Small RNAseq: Differential Expression Analysis

true

2022-09-27

Environment Setup

```
salloc -N 1 --exclusive -p amd -t 8:00:00
conda env create -f conda-env.yml
conda activate smallrna
```

Downloading datasets

Raw data

Raw data was downloaded from the sequencing facility using the secure link, with `wget` command. The downloaded files were checked for md5sum and compared against list of files expected as per the input samples provided.

```
wget https://oc1.rnet.missouri.edu/xyxz
# link masked
# GEO link will be included later
# merge files of same samples (technical replicates)
paste <(ls *_L001_R1_001.fastq.gz) <(ls *_L002_R1_001.fastq.gz) | \
   sed 's/\t/ /g' |\
   awk '{print "cat",$1,$2" > "$1}' |\
   sed 's/_L001_R1_001.fastq.gz/.fq.gz/2' > concatenate.sh
chmod +x concatenate.sh
sh concatenate.sh
```

Genome/annotation

Additional files required for the analyses were downloaded from GenCode. The downloaded files are as follows:

```
wget https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_mouse/release_M30/GRCm39.primary_assembly.genom
wget https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_mouse/release_M30/gencode.vM30.annotation.gff3
wget https://ftp.ebi.ac.uk/pub/databases/gencode/Gencode_mouse/release_M30/gencode.vM30.annotation.gtf.
gunzip GRCm39.primary_assembly.genome.fa.gz
gunzip gencode.vM30.annotation.gff3.gz
gunzip gencode.vM30.annotation.gtf.gz
```

FastQC (before processing)

```
for fq in *.fq.gz; do
  fastqc --threads $SLURM_JOB_CPUS_PER_NODE $fq;
done
mkdir -p fastqc_pre
mv *.zip *.html fastqc_pre/
```

Mapping

To index the genome, following command was run (in an interactive session).

```
fastaGenome="GRCm39.genome.fa"
gtf="gencode.vM30.annotation.gtf"
STAR --runThreadN $SLURM_JOB_CPUS_PER_NODE \
     --runMode genomeGenerate \
     --genomeDir $(pwd) \
     --genomeFastaFiles $fastaGenome \
     --sjdbGTFfile $gtf \
     --sjdbOverhang 1
```

Each `fastq` file was mapped to the indexed genome as using `runSTAR_map.sh` script shown below:

```
#!/bin/bash
read1=$1
out=$(basename ${read1%%.*})
STARgenomeDir=$(pwd)
# illumina adapter
adapterseq="AGATCGGAAGAGC"
STAR \
    --genomeDir ${STARgenomeDir} \
    --readFilesIn ${read1} \
    --outSAMunmapped Within \
    --readFilesCommand zcat \
    --outSAMtype BAM SortedByCoordinate \
    --quantMode GeneCounts \
    --outFilterMultimapNmax 20 \
    --clip3pAdapterSeq ${adapterseq} \
    --clip3pAdapterMMp 0.1 \
    --outFilterMismatchNoverLmax 0.03 \
    --outFilterScoreMinOverLread 0 \
    --outFilterMatchNminOverLread 0 \
    --outFilterMatchNmin 16 \
    --outFileNamePrefix ${out} \
    --alignSJDBoverhangMin 1000 \
    --alignIntronMax 1 \
    --runThreadN ${SLURM_JOB_CPUS_PER_NODE} \
    --genomeLoad LoadAndKeep \
    --limitBAMsortRAM 30000000000 \
    --outSAMheaderHD "@HD VN:1.4 SO:coordinate"
```

Mapping was run with a simple loop:

```
for fq in *.fq.gz; do
  runSTAR_map.sh $fq;
done
```

Counting Stats

Counts generated by `STAR` with option `--quantMode GeneCounts` were parsed to generate summary stats as well as to extract annotated small RNA feature counts.

```
mkdir -p counts_files
# copy counts for each sample
cp *ReadsPerGene.out.tab counts_files/
cd counts_files
# merge counts
join_files.sh *ReadsPerGene.out.tab |\
   sed 's/ReadsPerGene.out.tab//g' |\
   grep -v "^N_" > counts_star.tsv
# merge stats
join_files.sh *ReadsPerGene.out.tab |\
   sed 's/ReadsPerGene.out.tab//g' |\
   head -n 1 > summary_star.tsv
join_files.sh *ReadsPerGene.out.tab |\
   sed 's/ReadsPerGene.out.tab//g' |\
   grep "^N_" >> summary_star.tsv
# parse GTF to extact gene.id and its biotype:
gtf=gencode.vM30.annotation.gtf
awk 'BEGIN{OFS=FS="\t"} $3=="gene" {split($9,a,";"); print a[1],a[2]}' ${gtf} |\
   awk '{print $4"\t"$2}' |\
   sed 's/"//g' > GeneType_GeneID.tsv
cut -f 1 GeneType_GeneID.tsv | sort |uniq > features.txt
```

The information for `biotype` as provided by the `gencodegenes` were used for categorizing `biotype`.

The `smallRNA` group consists of following `biotype`:

```
miRNA
misc_RNA
scRNA
snRNA
snoRNA
sRNA
scaRNA
```

The full table is as follows:

```
library(knitr)
setwd("/work/LAS/geetu-lab/arnstrm/mouse.trophoblast.smallRNAseq")
file1="assets/GeneType_Group.tsv"
info <-
  read.csv(
```

```
    file1,
    header = TRUE,
    sep = "\t",
    stringsAsFactors = TRUE
  )
kable(info, caption = "Table 1: biotype and its groupings")
```

Table 1: Table 1: biotype and its groupings

biotype	group
protein_coding	coding_genes
pseudogene	pseudogenes
TR_C_gene	Ig_genes
TR_D_gene	Ig_genes
TR_J_gene	Ig_genes
TR_V_gene	Ig_genes
IG_C_gene	Ig_genes
IG_D_gene	Ig_genes
IG_J_gene	Ig_genes
IG_LV_gene	Ig_genes
IG_V_gene	Ig_genes
TR_J_pseudogene	pseudogenes
TR_V_pseudogene	pseudogenes
IG_C_pseudogene	pseudogenes
IG_D_pseudogene	pseudogenes
IG_pseudogene	pseudogenes
IG_V_pseudogene	pseudogenes
lncRNA	long_non_conding_RNA
miRNA	non_conding_RNA
misc_RNA	non_conding_RNA
ribozyme	non_conding_RNA
rRNA	non_conding_RNA
scaRNA	non_conding_RNA
scRNA	non_conding_RNA
snoRNA	non_conding_RNA
snRNA	non_conding_RNA
sRNA	non_conding_RNA
Mt_rRNA	non_conding_RNA
Mt_tRNA	non_conding_RNA
processed_pseudogene	pseudogenes
unprocessed_pseudogene	pseudogenes
translated_unprocessed_pseudogene	pseudogenes
transcribed_processed_pseudogene	pseudogenes
transcribed_unitary_pseudogene	pseudogenes
transcribed_unprocessed_pseudogene	pseudogenes
unitary_pseudogene	pseudogenes
TEC	unconfirmed_genes

