

DE analysis - Day8

Sergey Naumenko

2021-04-08

Contents

Overview	2
Checking to see that the transcript to gene mapping is correct	8
Sanity check that metadata matches your expression	8
Run DESeq2	9
Wald test	9
DEGreport QC	10
Size factor QC - samples 1-20	10
Mean-Variance QC plots	11
response	11
ER	11
tumor_percentage_high	12
Covariates effect on count data	13
Covariates correlation with metrics	14
Sample-level QC analysis	16
PCA - response	16
PCA - ER	17
PCA - tumor_percentage	18
PCA - tumor_percentage_high	19
PCA - date_of	20
Inter-correlation analysis	21
Without study_id	21
With study_id	22
Response Yes vs No for Day 8 - see Table13	23
ER : Positive vs Negative for Day8 - Table 14	24
tumor_percentage_high : High vs Low for Day8- Table 15	25
date_of: 20180323 vs 20180228 - for Day8: Table 16	26
Visualization	27

Functional analysis	41
Biological Process (BP)	41
R session	42

Overview

- Principal Investigator: Beth Overmoyer
- Experiment: RNAseq_analysis_of_inflammatory_breast_cancer_hbc04141
- study 6 was excluded because of low read depth in 3373-3
- <https://www.bioconductor.org/packages/release/bioc/vignettes/DEGreport/inst/doc/DEGreport.html>
- AnnotationHub. We use ensembl version matching bcbio pipeline - v94.
- HBC materials
- HBC materials - functional analysis
- <http://bioconductor.org/packages/release/bioc/vignettes/DESeq2/inst/doc/DESeq2.html>
- this is DE for Day8 samples

```
## Setup
### Bioconductor and CRAN libraries used

library(DESeq2)

## Loading required package: S4Vectors
## Loading required package: stats4
## Loading required package: BiocGenerics
## Loading required package: parallel
##
## Attaching package: 'BiocGenerics'
## The following objects are masked from 'package:parallel':
##
##   clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##   clusterExport, clusterMap, parApply, parCapply, parLapply,
##   parLapplyLB, parRapply, parSapply, parSapplyLB
## The following objects are masked from 'package:stats':
##
##   IQR, mad, sd, var, xtabs
## The following objects are masked from 'package:base':
##
##   anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##   dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##   grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##   order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##   rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##   union, unique, unsplit, which.max, which.min
##
## Attaching package: 'S4Vectors'
## The following object is masked from 'package:base':
##
##   expand.grid
```

```

## Loading required package: IRanges
## Loading required package: GenomicRanges
## Loading required package: GenomeInfoDb
## Loading required package: SummarizedExperiment
## Loading required package: MatrixGenerics
## Loading required package: matrixStats
##
## Attaching package: 'MatrixGenerics'
## The following objects are masked from 'package:matrixStats':
##
##   colAlls, colAnyNAs, colAnys, colAvgPerRowSet, colCollapse,
##   colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##   colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##   colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##   colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##   colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##   colWeightedMeans, colWeightedMedians, colWeightedSds,
##   colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgPerColSet,
##   rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##   rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##   rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##   rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##   rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,
##   rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##   rowWeightedSds, rowWeightedVars
## Loading required package: Biobase
## Welcome to Bioconductor
##
##   Vignettes contain introductory material; view with
##   'browseVignettes()'. To cite Bioconductor, see
##   'citation("Biobase)"', and for packages 'citation("pkgname)"'.
##
## Attaching package: 'Biobase'
## The following object is masked from 'package:MatrixGenerics':
##
##   rowMedians
## The following objects are masked from 'package:matrixStats':
##
##   anyMissing, rowMedians
library(tidyverse)

## -- Attaching packages ----- tidyverse 1.3.0 --
## v ggplot2 3.3.3    v purrr   0.3.4
## v tibble  3.1.0    v dplyr  1.0.5
## v tidyr   1.1.3    v stringr 1.4.0
## v readr   1.4.0    v forcats 0.5.1

```

```

## -- Conflicts ----- tidyverse_conflicts() --
## x dplyr::collapse() masks IRanges::collapse()
## x dplyr::combine() masks Biobase::combine(), BiocGenerics::combine()
## x dplyr::count() masks matrixStats::count()
## x dplyr::desc() masks IRanges::desc()
## x tidyr::expand() masks S4Vectors::expand()
## x dplyr::filter() masks stats::filter()
## x dplyr::first() masks S4Vectors::first()
## x dplyr::lag() masks stats::lag()
## x ggplot2::Position() masks BiocGenerics::Position(), base::Position()
## x purrr::reduce() masks GenomicRanges::reduce(), IRanges::reduce()
## x dplyr::rename() masks S4Vectors::rename()
## x dplyr::slice() masks IRanges::slice()

library(RColorBrewer)
library(pheatmap)
library(DEGreport)
library(tximport)
library(ggplot2)
library(ggrepel)
library(knitr)
library(AnnotationHub)

## Loading required package: BiocFileCache
## Loading required package: dbplyr
##
## Attaching package: 'dbplyr'
##
## The following objects are masked from 'package:dplyr':
##
##     ident, sql
##
## Attaching package: 'AnnotationHub'
##
## The following object is masked from 'package:Biobase':
##
##     cache

library(ensembladb)

## Loading required package: GenomicFeatures
## Loading required package: AnnotationDbi
##
## Attaching package: 'AnnotationDbi'
##
## The following object is masked from 'package:dplyr':
##
##     select
##
## Loading required package: AnnotationFilter
##
## Attaching package: 'ensembladb'
##
## The following object is masked from 'package:dplyr':
##

```

```

##      filter
## The following object is masked from 'package:stats':
##
##      filter
library(org.Hs.eg.db)

##
library(clusterProfiler)

##
## clusterProfiler v3.18.1 For help: https://guangchuangyu.github.io/software/clusterProfiler
##
## If you use clusterProfiler in published research, please cite:
## Guangchuang Yu, Li-Gen Wang, Yanyan Han, Qing-Yu He. clusterProfiler: an R package for comparing bio
##
## Attaching package: 'clusterProfiler'
## The following objects are masked from 'package:ensembldb':
##
##      filter, select
## The following object is masked from 'package:AnnotationDbi':
##
##      select
## The following object is masked from 'package:purrr':
##
##      simplify
## The following object is masked from 'package:IRanges':
##
##      slice
## The following object is masked from 'package:S4Vectors':
##
##      rename
## The following object is masked from 'package:stats':
##
##      filter
ggplot2::theme_set(theme_light(base_size = 14))

opts_chunk[["set"]](
  cache = FALSE,
  dev = c("png", "pdf"),
  error = TRUE,
  highlight = TRUE,
  message = FALSE,
  prompt = FALSE,
  tidy = FALSE,
  warning = FALSE)

# Have a folder called 'data', and copy your Salmon folders here from the cluster.
## List all directories containing data
### change the pattern to something specific to your Salmon folders

```

```

samples <- list.files(path = "./data/final",
                      full.names = T,
                      pattern = "^S")

## Obtain a vector of all filenames including the path
files <- file.path(samples, "salmon", "quant.sf")
files

## [1] "./data/final/S3154-2/salmon/quant.sf"
## [2] "./data/final/S3169-2/salmon/quant.sf"
## [3] "./data/final/S3188-2/salmon/quant.sf"
## [4] "./data/final/S3190-4/salmon/quant.sf"
## [5] "./data/final/S3193-1/salmon/quant.sf"
## [6] "./data/final/S3194-3/salmon/quant.sf"
## [7] "./data/final/S3220-1/salmon/quant.sf"
## [8] "./data/final/S3234-1/salmon/quant.sf"
## [9] "./data/final/S3291-3/salmon/quant.sf"
## [10] "./data/final/S3292-3/salmon/quant.sf"
## [11] "./data/final/S3372-1/salmon/quant.sf"
## [12] "./data/final/S3374-2/salmon/quant.sf"
## [13] "./data/final/S3404-1/salmon/quant.sf"
## [14] "./data/final/S3424-1/salmon/quant.sf"
## [15] "./data/final/S3474-3/salmon/quant.sf"
## [16] "./data/final/S3477-1/salmon/quant.sf"
## [17] "./data/final/S3563-3/salmon/quant.sf"
## [18] "./data/final/S3582-4/salmon/quant.sf"
## [19] "./data/final/S3644-1/salmon/quant.sf"
## [20] "./data/final/S3652-1/salmon/quant.sf"
## [21] "./data/final/S3688-2/salmon/quant.sf"
## [22] "./data/final/S3697-3/salmon/quant.sf"
## [23] "./data/final/S3713-1/salmon/quant.sf"
## [24] "./data/final/S3715-2/salmon/quant.sf"
## [25] "./data/final/S3723-1/salmon/quant.sf"
## [26] "./data/final/S3728-3/salmon/quant.sf"
## [27] "./data/final/S3732-1/salmon/quant.sf"
## [28] "./data/final/S3741-3/salmon/quant.sf"
## [29] "./data/final/S3816-1/salmon/quant.sf"
## [30] "./data/final/S3822-1/salmon/quant.sf"
## [31] "./data/final/S3825-1/salmon/quant.sf"
## [32] "./data/final/S3837-2/salmon/quant.sf"
## [33] "./data/final/S4047-1/salmon/quant.sf"
## [34] "./data/final/S4056-1/salmon/quant.sf"
## [35] "./data/final/S4089-1/salmon/quant.sf"
## [36] "./data/final/S4101-3/salmon/quant.sf"
## [37] "./data/final/S4136-1/salmon/quant.sf"
## [38] "./data/final/S4144-2/salmon/quant.sf"
## [39] "./data/final/S4172-1/salmon/quant.sf"
## [40] "./data/final/S4176-3/salmon/quant.sf"
## [41] "./data/final/S4237-1/salmon/quant.sf"
## [42] "./data/final/S4249-1/salmon/quant.sf"
## [43] "./data/final/S4261-1/salmon/quant.sf"
## [44] "./data/final/S4295-5/salmon/quant.sf"

