_0.8-90           beeswarm_0.4.0
 [55] nnet_7.3-20              glue_1.8.0
 [57] restfulr_0.0.15          nlme_3.1-168
 [59] checkmate_2.3.2          cluster_2.1.8.1
 [61] gtable_0.3.6             tzdb_0.5.0
 [63] ensembldb_2.32.0         data.table_1.17.0
 [65] hms_1.1.3                utf8_1.2.5
```

```
 [67] metapod_1.16.0           xml2_1.3.8
 [69] pillar_1.10.2            splines_4.5.0
 [71] BiocFileCache_2.16.0     lattice_0.22-7
 [73] deldir_2.0-4             bit_4.6.0
 [75] biovizBase_1.56.0        tidyselect_1.2.1
 [77] locfit_1.5-9.12          gridExtra_2.3
 [79] ProtGenerics_1.40.0      xfun_0.52
 [81] statmod_1.5.0            stringi_1.8.7
 [83] UCSC.utils_1.4.0         lazyeval_0.2.2
 [85] yaml_2.3.10              evaluate_1.0.3
 [87] codetools_0.2-20         interp_1.1-6
 [89] BiocManager_1.30.25      cli_3.6.5
 [91] rpart_4.1.24             dichromat_2.0-0.1
 [93] Rcpp_1.0.14              dbplyr_2.5.0
 [95] png_0.1-8                XML_3.99-0.18
 [97] parallel_4.5.0           blob_1.2.4
 [99] prettyunits_1.2.0        jpeg_0.1-11
[101] latticeExtra_0.6-30      AnnotationFilter_1.32.0
[103] bitops_1.0-9             VariantAnnotation_1.54.0
[105] scales_1.4.0             crayon_1.5.3
[107] rlang_1.1.6              KEGGREST_1.48.0
```