A samples table (`samples.tsv`) categorizing samples to its condition were also generated:

```
file2="assets/samples.tsv"
samples <-
  read.csv(
    file2,
    header = TRUE,
    sep = "\t",
    stringsAsFactors = TRUE
  )
kable(samples, caption = "Table 2: Samples in the study")
```

Table 2: Table 2: Samples in the study

Sample	Group
Dif_D6_1_S4	Diff
Dif_D6_2_S3	Diff
Dif_D6_3_S2	Diff
Dif_D6_4_S1	Diff
Undif_D2_1_S8	Undf
Undif_D2_2_S7	Undf
Undif_D2_3_S6	Undf
Undif_D2_4_S5	Undf

This information was then merged withe counts table to generate QC plots:

```
awk 'BEGIN{OFS=FS="\t"}FNR==NR{a[$1]=$2;next}{ print $2,$1,a[$1]}' \
    GeneType_Group.tsv GeneType_GeneID.tsv > GeneID_GeneType_Group.tsv

awk 'BEGIN{OFS=FS="\t"}FNR==NR{a[$1]=$2"\t"$3;next}{print $1,a[$1],$0}' \
    GeneID_GeneType_Group.tsv counts_star.tsv |\
    cut -f 1-3,5- > processed_counts_star.tsv
```

Plotting the mapping summary and count statistics for various biotypes:

```
library(scales)
library(tidyverse)
#library(plotly)
```

```
setwd("/work/LAS/geetu-lab/arnstrm/mouse.trophoblast.smallRNAseq")
file1="assets/processed_counts_star.tsv"
file2="assets/summary_stats_star.tsv"
counts <-
  read.csv(
    file1,
    sep = "\t",
    stringsAsFactors = TRUE
  )
subread <-
  read.csv(
    file2,
```

```
    sep = "\t",
    stringsAsFactors = TRUE
  )
# convert long format
counts.long <- gather(counts, Sample, Count, Dif_D6_1_S4:Undif_D2_4_S5, factor_key=TRUE)
subread.long <- gather(subread, Sample,  Count, Dif_D6_1_S4:Undif_D2_4_S5, factor_key=TRUE)
# organize
counts.long$Group <-
  factor(
    counts.long$Group,
    levels = c(
      "coding_genes",
      "non_conding_RNA",
      "long_non_conding_RNA",
      "pseudogenes",
      "unconfirmed_genes",
      "Ig_genes"
    )
  )

subread.long$Assignments <-
  factor(
    subread.long$Assignments,
    levels = c(
      "N_input",
      "N_unmapped",
      "N_multimapping",
      "N_unique",
      "N_ambiguous",
      "N_noFeature"
    )
  )
```

```
ggplot(subread.long, aes(x = Assignments, y = Count, fill = Assignments)) +
  geom_bar(stat = 'identity') +
  labs(x = "Subread assingments", y = "reads") + theme_minimal() +
  scale_y_continuous(labels = label_comma()) +
  theme(
    axis.text.x = element_text(
      angle = 45,
      vjust = 1,
      hjust = 1,
      size = 12
      ),
    strip.text = element_text(
      face = "bold",
      color = "gray35",
      hjust = 0,
      size = 10
    ),
    strip.background = element_rect(fill = "white", linetype = "blank"),
    legend.position = "none"
  ) +
```

```
  facet_wrap("Sample", scales = "free_y", ncol = 4)
```