```

```
## Since all quant files have the same name it is useful to have names for each element
### change the string in str_replace so the pattern matches your filenames
names(files) <- str_replace(samples, "./data/final/", "")
```

```
# Load the data and metadata
```

```
meta <- read_csv("tables/metadata_corrected.csv") %>%
  column_to_rownames(var = "samplename") %>%
  dplyr::filter(treatment == "post") %>%
  drop_na(response)
protein_coding_genes <- read_csv("tables/ensembl_w_description.protein_coding.csv")
```

```
# Connect to AnnotationHub
```

```
ah <- AnnotationHub()
```

```
# Query AnnotationHub
```

```
hs_ens <- query(ah, c("Homo sapiens", "EnsDb"))
```

```
# Get Ensembl94 - used in bcbio
```

```
hs_ens <- hs_ens[["AH64923"]]
```

```
# Extract gene-level information
```

```
txdb <- transcripts(hs_ens,
  return.type = "data.frame") %>%
  dplyr::select(tx_id, gene_id)
```

```
genedb <- genes(hs_ens,
  return.type = "data.frame") %>%
  dplyr::select(gene_id, gene_name, symbol)
```

```
gene_symbol <- genedb %>% dplyr::select(gene_id, symbol)
```

```
hsdb <- inner_join(txdb, genedb)
```

```
write.table(hsdb,
  file = "data/ensembl94_hg38_annotations.txt",
  sep = "\t",
  row.names = F,
  quote = F)
```

```
# Read in a tx2gene file with transcript identifiers in the first column and gene identifiers in the s
```

```
#wormdb <- read.table("ensembl94_WBcel235_annotations.txt", sep="\t", header=T)
```

```
tx2gene <- hsdb[, c("tx_id", "gene_id")]
```

```
# Run tximport
```

```
files <- files[rownames(meta)]
txi_file <- "data/txi.day8.RDS"
if (file.exists(txi_file)){
  txi <- readRDS(txi_file)
}else{
  txi <- tximport(files,
    type = "salmon",
    tx2gene = tx2gene,
    countsFromAbundance = "lengthScaledTPM",
    ignoreTxVersion = FALSE)
```

```

    saveRDS(txi, txi_file)
}

# Look at the counts
class(txi)

## [1] "list"

attributes(txi)

## $names
## [1] "abundance"          "counts"              "length"
## [4] "countsFromAbundance"

txi$counts %>% View()

```

Checking to see that the transcript to gene mapping is correct

When you have annotations that are from a different source from your reference you can run into problems (i.e. lose genes). Some checks you can do before proceeding:

1. Look at the dimensions of your count matrix. Do you have ~20k genes present? `dim(txi$counts)`
2. When running `tximport()` you will get a message in your console. If you see something like `transcripts missing from tx2gene` start troubleshooting.

```
dim(txi$counts)
```

```
## [1] 58735    20
```

Sanity check that metadata matches your expression

It is always a good idea to check if:

1. Do you have expression data for all samples listed in your metadata?
2. Are the samples in your expression data in the same order as your metadata?

```

### Check that sample names match in both files
all(colnames(txi$counts) %in% rownames(meta))

## [1] TRUE

# Not the same? Make them the same
### This will change depending on what names you have listed!
#paste0(meta$samplename, "_", meta$library)
#rownames(meta) <- paste0(meta$samplename, "_", meta$library)
#meta$genotype <- releval(meta$genotype, ref="Wildtype")

### Check that sample names match in both files
all(colnames(txi$counts) %in% rownames(meta))

## [1] TRUE

### Check that all samples are in the same order
meta <- meta[colnames(txi$counts),]
all(colnames(txi$counts) == rownames(meta))

## [1] TRUE

```


Run DESeq2

estimating size factors
estimating dispersions
gene-wise dispersion estimates
mean-dispersion relationship
final dispersion estimates
fitting model and testing

- Estimating size factors and count normalization
- Gene-wise dispersions
- Mean-dispersion(variance) relationship and the Negative Binomial Model
- Model fitting and hypothesis testing

Wald test

Here we subset protein coding genes.

```
## Create DESeq2Dataset object
dds_file <- "data/dds.day8.RDS"
meta$treatment <- as.factor(meta$treatment)
meta$response <- as.factor(meta$response)
meta$er <- as.factor(meta$er)
meta$date_of <- as.factor(meta$date_of)
meta$tumor_percentage <- as.factor(meta$tumor_percentage)
meta$tumor_percentage_high <- as.factor(meta$tumor_percentage_high)

non_responders <- meta %>% dplyr::filter(study_id %in% c(2, 19)) %>% row.names()

if (file.exists(dds_file)){
  dds <- readRDS(dds_file)
}else{
  dds <- DESeqDataSetFromTximport(txi,
                                colData = meta,
                                design = ~response)

  #dds <- dds[,!colnames(dds) %in% non_responders]
  design(dds) <- formula(~response + er + tumor_percentage_high + date_of)

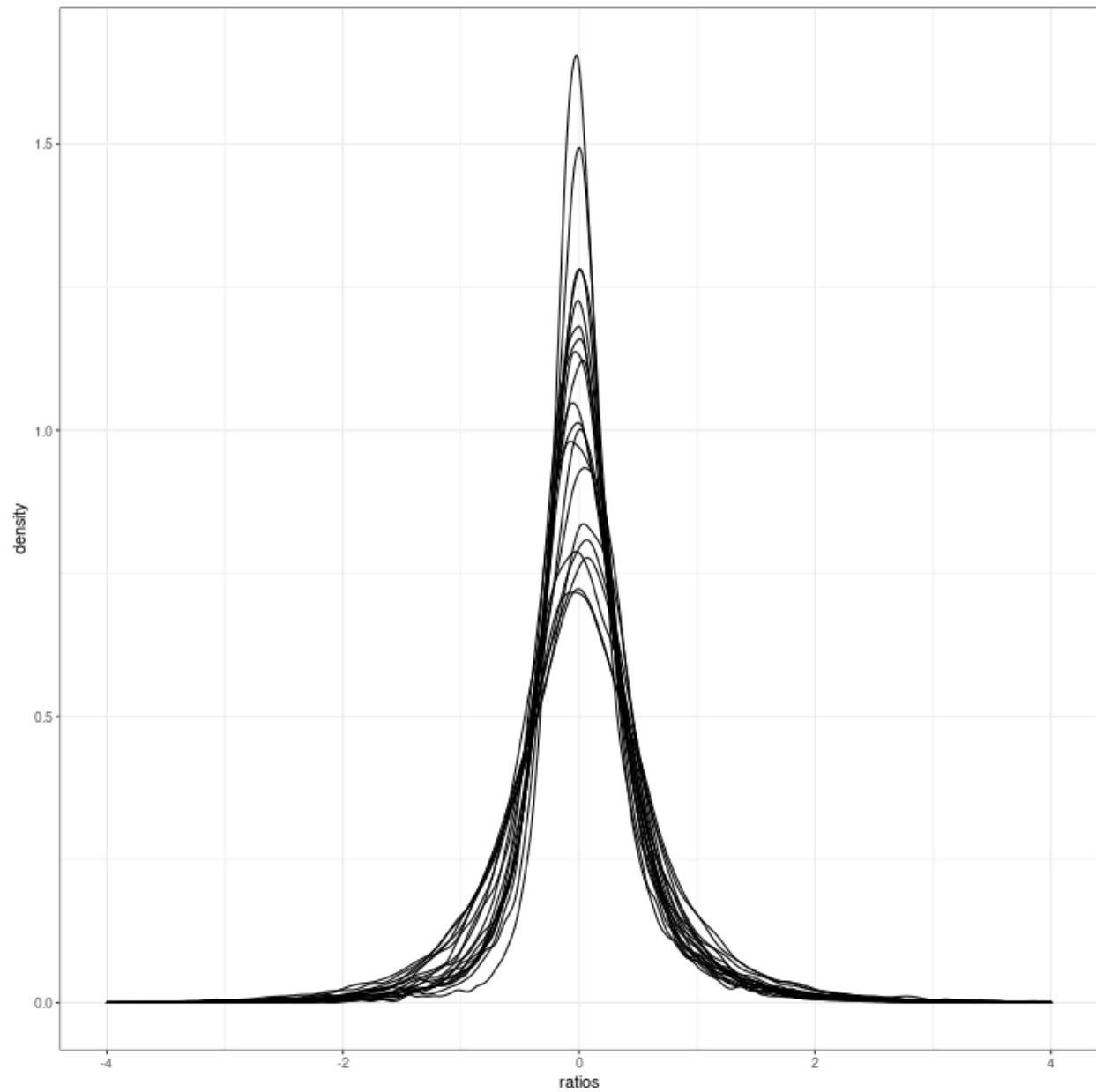
  # subset protein-coding genes
  pc_genes <- intersect(protein_coding_genes$ensembl_gene_id, row.names(dds))
  dds <- dds[pc_genes,]
  # 100 reads / 20 samples
  keep <- rowSums(counts(dds)) >= 100
  dds <- dds[keep,]

  # Run DESeq2
  dds <- DESeq(dds)
  saveRDS(dds, dds_file)
}
```

DEGreport QC

Size factor QC - samples 1-20

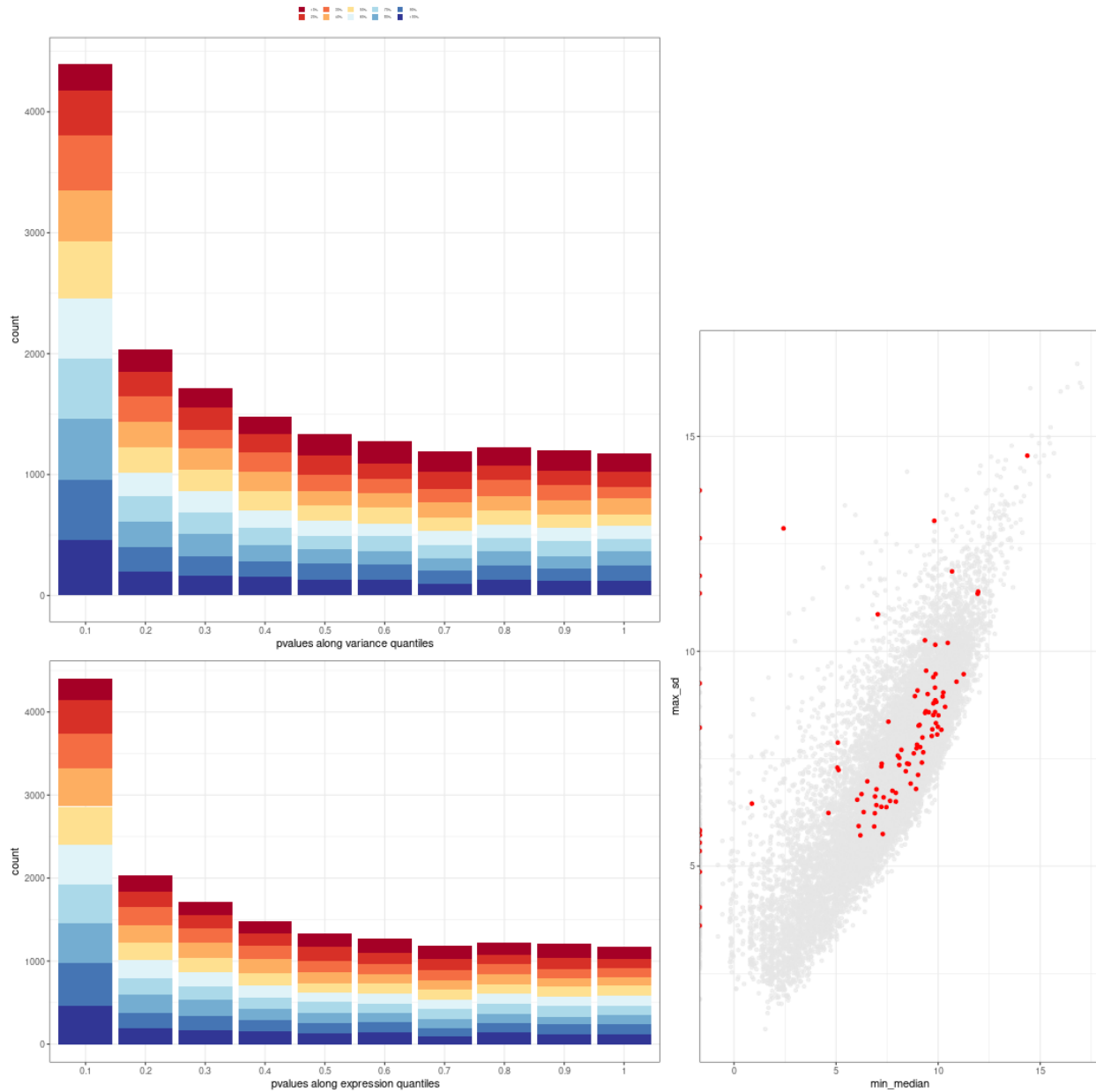
```
counts <- counts(dds, normalized = TRUE)
design <- as.data.frame(colData(dds))
degCheckFactors(counts[, 1:20])
```