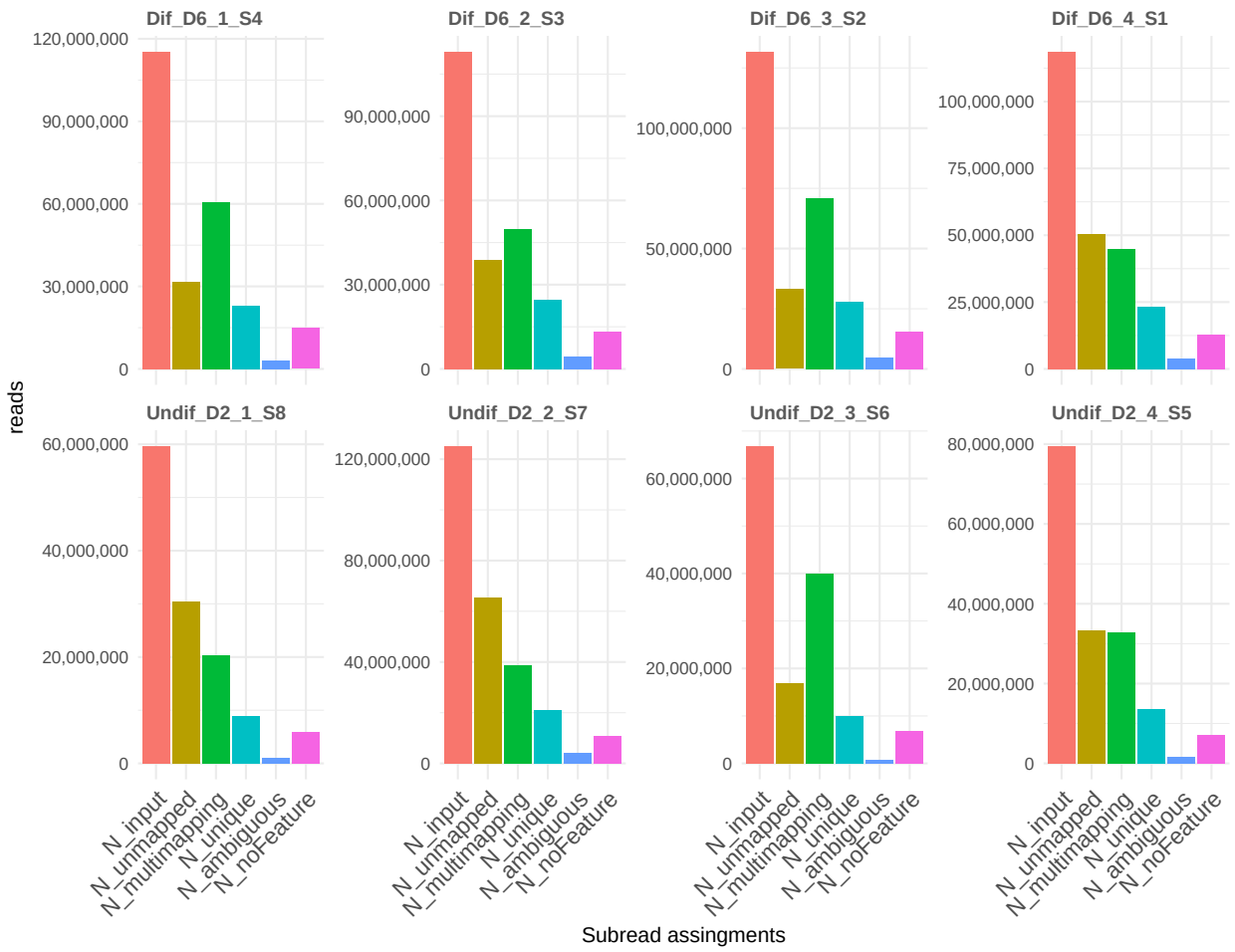


Figure 1: Figure 1: STAR read mapping and feature assignment. Here, `N_input` is total input reads, `N_unmapped` is reads that were either too short to map after adapter removal or had higher mismatch rate to place reliably on the genome, `N_multimapping` is reads mapped to multiple loci, `N_unique` is reads mapped to unique loci. A subset of `N_unique` reads that were unable to clearly assign to a feature or assign any feature at all are grouped as `N_ambigious` or `N_noFeature`, respectively

```
g <- ggplot(counts.long, aes(x = Group, y = Count, fill = Group)) +
  geom_bar(stat = 'sum') +
  labs(x = "biotype", y = "read counts") + theme_minimal() +
  scale_y_continuous(labels = label_comma()) +
  theme(
    axis.text.x = element_text(
      angle = 45,
      vjust = 1,
      hjust = 1,
      size = 12
    ),
    strip.text = element_text(
      face = "bold",
```

```
      color = "gray35",
      hjust = 0,
      size = 10
    ),
    strip.background = element_rect(fill = "white", linetype = "blank"),
    legend.position = "none"
  ) +
  facet_wrap("Sample", scales = "free_y", ncol = 4)
#ggplotly(g)
g
```

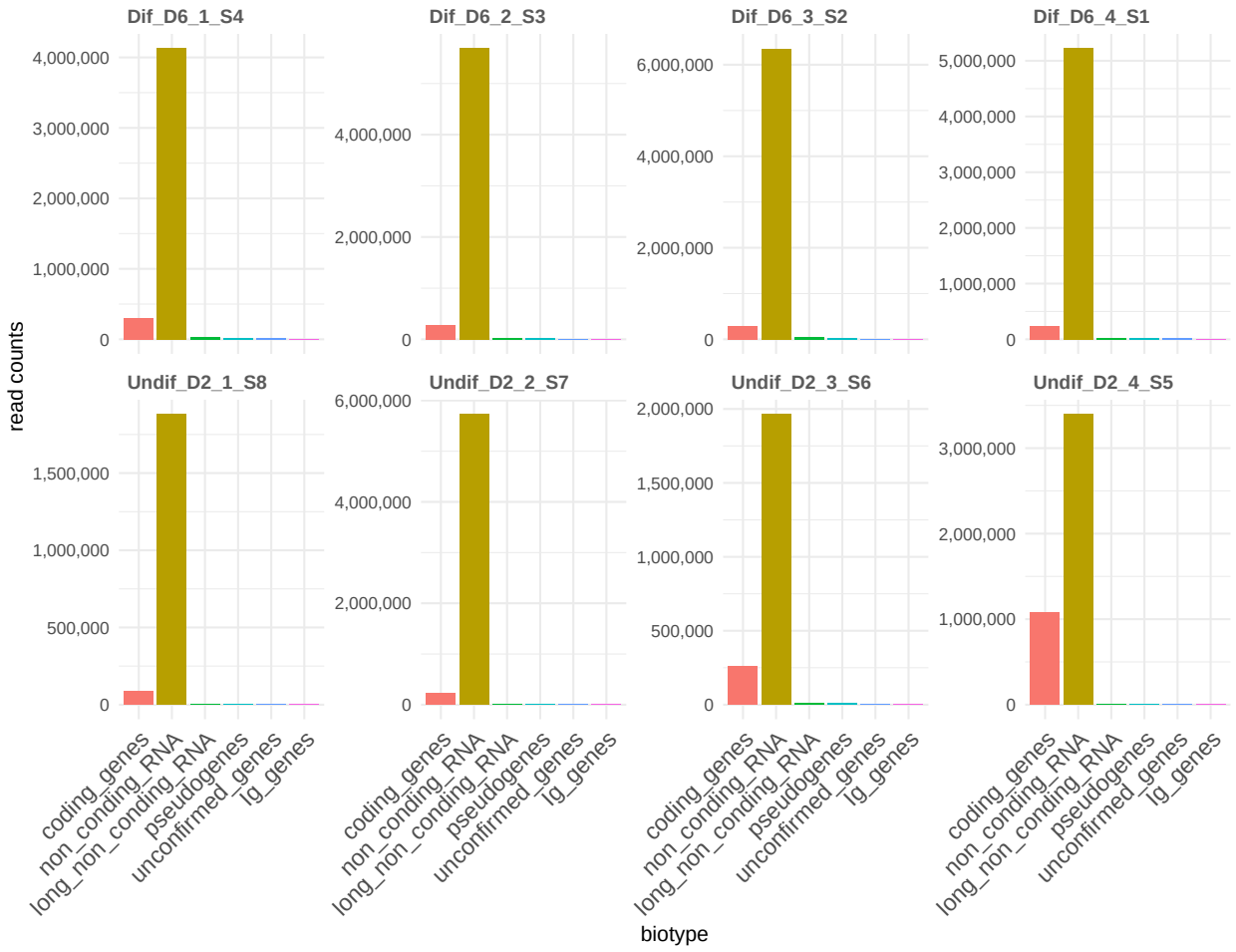


Figure 2: Figure 2: Features with read counts

```
counts.nc <- filter(counts.long, Group %in% "non_conding_RNA" )
counts.nc$GeneType <-
  factor(
    counts.nc$GeneType,
    levels = c(
      "miRNA",
      "misc_RNA",
      "snoRNA",
```

```
      "snRNA",
      "sRNA",
      "scRNA",
      "scaRNA",
      "Mt_tRNA",
      "Mt_rRNA",
      "rRNA",
      "ribozyme"
    )
  )

g <- ggplot(counts.nc, aes(x = GeneType, y = Count, fill = GeneType)) +
  geom_bar(stat = 'sum') +
  labs(x = "biotype", y = "read counts") + theme_minimal() +
  scale_y_continuous(labels = label_comma()) +
  theme(
    axis.text.x = element_text(
      angle = 45,
      vjust = 1,
      hjust = 1,
      size = 12
    ),
    strip.text = element_text(
      face = "bold",
      color = "gray35",
      hjust = 0,
      size = 10
    ),
    strip.background = element_rect(fill = "white", linetype = "blank"),
    legend.position = "none"
  ) +
  facet_wrap("Sample", scales = "free_y", ncol = 4)
#ggplotly(g)
g
```

subset the counts file to select only smallRNA genes

```
snrna <- c('miRNA',
           'misc_RNA',
           'scRNA',
           'snRNA',
           'snoRNA',
           'sRNA',
           'scaRNA')
cts <- filter(counts, GeneType %in% snrna) %>%
  select(Geneid, Dif_D6_1_S4:Undif_D2_4_S5)
write_delim(cts, file = "assets/noncoding_counts_star.tsv", delim = "\t")
```

This `noncoding_counts_star.tsv` and `samples.tsv` file will be used for `DESeq2` analyses.

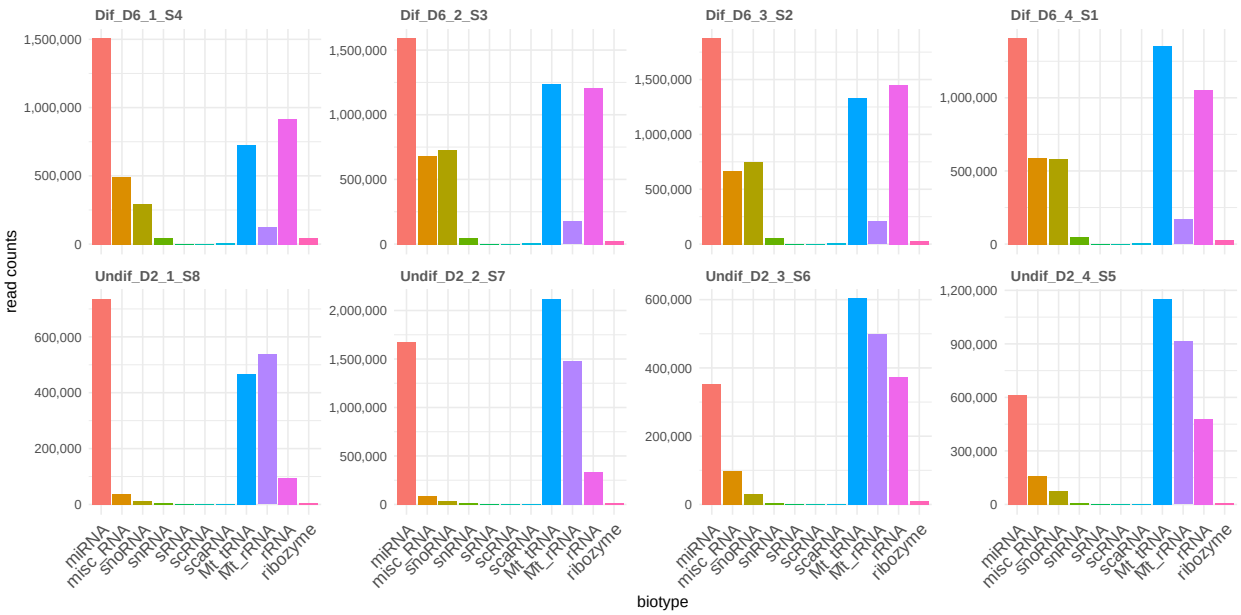


Figure 3: Figure 3: non-coding biotype read counts

DESeq2

For the next steps, we used `DESeq2` for performing the DE analyses. Results were visualized as volcano plots and tables were exported to excel.

Load packages

```
setwd("/work/LAS/geetu-lab/arnstrm/mouse.trophoblast.smallRNAseq")
library(DESeq2)
library(RColorBrewer)
library(pheatmap)
library(genefilter)
library(ggrepel)
```

Import counts and sample metadata

The `counts` data and its associated metadata (`coldata`) are imported for analyses.

```
counts = 'assets/noncoding_counts_star.tsv'
groupFile = 'assets/samples.tsv'
coldata <-
  read.csv(
    groupFile,
    row.names = 1,
    sep = "\t",
    stringsAsFactors = TRUE
  )
cts <- as.matrix(read.csv(counts, sep = "\t", row.names = "Geneid"))
```

Reorder columns of `cts` according to `coldata` rows. Check if samples in both files match.

```
colnames(cts)
#> [1] "Dif_D6_1_S4"   "Dif_D6_2_S3"   "Dif_D6_3_S2"   "Dif_D6_4_S1"
#> [5] "Undif_D2_1_S8" "Undif_D2_2_S7" "Undif_D2_3_S6" "Undif_D2_4_S5"
all(rownames(coldata) %in% colnames(cts))
#> [1] TRUE
cts <- cts[, rownames(coldata)]
```

Normalize

The batch corrected read counts are then used for running DESeq2 analyses

```
dds <- DESeqDataSetFromMatrix(countData = cts,
                              colData = coldata,
                              design = ~ Group)
vsd <- vst(dds, blind = FALSE, nsub =500)
keep <- rowSums(counts(dds)) >= 5
dds <- dds[keep, ]
dds <- DESeq(dds)
dds
#> class: DESeqDataSet
#> dim: 1266 8
#> metadata(1): version
#> assays(4): counts mu H cooks
#> rownames(1266): ENSMUSG00000119106.1 ENSMUSG00000119589.1 ...
#>   ENSMUSG00000065444.3 ENSMUSG00000077869.3
#> rowData names(22): baseMean baseVar ... deviance maxCooks
#> colnames(8): Dif_D6_1_S4 Dif_D6_2_S3 ... Undif_D2_3_S6 Undif_D2_4_S5
#> colData names(2): Group sizeFactor
```

```
vst <- assay(vst(dds, blind = FALSE, nsub = 500))
vsd <- vst(dds, blind = FALSE, nsub = 500)
pcaData <-
  plotPCA(vsd,
          intgroup = "Group",
          returnData = TRUE)
percentVar <- round(100 * attr(pcaData, "percentVar"))
```