Mean-Variance QC plots

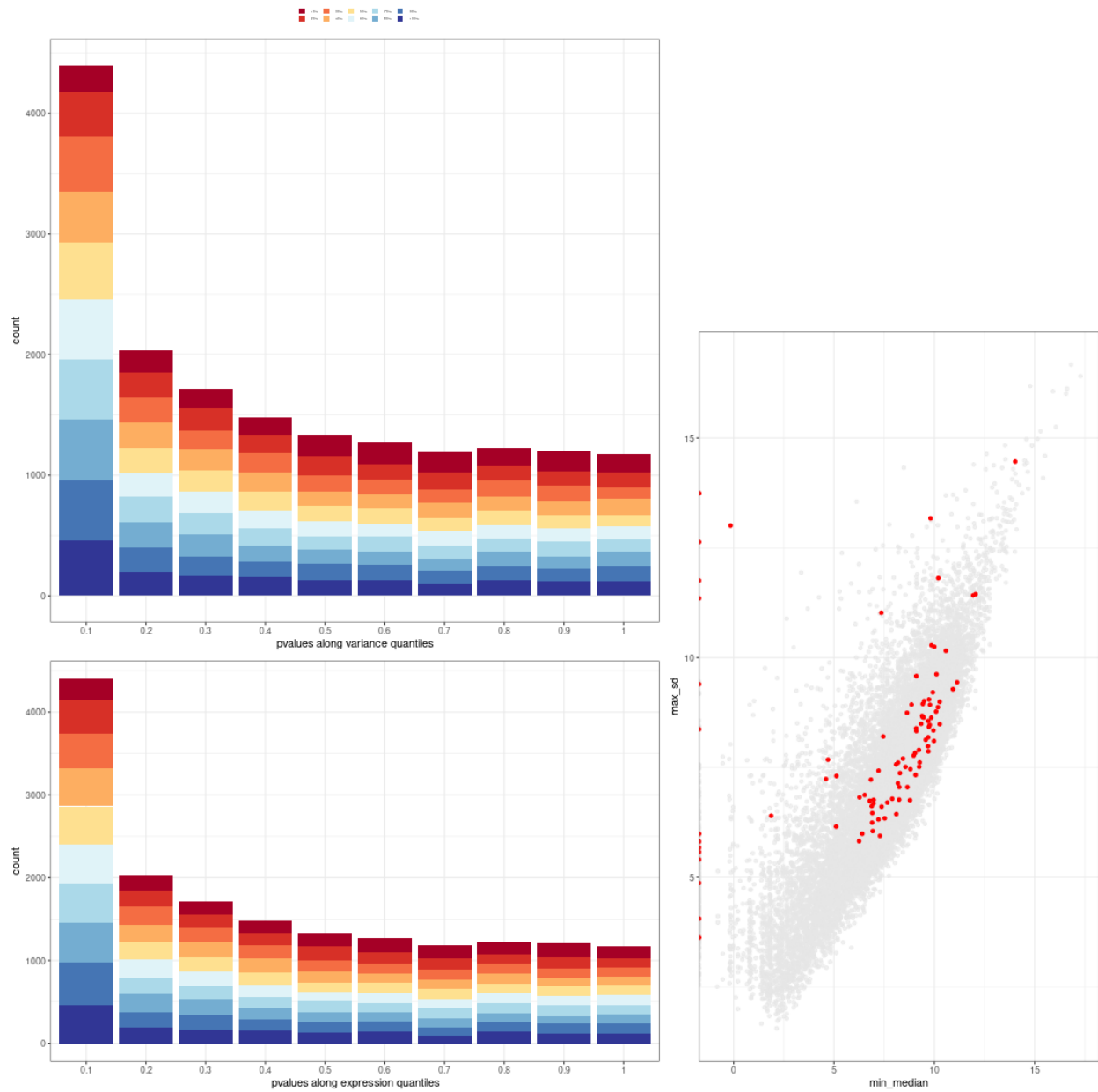
response

```
res <- results(dds)
degQC(counts, design[["response"]], pvalue = res[["pvalue"]])
```



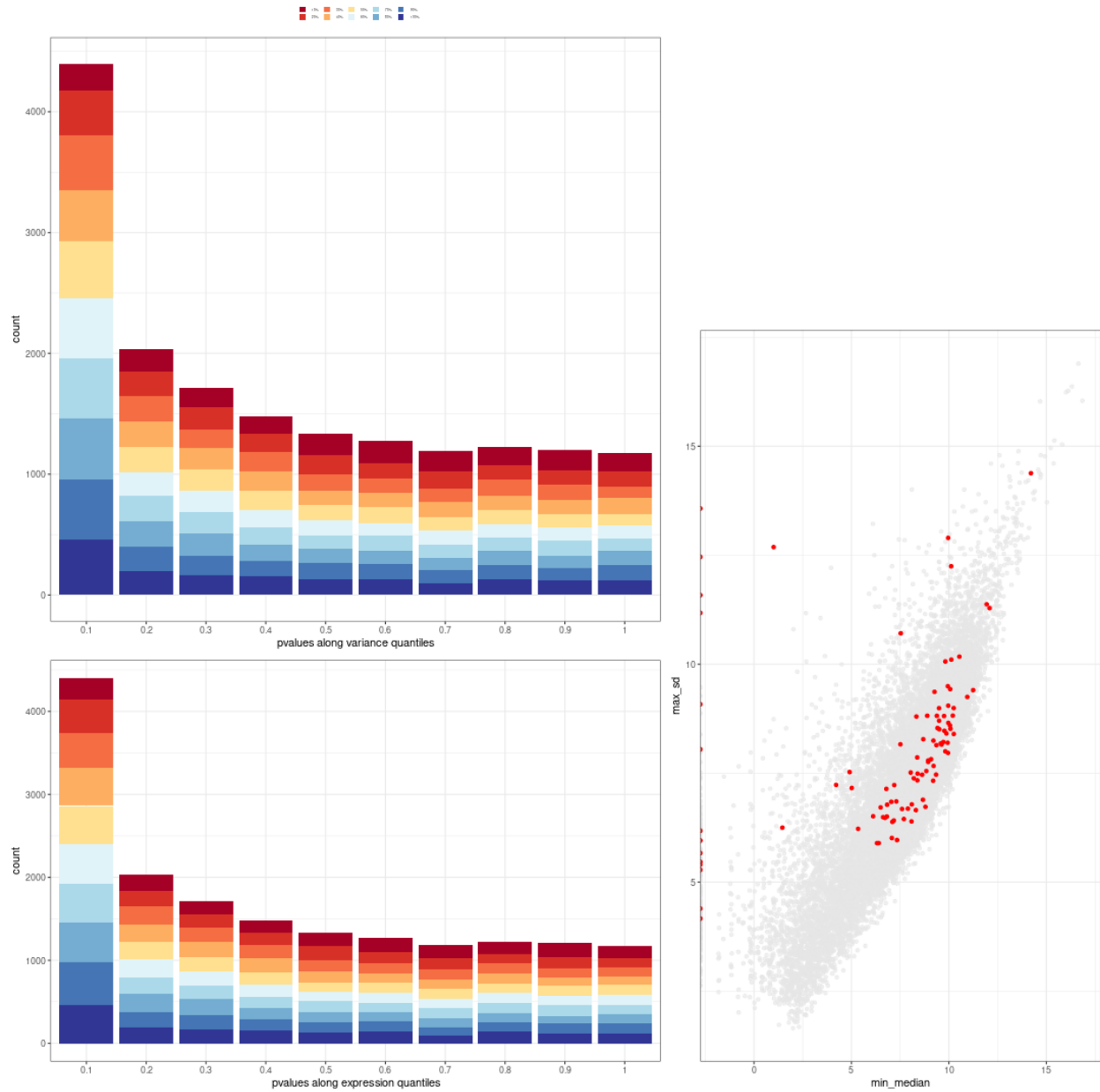
ER

```
degQC(counts, design[["er"]], pvalue = res[["pvalue"]])
```



tumor_percentage_high

```
degQC(counts, design[["tumor_percentage_high"]], pvalue = res[["pvalue"]])
```



Covariates effect on count data

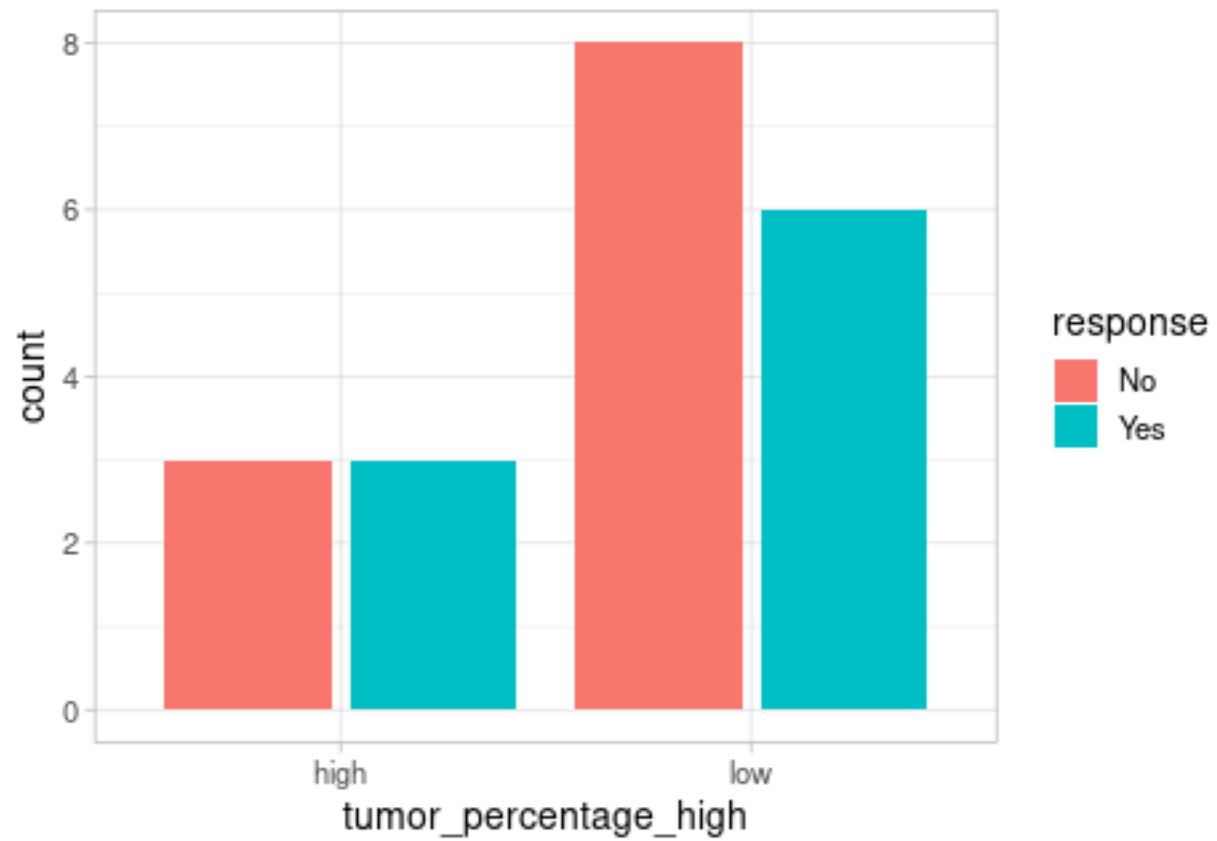
```

mdata <- colData(dds) %>% as.data.frame() %>%
  dplyr::select(response, er, date_of, tumor_percentage_high)

#resCov <- degCovariates(log2(counts(dds)+0.5), mdata)

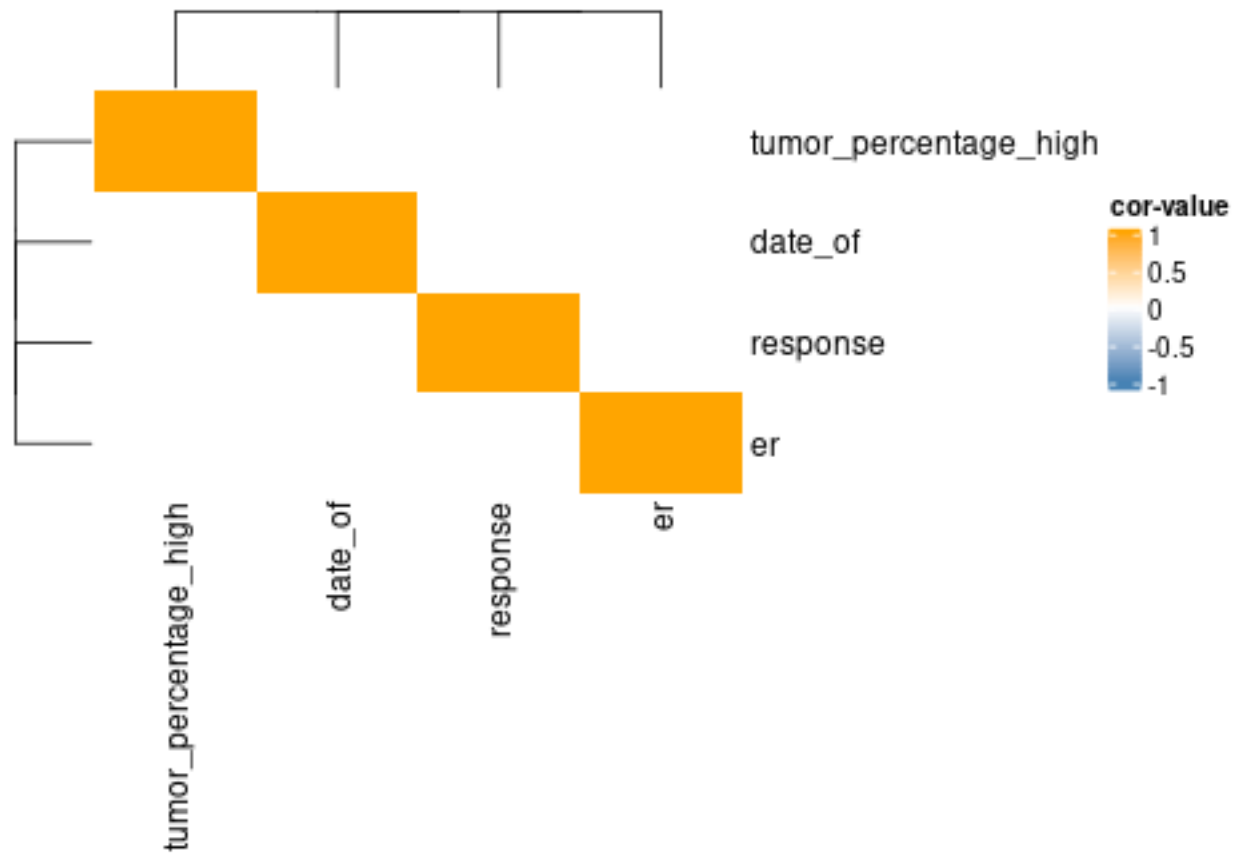
mdata %>% ggplot(aes(tumor_percentage_high, fill = response)) + geom_bar(position = "dodge2")

```

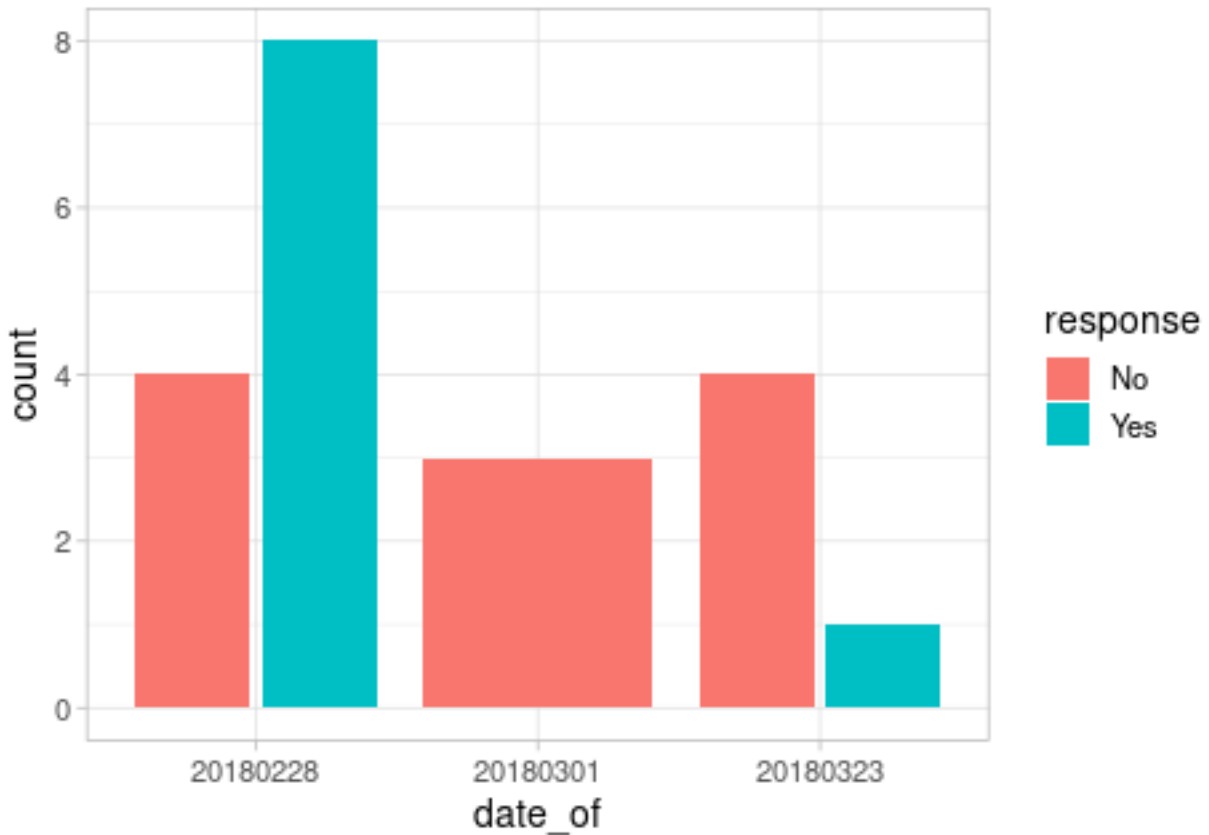


Covariates correlation with metrics

```
cor <- degCorCov(mdata)
```



```
mdata %>% ggplot(aes(date_of, fill = response)) + geom_bar(position = "dodge2")
```



Sample-level QC analysis

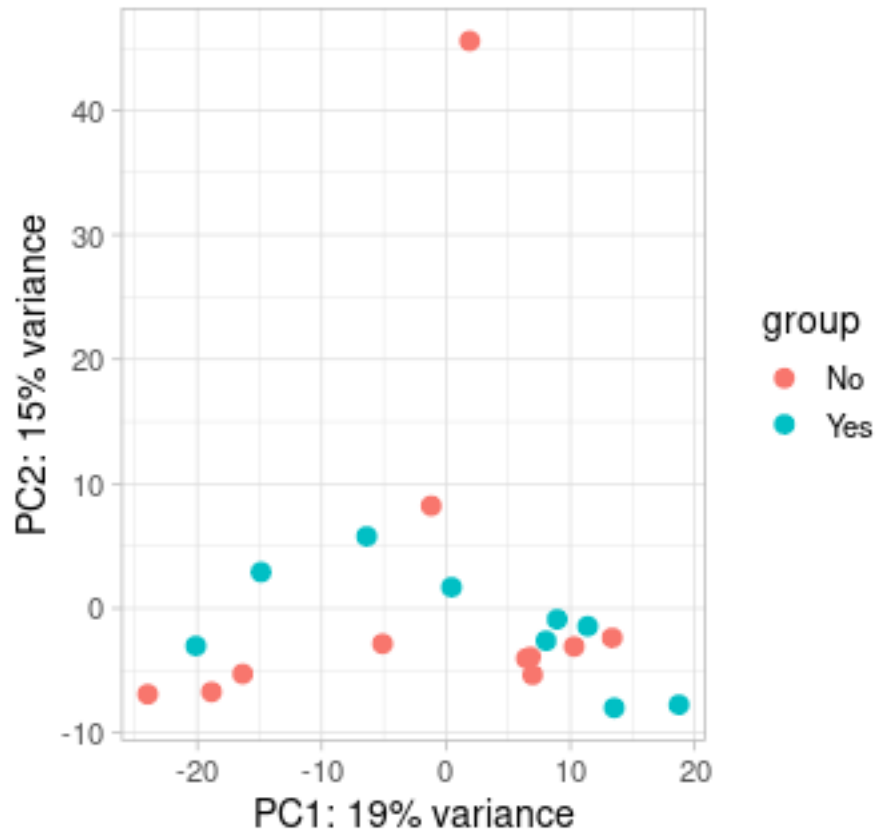
```
### Transform counts for data visualization (unsupervised analysis)
rld_file <- "data/rld.day8.RDS"
if (file.exists(rld_file)){
  rld <- readRDS(rld_file)
}else{
  rld <- rlog(dds, blind = TRUE)
  saveRDS(rld, rld_file)
}
class(rld) # what type of object is this
```

```
## [1] "DESeqTransform"
## attr(,"package")
## [1] "DESeq2"
```

```
# we also need just a matrix of transformed counts
rld_mat <- assay(rld)
```