PCA plot for QC

PCA plot for the dataset that includes all libraries.

```
rv <- rowVars(assay(vsd))
select <-
  order(rv, decreasing = TRUE)[seq_len(min(500, length(rv)))]
pca <- prcomp(t(assay(vsd)[select, ]))
percentVar <- pca$sdev ^ 2 / sum(pca$sdev ^ 2)
intgroup = "Group"
intgroup.df <- as.data.frame(colData(vsd)[, intgroup, drop = FALSE])
group <- if (length(intgroup) == 1) {
```

```
  factor(apply(intgroup.df, 1, paste, collapse = " : "))
}
d <- data.frame(
  PC1 = pca$x[, 1],
  PC2 = pca$x[, 2],
  intgroup.df,
  name = colnames(vsd)
)
```

plot PCA for components 1 and 2

```
g <- ggplot(d, aes(PC1, PC2, color = Group)) +
  scale_shape_manual(values = 1:8) +
  theme_bw() +
  theme(legend.title = element_blank()) +
  geom_point(size = 2, stroke = 2) +
  xlab(paste("PC1", round(percentVar[1] * 100, 2), "% variance")) +
  ylab(paste("PC2", round(percentVar[2] * 100, 2), "% variance"))
#ggplotly(g)
g
```

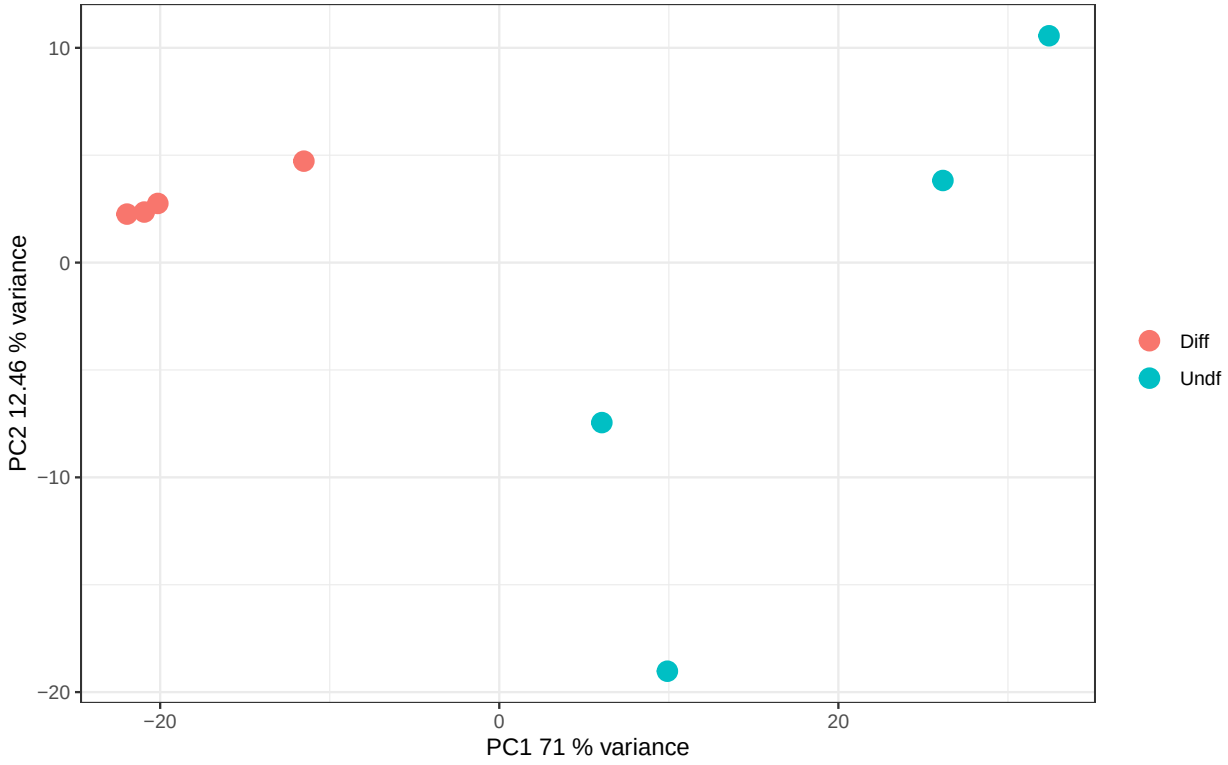


Figure 4: Figure 4: PCA plot for the first 2 principal components

Sample distance for QC

```
sampleDists <- dist(t(assay(vsd)))
sampleDistMatrix <- as.matrix( sampleDists )
rownames(sampleDistMatrix) <- colnames(vsd)
colnames(sampleDistMatrix) <- NULL
colors <- colorRampPalette( rev(brewer.pal(9, "Blues")) )(255)
pheatmap(sampleDistMatrix,
         clustering_distance_rows = sampleDists,
         clustering_distance_cols = sampleDists,
         col = colors)
```