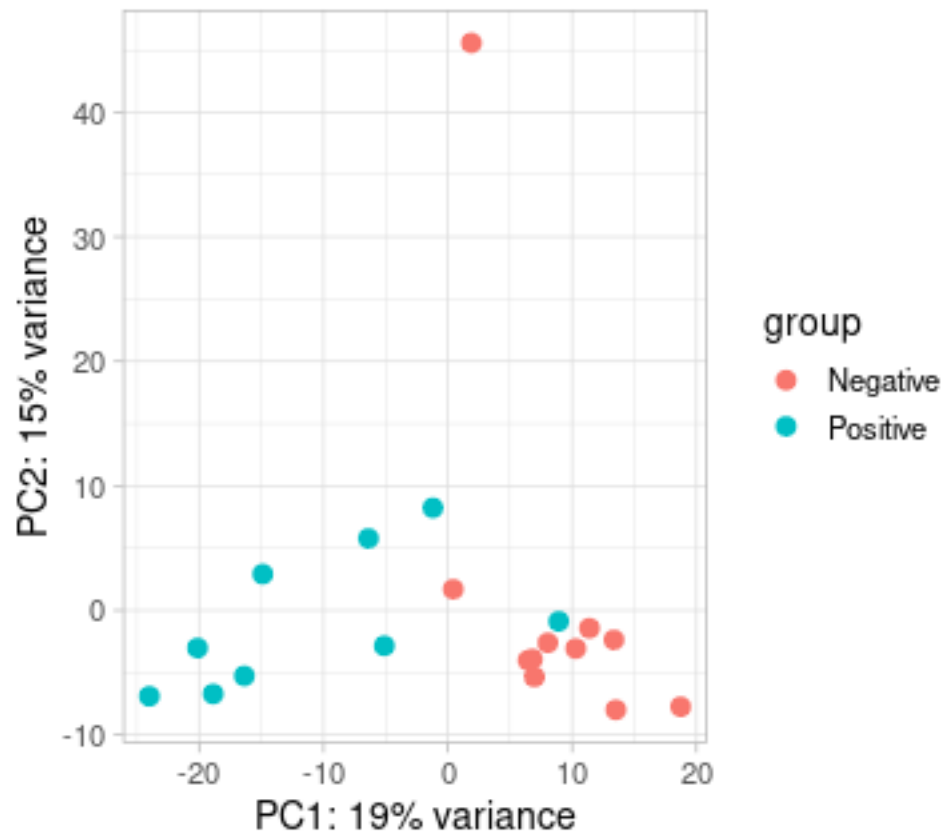
PCA - response

```
# Use the DESeq2 function
plotPCA(rld, intgroup = c("response"))
```

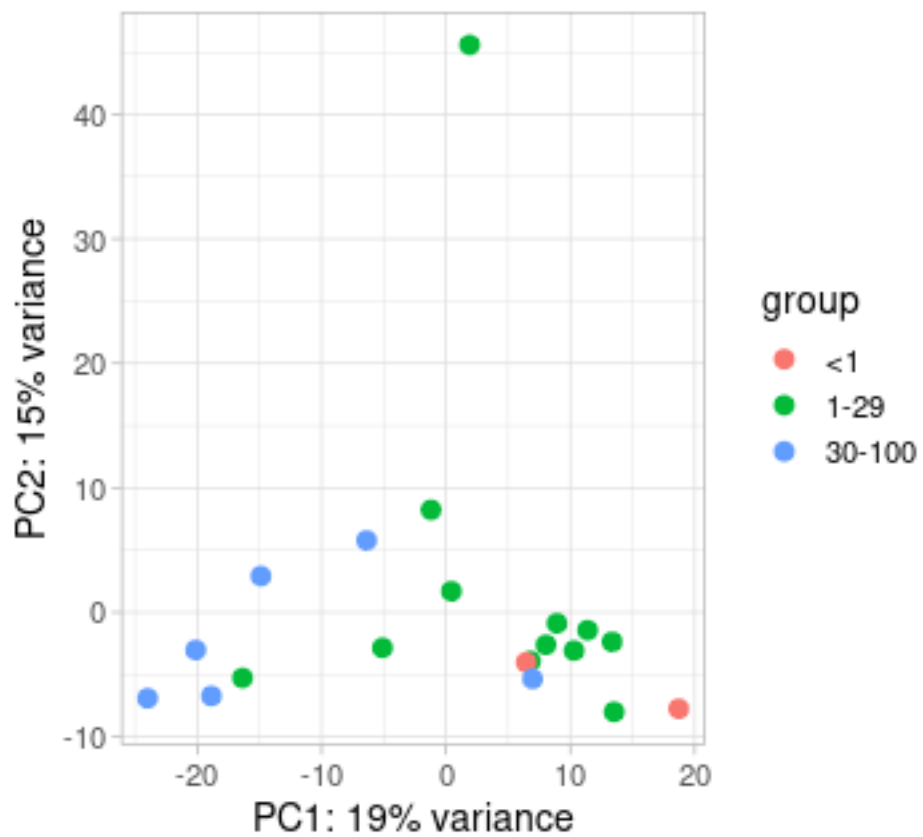
PCA - ER

```
# Use the DESeq2 function  
plotPCA(rld, intgroup = c("er"))
```



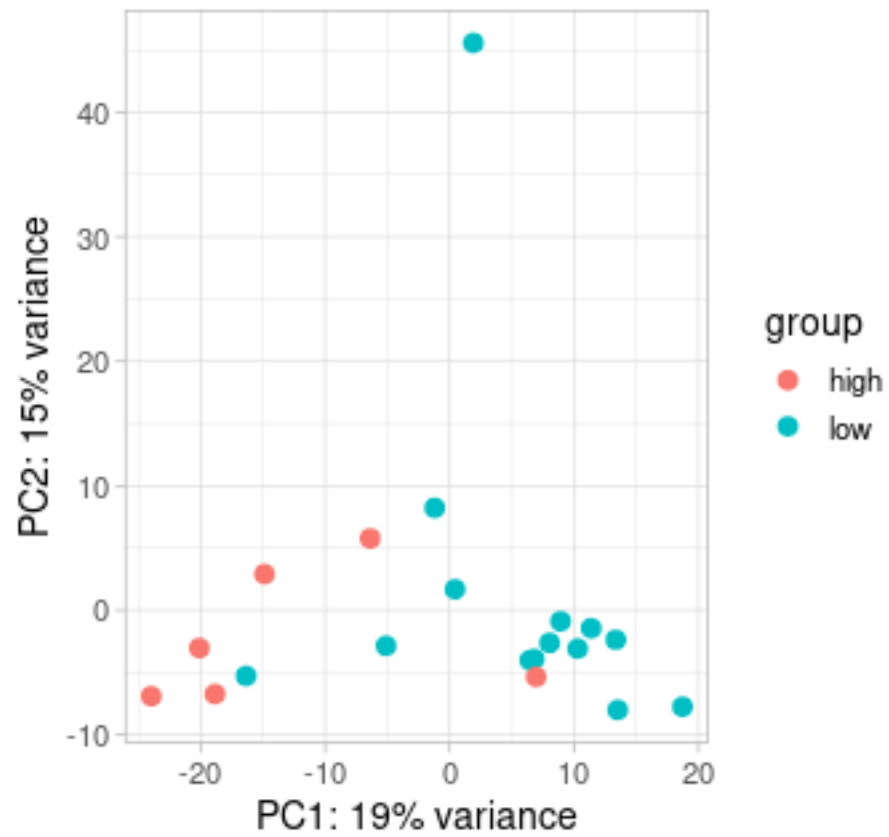
PCA - tumor_percentage

```
# Use the DESeq2 function  
plotPCA(rld, intgroup = c("tumor_percentage"))
```



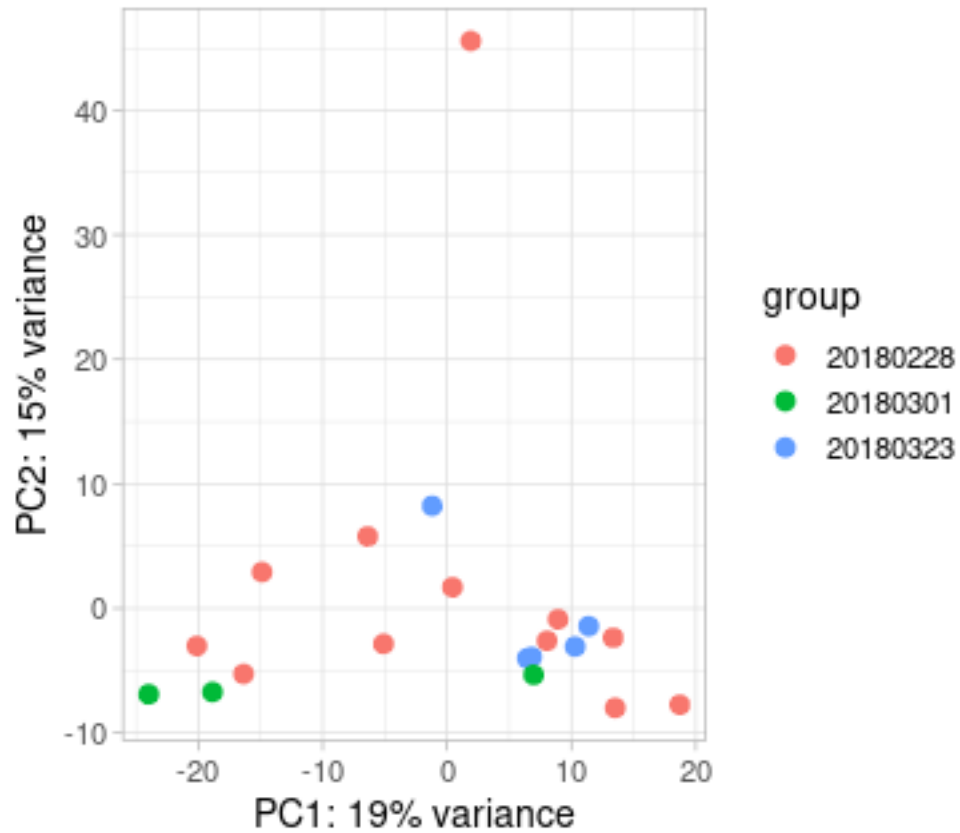
PCA - tumor_percentage_high

```
# Use the DESeq2 function  
plotPCA(rld, intgroup = c("tumor_percentage_high"))
```



PCA - date_of

```
# Use the DESeq2 function  
plotPCA(rld, intgroup = c("date_of"))
```



Inter-correlation analysis

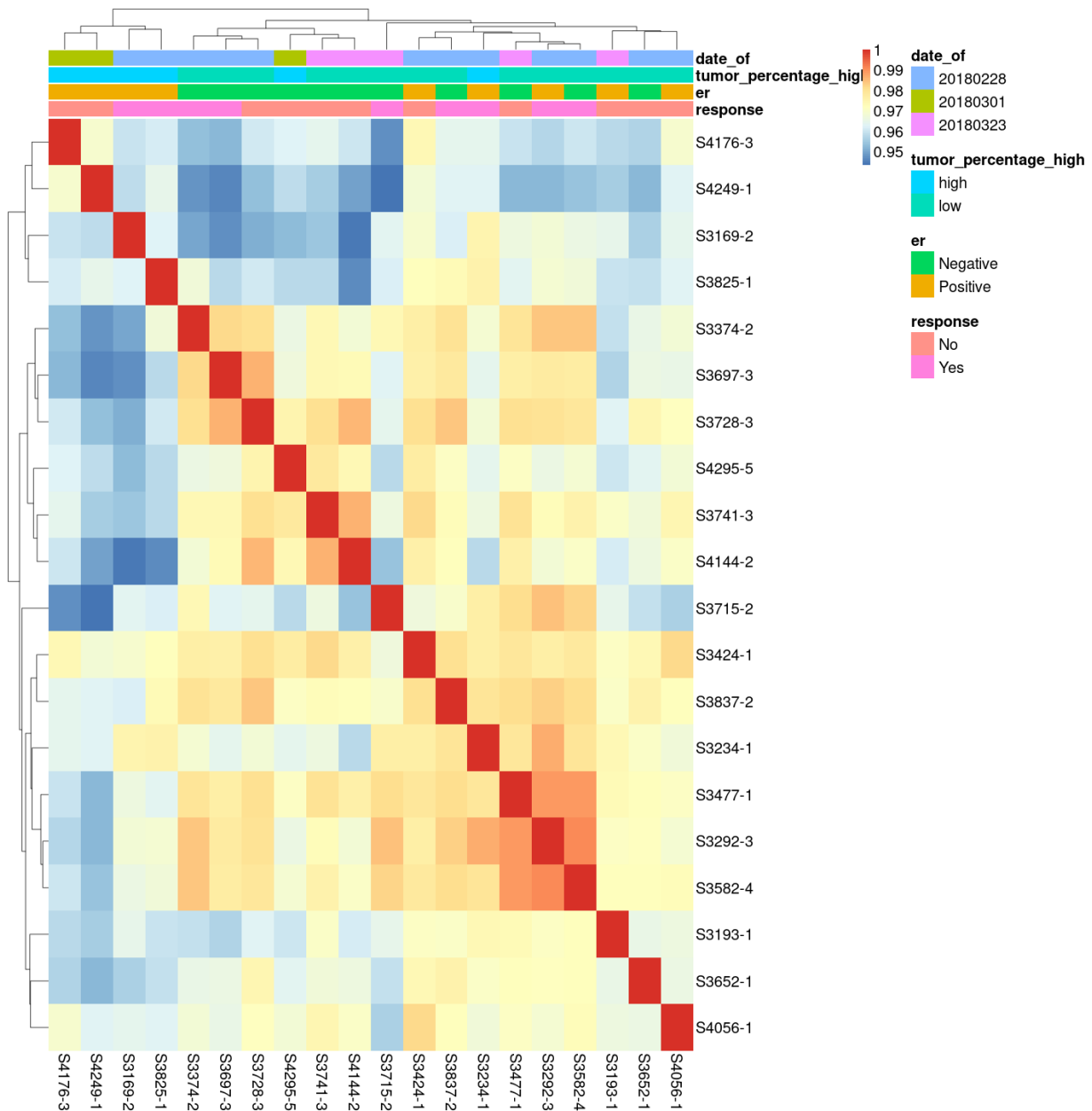
Without study_id

```
# Correlation matrix
rld_cor <- cor(rld_mat)

meta$study_id <- as.factor(meta$study_id)
# Create annotation file for samples
annotation <- meta[, c("response", "er", "tumor_percentage_high", "date_of")]

# Change colors
heat.colors <- brewer.pal(6, "Blues")

# Plot heatmap
pheatmap(rld_cor,
  annotation = annotation,
  border = NA,
  fontsize = 20)
```



With study_id

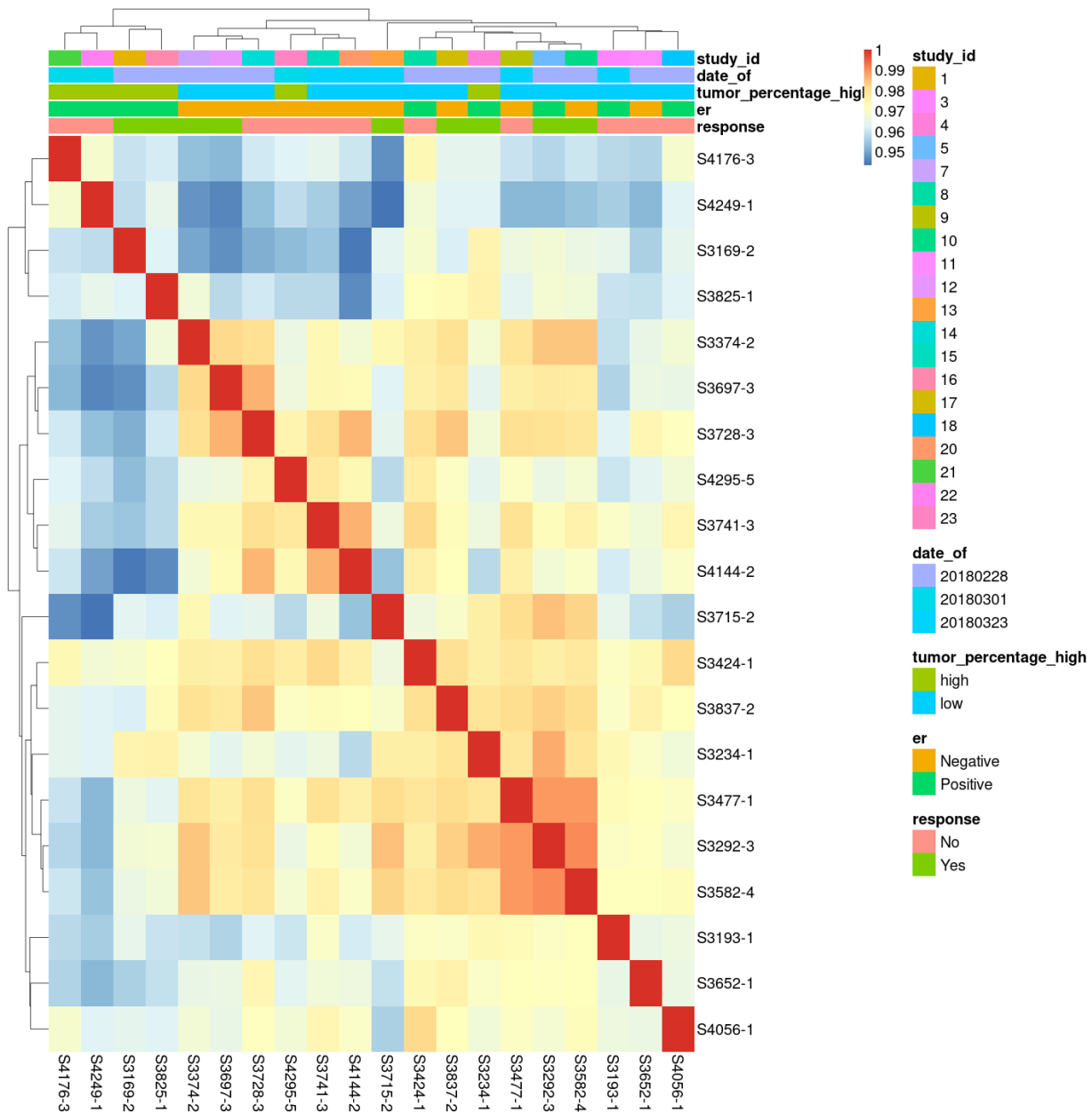
```
# Correlation matrix
rld_cor <- cor(rld_mat)

meta$study_id <- as.factor(meta$study_id)
# Create annotation file for samples
annotation <- meta[, c("response", "er", "tumor_percentage_high", "date_of", "study_id")]

# Change colors
heat.colors <- brewer.pal(6, "Blues")

# Plot heatmap
```

```
pheatmap(rld_cor,
  annotation = annotation,
  border = NA,
  fontsize = 20)
```



Response Yes vs No for Day 8 - see Table13

```
# Get results for rescue vs wt
contrast <- c("response", "Yes", "No")
resResponse <- results(dds, contrast = contrast, alpha = 0.05)
length(which(resResponse$padj < 0.05))
```

```
## [1] 818
```

```

# Add annotations
resResponse_tb <- resResponse %>%
  data.frame() %>%
  rownames_to_column(var = "gene") %>%
  as_tibble() %>%
  left_join(gene_symbol, by = c("gene" = "gene_id"))

resResponse_tb_significant <- dplyr::filter(resResponse_tb, padj < 0.05)

samples_no <- meta %>% dplyr::filter(response == "No") %>% row.names()

counts_no <- txi$abundance %>%
  as.data.frame() %>%
  dplyr::select(any_of(samples_no)) %>%
  rowSums() %>%
  as.data.frame() %>%
  rownames_to_column(var = "ensembl_gene_id")
colnames(counts_no) <- c("ensembl_gene_id", "no_expression_mean_tpm")

samples_yes <- meta %>% dplyr::filter(response == "Yes") %>% row.names()

counts_yes <- txi$abundance %>%
  as.data.frame() %>%
  dplyr::select(any_of(samples_yes)) %>%
  rowSums() %>%
  as.data.frame() %>%
  rownames_to_column(var = "ensembl_gene_id")

colnames(counts_yes) <- c("ensembl_gene_id", "yes_expression_mean_tpm")

counts_yes <- counts_yes %>%
  left_join(counts_no,
    by = c("ensembl_gene_id" = "ensembl_gene_id"))

resResponse_tb_significant <- resResponse_tb_significant %>%
  left_join(counts_yes, by = c("gene" = "ensembl_gene_id"))

write_csv(resResponse_tb_significant,
  "tables/T13.DE_response_day8.csv")

# Separate into up and down-regulated gene sets
sigResponse_up <- rownames(resResponse)[which(resResponse$padj < 0.01 & resResponse$log2FoldChange > 0)]
sigResponse_down <- rownames(resResponse)[which(resResponse$padj < 0.01 & resResponse$log2FoldChange < 0)]