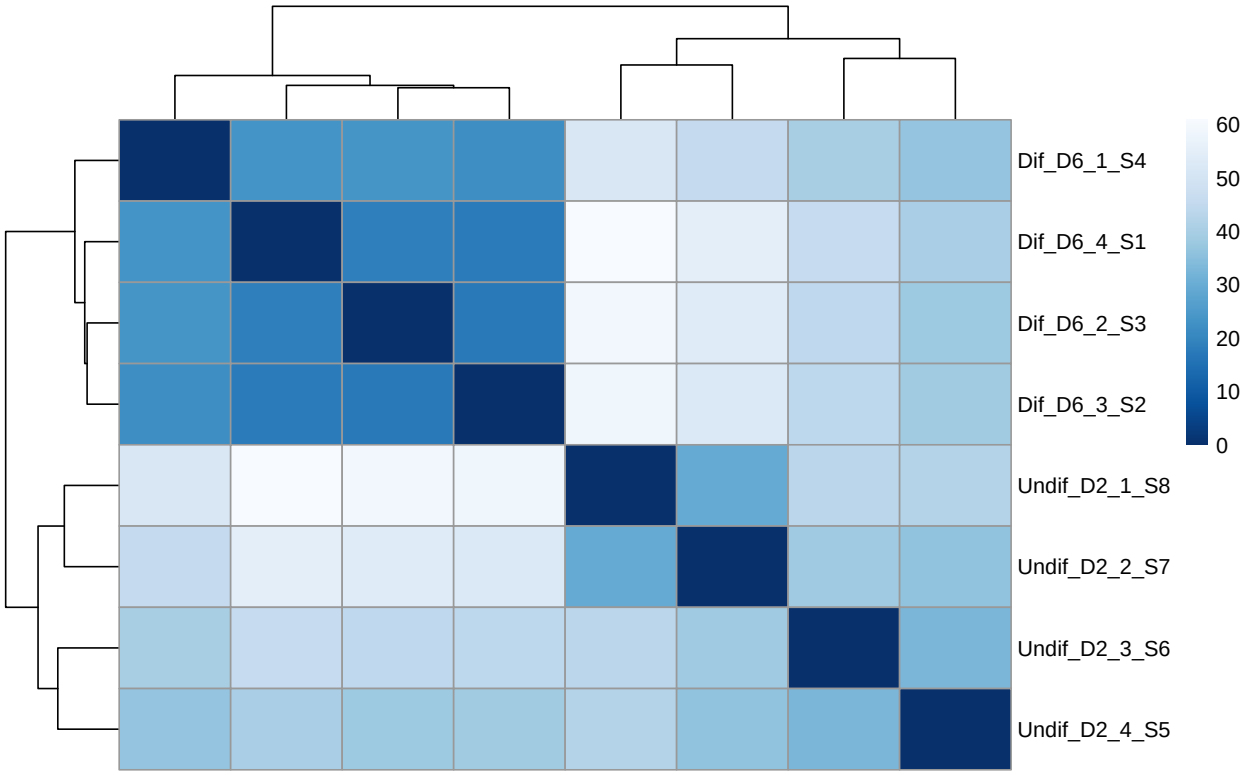


Figure 5: Figure 5: Euclidean distance between samples

Set contrasts and find DE genes

```
resultsNames(dds)
#> [1] "Intercept"           "Group_Undf_vs_Diff"
res.UndfvsDiff <- results(dds, contrast = c("Group", "Undf", "Diff"))
table(res.UndfvsDiff$padj < 0.05)
#>
#> FALSE  TRUE
#>   579   294
res.UndfvsDiff <- res.UndfvsDiff[order(res.UndfvsDiff$padj),]
```

```
res.UndfvsDiffdata <-
  merge(
    as.data.frame(res.UndfvsDiff),
    as.data.frame(counts(dds, normalized = TRUE)),
    by = "row.names",
    sort = FALSE
  )
names(res.UndfvsDiffdata)[1] <- "Gene"
write_delim(res.UndfvsDiffdata, file = "DESeq2results-UndfvsDiff_fc.tsv", delim = "\t")
```

Volcano plots

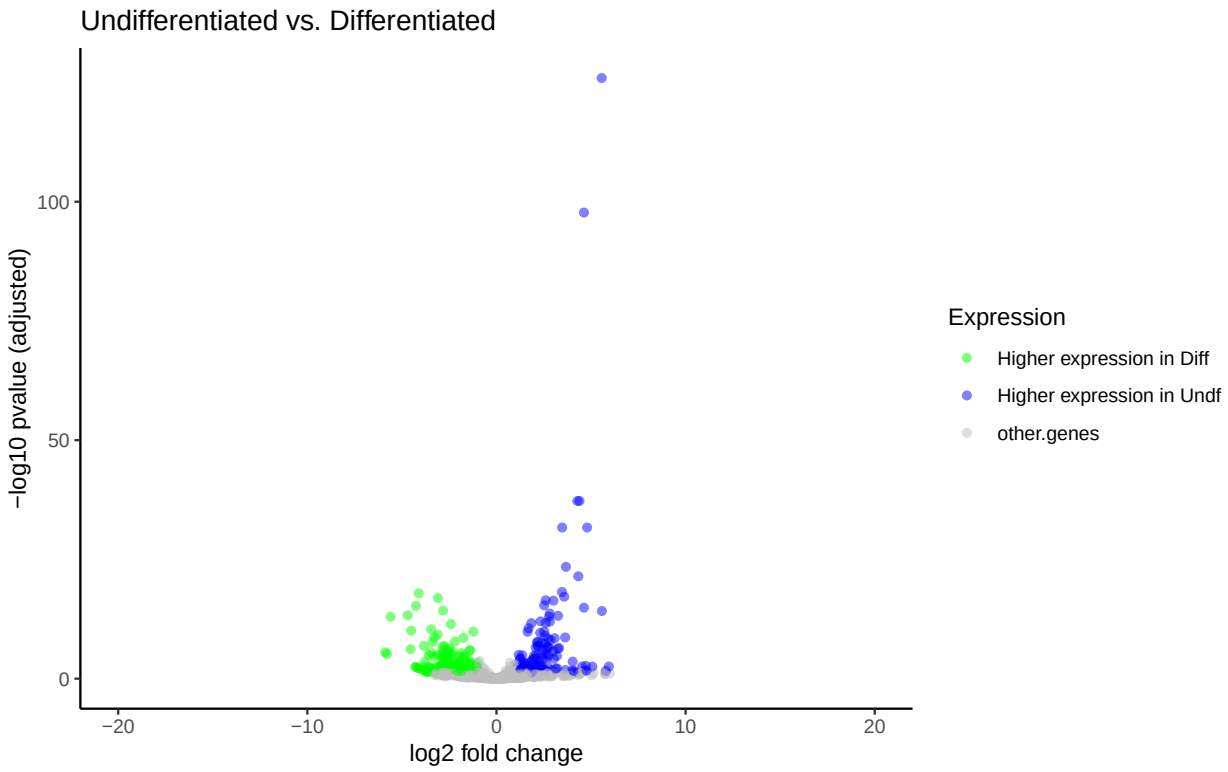
```
mart <-
  read.csv(
    "assets/mart_export.txt",
    sep = "\t",
    stringsAsFactors = TRUE,
    header = TRUE
  ) #this object was obtained from Ensembl as we illustrated in "Creating gene lists"
```

```
volcanoPlots2 <-
  function(res.se,
           string,
           first,
           second,
           color1,
           color2,
           color3,
           ChartTitle) {
    res.se <- res.se[order(res.se$padj), ]
    res.se <-
      rownames_to_column(as.data.frame(res.se[order(res.se$padj), ]))
    names(res.se)[1] <- "Gene"
    res.data <-
      merge(res.se,
            mart,
            by.x = "Gene",
            by.y = "geneid.version")
    res.data <- res.data %>% mutate_all(na_if, "")
    res.data <- res.data %>% mutate_all(na_if, " ")
    res.data <-
      res.data %>% mutate(gene_symbol = coalesce(gene.symbol, Gene))
    res.data$diffexpressed <- "other.genes"
    res.data$diffexpressed[res.data$log2FoldChange >= 1 &
                             res.data$padj <= 0.05] <-
      paste("Higher expression in", first)
    res.data$diffexpressed[res.data$log2FoldChange <= -1 &
                             res.data$padj <= 0.05] <-
      paste("Higher expression in", second)
    res.data$delabel <- ""
    res.data$delabel[res.data$log2FoldChange >= 1
```

```
                      & res.data$padj <= 0.05
                      &
                        !is.na(res.data$padj)] <-
      res.data$gene_symbol[res.data$log2FoldChange >= 1
                           &
                             res.data$padj <= 0.05
                           &
                             !is.na(res.data$padj)]
    res.data$delabel[res.data$log2FoldChange <= -1
                      & res.data$padj <= 0.05
                      &
                        !is.na(res.data$padj)] <-
      res.data$gene_symbol[res.data$log2FoldChange <= -1
                           &
                             res.data$padj <= 0.05
                           &
                             !is.na(res.data$padj)]
    ggplot(res.data,
             aes(
               x = log2FoldChange,
               y = -log10(padj),
               col = diffexpressed,
               label = delabel
             )) +
      geom_point(alpha = 0.5) +
      xlim(-20, 20) +
      theme_classic() +
      scale_color_manual(name = "Expression", values = c(color1, color2, color3)) +
      # geom_text_repel(
      #   data = subset(res.data, padj <= 0.05),
      #   max.overlaps  = 15,
      #   show.legend = F,
      #   min.segment.length = Inf,
      #   seed = 42,
      #   box.padding = 0.5
      # ) +
      ggtitle(ChartTitle) +
      xlab(paste("log2 fold change")) +
      ylab("-log10 pvalue (adjusted)") +
      theme(legend.text.align = 0)
}
```

```
g <- volcanoPlots2(
  res.UndfvsDiff,
  "UndfvsDiff",
  "Undf",
  "Diff",
  "green",
  "blue",
  "grey",
  ChartTitle = "Undifferentiated vs. Differentiated"
)
#ggplotly(g)
```

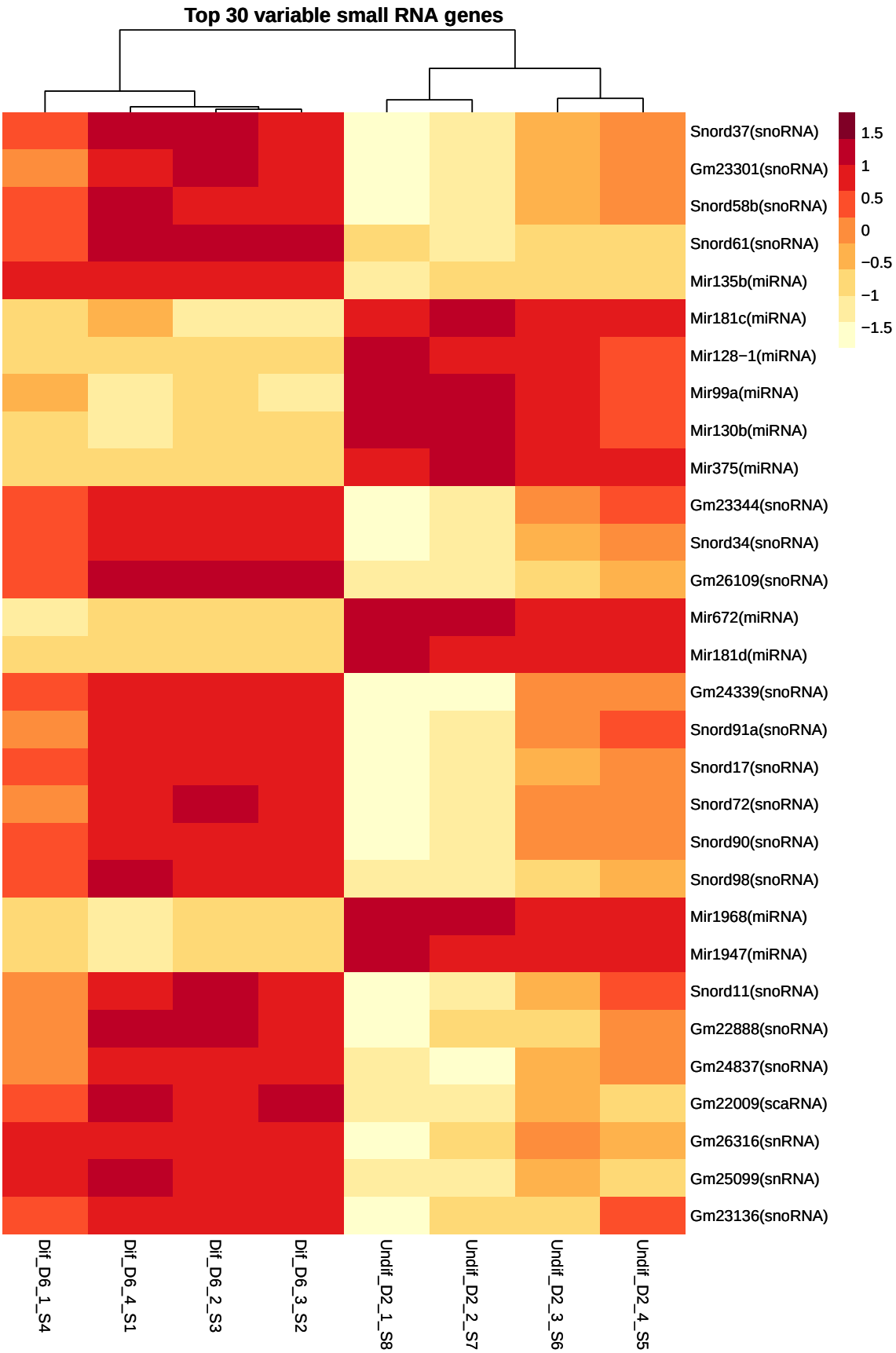
```
g
#> Warning: Removed 393 rows containing missing values (geom_point).
```