```

ER : Positive vs Negative for Day8 - Table 14

```

contrast <- c("er", "Positive", "Negative")
resER <- results(dds, contrast = contrast, alpha = 0.05)
length(which(resER$padj < 0.05))

```

```
## [1] 182
```



```

# Add annotations
resER_tb <- resER %>%
  data.frame() %>%
  rownames_to_column(var = "gene") %>%
  as_tibble() %>%
  left_join(gene_symbol, by = c("gene" = "gene_id"))

resER_tb_significant <- dplyr::filter(resER_tb, padj < 0.05)

samples_pos <- meta %>% dplyr::filter(er == "Positive") %>% row.names()

counts_pos <- txi$abundance %>%
  as.data.frame() %>%
  dplyr::select(any_of(samples_pos)) %>%
  rowMeans() %>%
  as.data.frame() %>%
  rownames_to_column(var = "ensembl_gene_id")
colnames(counts_pos) <- c("ensembl_gene_id", "Positive_expression_mean_tpm")

samples_neg <- meta %>% dplyr::filter(er == "Negative") %>% row.names()

counts_neg <- txi$abundance %>%
  as.data.frame() %>%
  dplyr::select(any_of(samples_neg)) %>%
  rowMeans() %>%
  as.data.frame() %>%
  rownames_to_column(var = "ensembl_gene_id")

colnames(counts_neg) <- c("ensembl_gene_id", "Negative_expression_mean_tpm")

counts_pos <- counts_pos %>%
  left_join(counts_neg,
    by = c("ensembl_gene_id" = "ensembl_gene_id"))

resER_tb_significant <- resER_tb_significant %>%
  left_join(counts_pos, by = c("gene" = "ensembl_gene_id"))

write_csv(resER_tb_significant,
  "tables/T14.DE_ER.day8.csv")

# Separate into up and down-regulated gene sets
sigER_up <- rownames(resER)[which(resER$padj < 0.01 & resER$log2FoldChange > 0)]
sigER_down <- rownames(resER)[which(resER$padj < 0.01 & resER$log2FoldChange < 0)]

```

tumor_percentage_high : High vs Low for Day8- Table 15

```

contrast <- c("tumor_percentage_high", "high", "low")
resTP <- results(dds, contrast = contrast, alpha = 0.05)
length(which(resTP$padj < 0.05))

```

```
## [1] 128
```

```

# Add annotations
resTP_tb <- resTP %>%
  data.frame() %>%
  rownames_to_column(var = "gene") %>%
  as_tibble() %>%
  left_join(gene_symbol, by = c("gene" = "gene_id"))

resTP_tb_significant <- dplyr::filter(resTP_tb, padj < 0.05)

samples_high <- meta %>% dplyr::filter(tumor_percentage_high == "high") %>% row.names()

counts_high <- txi$abundance %>%
  as.data.frame() %>%
  dplyr::select(any_of(samples_high)) %>%
  rowMeans() %>%
  as.data.frame() %>%
  rownames_to_column(var = "ensembl_gene_id")
colnames(counts_high) <- c("ensembl_gene_id", "High_expression_mean_tpm")

samples_low <- meta %>% dplyr::filter(tumor_percentage_high == "low") %>% row.names()

counts_low <- txi$abundance %>%
  as.data.frame() %>%
  dplyr::select(any_of(samples_low)) %>%
  rowMeans() %>%
  as.data.frame() %>%
  rownames_to_column(var = "ensembl_gene_id")

colnames(counts_low) <- c("ensembl_gene_id", "Low_expression_mean_tpm")

counts_high <- counts_high %>%
  left_join(counts_low,
    by = c("ensembl_gene_id" = "ensembl_gene_id"))

resTP_tb_significant <- resTP_tb_significant %>%
  left_join(counts_high, by = c("gene" = "ensembl_gene_id"))

write_csv(resTP_tb_significant,
  "tables/T15.DE_tumor_percentage_high.day8.csv")

# Separate into up and down-regulated gene sets
sigTP_up <- rownames(resTP)[which(resTP$padj < 0.01 & resTP$log2FoldChange > 0)]
sigTP_down <- rownames(resTP)[which(resTP$padj < 0.01 & resTP$log2FoldChange < 0)]

```

date_of: 20180323 vs 20180228 - for Day8: Table 16

```

contrast <- c("date_of", "20180323", "20180228")
resD0 <- results(dds, contrast = contrast, alpha = 0.05)
length(which(resD0$padj < 0.05))

```

```
## [1] 408
```

```

# Add annotations
resD0_tb <- resD0 %>%
  data.frame() %>%
  rownames_to_column(var = "gene") %>%
  as_tibble() %>%
  left_join(gene_symbol, by = c("gene" = "gene_id"))

resD0_tb_significant <- dplyr::filter(resD0_tb, padj < 0.05)

samples_23 <- meta %>% dplyr::filter(date_of == "20180323") %>% row.names()

counts_23 <- txi$abundance %>%
  as.data.frame() %>%
  dplyr::select(any_of(samples_23)) %>%
  rowMeans() %>%
  as.data.frame() %>%
  rownames_to_column(var = "ensembl_gene_id")
colnames(counts_23) <- c("ensembl_gene_id", "20180323_expression_mean_tpm")

samples_28 <- meta %>% dplyr::filter(date_of == "20180228") %>% row.names()

counts_28 <- txi$abundance %>%
  as.data.frame() %>%
  dplyr::select(any_of(samples_28)) %>%
  rowMeans() %>%
  as.data.frame() %>%
  rownames_to_column(var = "ensembl_gene_id")

colnames(counts_28) <- c("ensembl_gene_id", "20180228_expression_mean_tpm")

counts_23 <- counts_23 %>%
  left_join(counts_28,
    by = c("ensembl_gene_id" = "ensembl_gene_id"))

resD0_tb_significant <- resD0_tb_significant %>%
  left_join(counts_23, by = c("gene" = "ensembl_gene_id"))

write_csv(resD0_tb_significant,
  "tables/T16.DE_date_of.day8.csv")

# Separate into up and down-regulated gene sets
sigD0_up <- rownames(resD0)[which(resD0$padj < 0.01 & resD0$log2FoldChange > 0)]
sigD0_down <- rownames(resD0)[which(resD0$padj < 0.01 & resD0$log2FoldChange < 0)]

```

Visualization

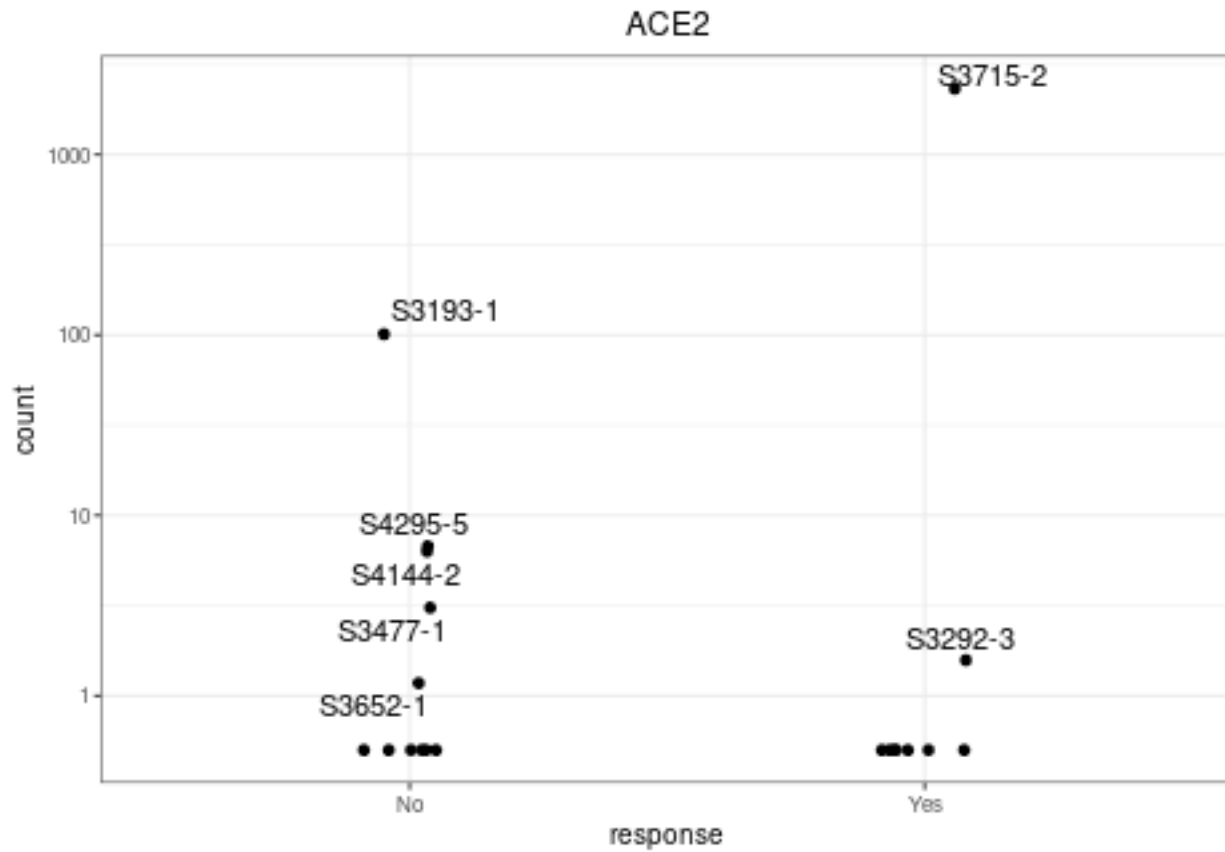
Gene example

```

d <- plotCounts(dds,
  gene = "ENSG00000130234",
  intgroup = "response",
  returnData = TRUE)