Heatmap

Heatmap for the top 30 variable genes:

```
topVarGenes <- head(order(rowVars(assay(vsd)), decreasing = TRUE), 30)
mat  <- assay(vsd)[ topVarGenes, ]
mat  <- mat - rowMeans(mat)
mat2 <-    merge(mat,
          mart,
          by.x = 'row.names',
          by.y = "geneid.version")
rownames(mat2) <- mat2[,10]
mat2 <- mat2[2:9]
heat_colors <- brewer.pal(9, "YlOrRd")
g <- pheatmap(
      mat2,
      color = heat_colors,
      main = "Top 30 variable small RNA genes",
      cluster_rows = F,
      cluster_cols  = T,
      show_rownames = T,
      border_color = NA,
      fontsize = 10,
      scale = "row",
      fontsize_row = 10
    )
    g
```


Top 30 variable small RNA genes
Snord37(snoRNA)
Gm23301(snoRNA)
Snord58b(snoRNA)
Snord61(snoRNA)
Mir135b(miRNA)
Mir181c(miRNA)
Mir128−1(miRNA)
Mir99a(miRNA)
Mir130b(miRNA)
Mir375(miRNA)
Gm23344(snoRNA)
Snord34(snoRNA)
Gm26109(snoRNA)
Mir672(miRNA)
Mir181d(miRNA)
Gm24339(snoRNA)
Snord91a(snoRNA)
Snord17(snoRNA)
Snord72(snoRNA)
Snord90(snoRNA)
Snord98(snoRNA)
Mir1968(miRNA)
Mir1947(miRNA)
Snord11(snoRNA)
Gm22888(snoRNA)
Gm24837(snoRNA)
Gm22009(scaRNA)
Gm26316(snRNA)
Gm25099(snRNA)
Gm23136(snoRNA)
1.5
1
0.5
0
−0.5
−1
−1.5
Dif_D6_1_S4
Dif_D6_4_S1
Dif_D6_2_S3
Dif_D6_3_S2
Undif_D2_1_S8
Undif_D2_2_S7
Undif_D2_3_S6
Undif_D2_4_S5

##

MultiQC report:

MultiQC report is available at this link

Session Information

```
options(max.print=999999)
sessionInfo()
#> R version 4.2.1 (2022-06-23)
#> Platform: x86_64-pc-linux-gnu (64-bit)
#> Running under: Ubuntu 20.04.4 LTS
#>
#> Matrix products: default
#> BLAS:   /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so.3
#> LAPACK: /usr/lib/x86_64-linux-gnu/openblas-pthread/liblapack.so.3
#>
#> locale:
#>  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
#>  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
#>  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
#>  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
#>  [9] LC_ADDRESS=C               LC_TELEPHONE=C
#> [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
#>
#> attached base packages:
#> [1] stats4    stats     graphics  grDevices utils     datasets  methods
#> [8] base
#>
#> other attached packages:
#>  [1] ggrepel_0.9.1               genefilter_1.78.0
#>  [3] pheatmap_1.0.12             RColorBrewer_1.1-3
#>  [5] DESeq2_1.36.0               SummarizedExperiment_1.26.1
#>  [7] Biobase_2.56.0              MatrixGenerics_1.8.1
#>  [9] matrixStats_0.62.0          GenomicRanges_1.48.0
#> [11] GenomeInfoDb_1.32.2         IRanges_2.30.0
#> [13] S4Vectors_0.34.0            BiocGenerics_0.42.0
#> [15] forcats_0.5.1               stringr_1.4.0
#> [17] dplyr_1.0.9                 purrr_0.3.4
#> [19] readr_2.1.2                 tidyr_1.2.0
#> [21] tibble_3.1.8                ggplot2_3.3.6
#> [23] tidyverse_1.3.2             scales_1.2.0
#> [25] knitr_1.39
#>
#> loaded via a namespace (and not attached):
#>  [1] bitops_1.0-7           fs_1.5.2               lubridate_1.8.0
#>  [4] bit64_4.0.5            httr_1.4.3             tools_4.2.1
#>  [7] backports_1.4.1        utf8_1.2.2             R6_2.5.1
#> [10] DBI_1.1.3              colorspace_2.0-3       withr_2.5.0
#> [13] tidyselect_1.1.2       bit_4.0.4              compiler_4.2.1
#> [16] cli_3.3.0              rvest_1.0.2            xml2_1.3.3
#> [19] DelayedArray_0.22.0    labeling_0.4.2         digest_0.6.29
#> [22] rmarkdown_2.14         XVector_0.36.0         pkgconfig_2.0.3
```

```
#> [25] htmltools_0.5.3        dbplyr_2.2.1            fastmap_1.1.0
#> [28] highr_0.9              rlang_1.0.4             readxl_1.4.0
#> [31] RSQLite_2.2.15         rstudioapi_0.13         farver_2.1.1
#> [34] generics_0.1.3         jsonlite_1.8.0          BiocParallel_1.30.3
#> [37] vroom_1.5.7            googlesheets4_1.0.0     RCurl_1.98-1.8
#> [40] magrittr_2.0.3         GenomeInfoDbData_1.2.8 Matrix_1.4-1
#> [43] Rcpp_1.0.9             munsell_0.5.0           fansi_1.0.3
#> [46] lifecycle_1.0.1        stringi_1.7.8           yaml_2.3.5
#> [49] zlibbioc_1.42.0        blob_1.2.3              grid_4.2.1
#> [52] parallel_4.2.1         crayon_1.5.1            lattice_0.20-45
#> [55] splines_4.2.1          Biostrings_2.64.0       haven_2.5.0
#> [58] annotate_1.74.0        KEGGREST_1.36.3         hms_1.1.1
#> [61] locfit_1.5-9.6         pillar_1.8.0            geneplotter_1.74.0
#> [64] codetools_0.2-18       XML_3.99-0.10           reprex_2.0.1
#> [67] glue_1.6.2             evaluate_0.15           modelr_0.1.8
#> [70] png_0.1-7              vctrs_0.4.1             tzdb_0.3.0
#> [73] cellranger_1.1.0       gtable_0.3.0            assertthat_0.2.1
#> [76] cachem_1.0.6           xfun_0.31               xtable_1.8-4
#> [79] broom_1.0.0            survival_3.3-1          googledrive_2.0.0
#> [82] gargle_1.2.0           memoise_2.0.1           AnnotationDbi_1.58.0
#> [85] ellipsis_0.3.2
```