```

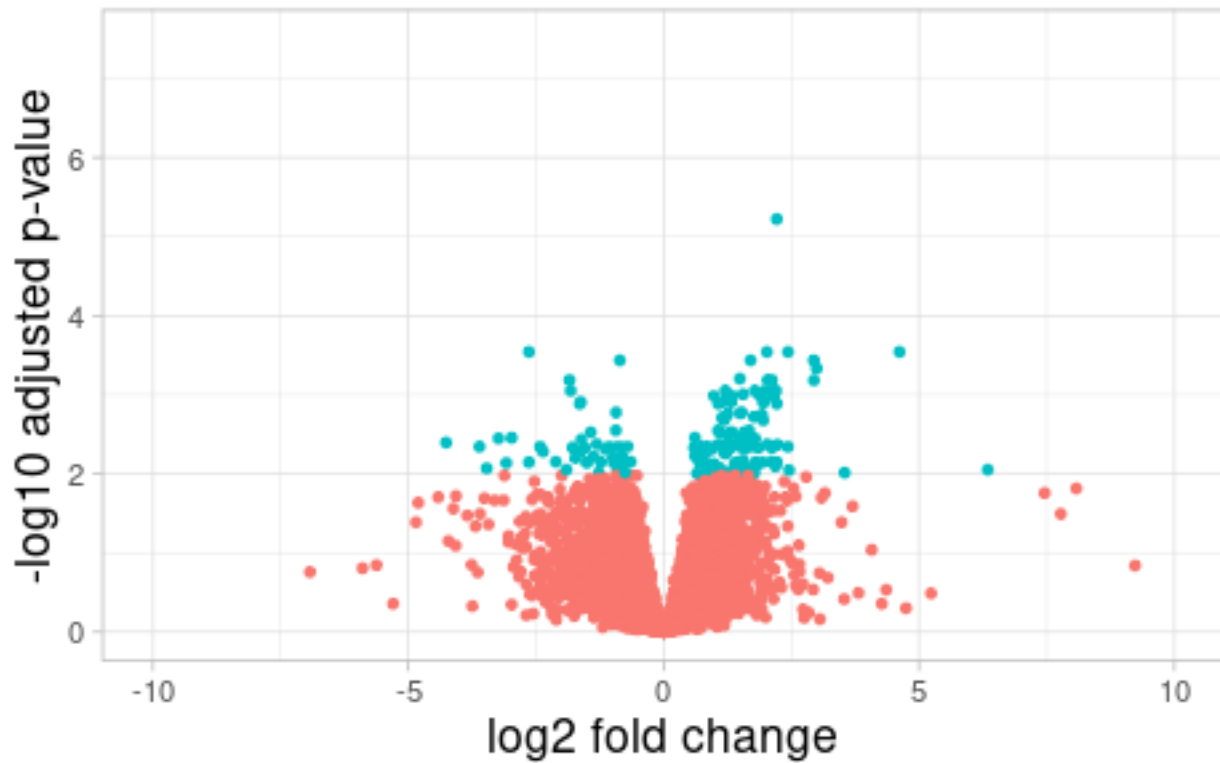
```
ggplot(d, aes(x = response, y = count)) +
  geom_point(position = position_jitter(w = 0.1, h = 0)) +
  geom_text_repel(aes(label = rownames(d))) +
  theme_bw(base_size = 10) +
  ggtitle("ACE2") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_y_log10()
```



```
# Add a column for significant genes
resResponse_tb <- resResponse_tb %>% mutate(threshold = padj < 0.01)

ggplot(resResponse_tb) +
  geom_point(aes(x = log2FoldChange, y = -log10(padj), colour = threshold)) +
  ggtitle("Response Yes vs No") +
  xlab("log2 fold change") +
  ylab("-log10 adjusted p-value") +
  scale_x_continuous(limits = c(-10,10)) +
  scale_y_continuous(limits = c(0, 7.5))+
  theme(legend.position = "none",
        plot.title = element_text(size = rel(1.5), hjust = 0.5),
        axis.title = element_text(size = rel(1.25)))
```

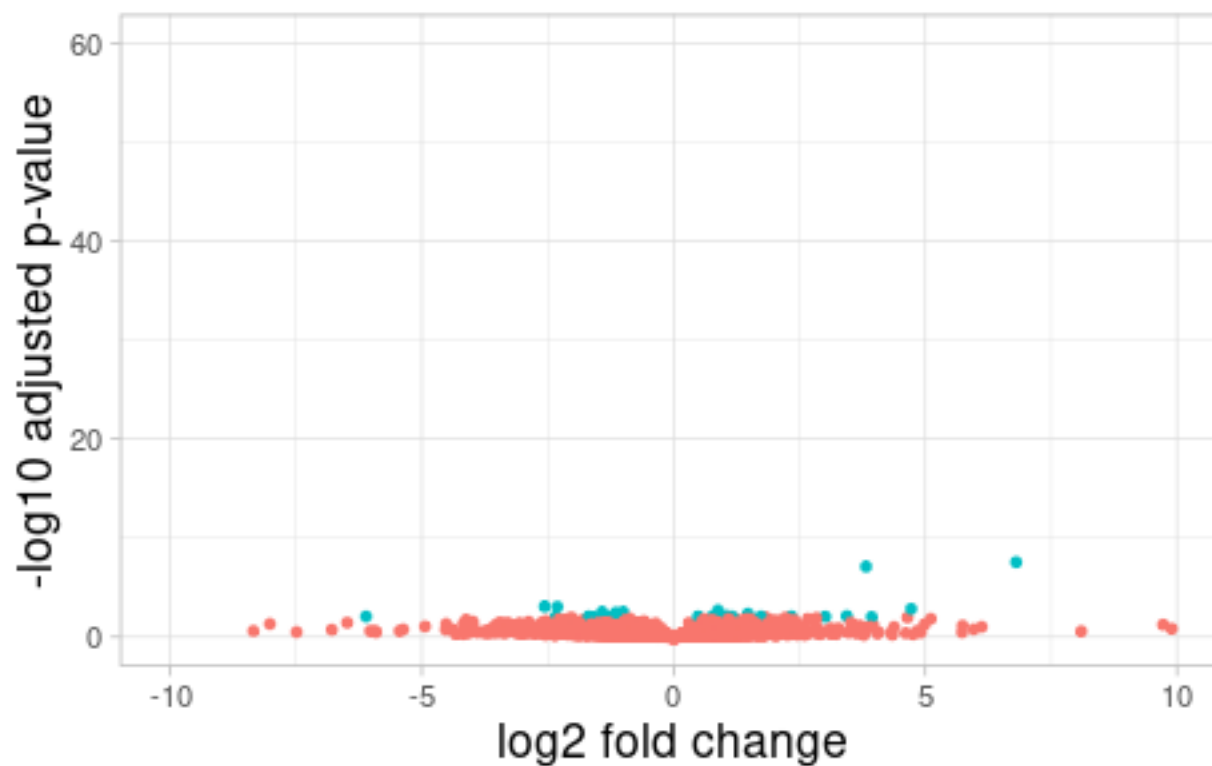
Response Yes vs No



```
# Add a column for significant genes
resER_tb <- resER_tb %>% mutate(threshold = padj < 0.01)

ggplot(resER_tb) +
  geom_point(aes(x = log2FoldChange, y = -log10(padj), colour = threshold)) +
  ggtitle("ER: Positive vs Negative") +
  xlab("log2 fold change") +
  ylab("-log10 adjusted p-value") +
  scale_x_continuous(limits = c(-10,10)) +
  theme(legend.position = "none",
        plot.title = element_text(size = rel(1.5), hjust = 0.5),
        axis.title = element_text(size = rel(1.25)))
```

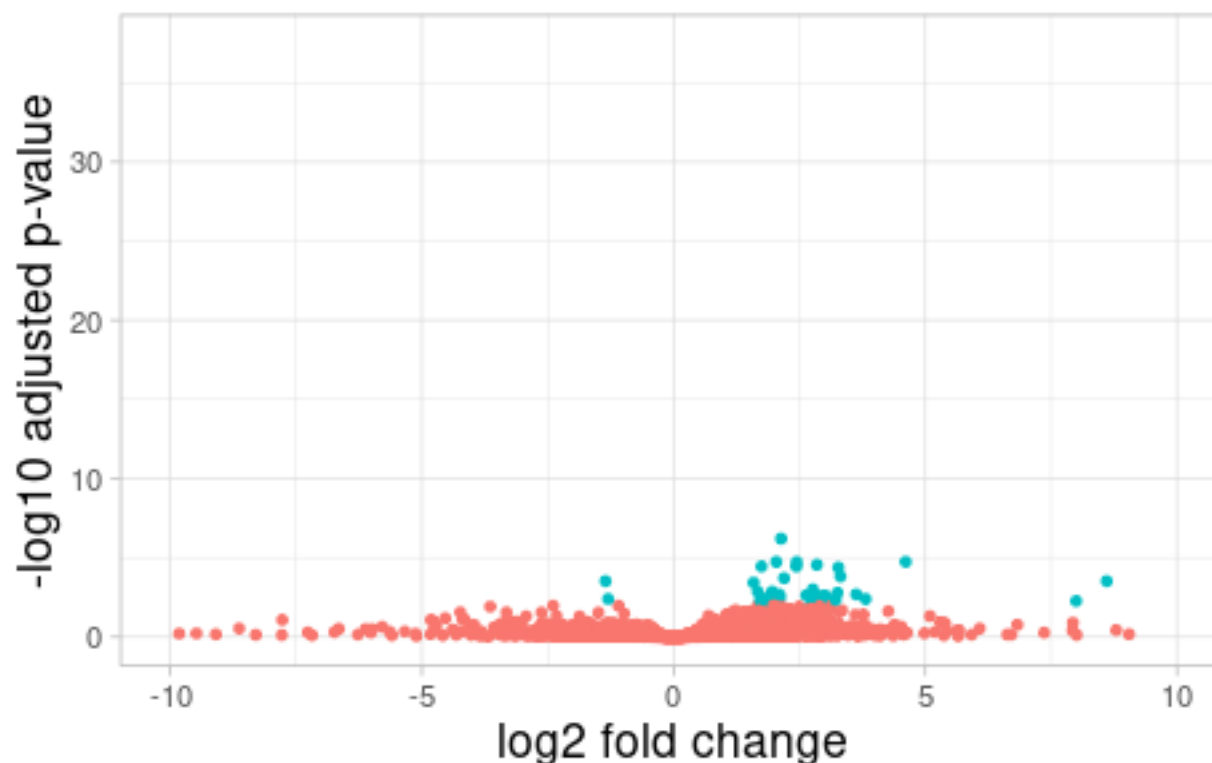
ER: Positive vs Negative



```
# Add a column for significant genes
resTP_tb <- resTP_tb %>% mutate(threshold = padj < 0.01)

ggplot(resTP_tb) +
  geom_point(aes(x = log2FoldChange, y = -log10(padj), colour = threshold)) +
  ggtitle("Tumor_percentage_high: High vs Low") +
  xlab("log2 fold change") +
  ylab("-log10 adjusted p-value") +
  scale_x_continuous(limits = c(-10,10)) +
  theme(legend.position = "none",
        plot.title = element_text(size = rel(1.5), hjust = 0.5),
        axis.title = element_text(size = rel(1.25)))
```

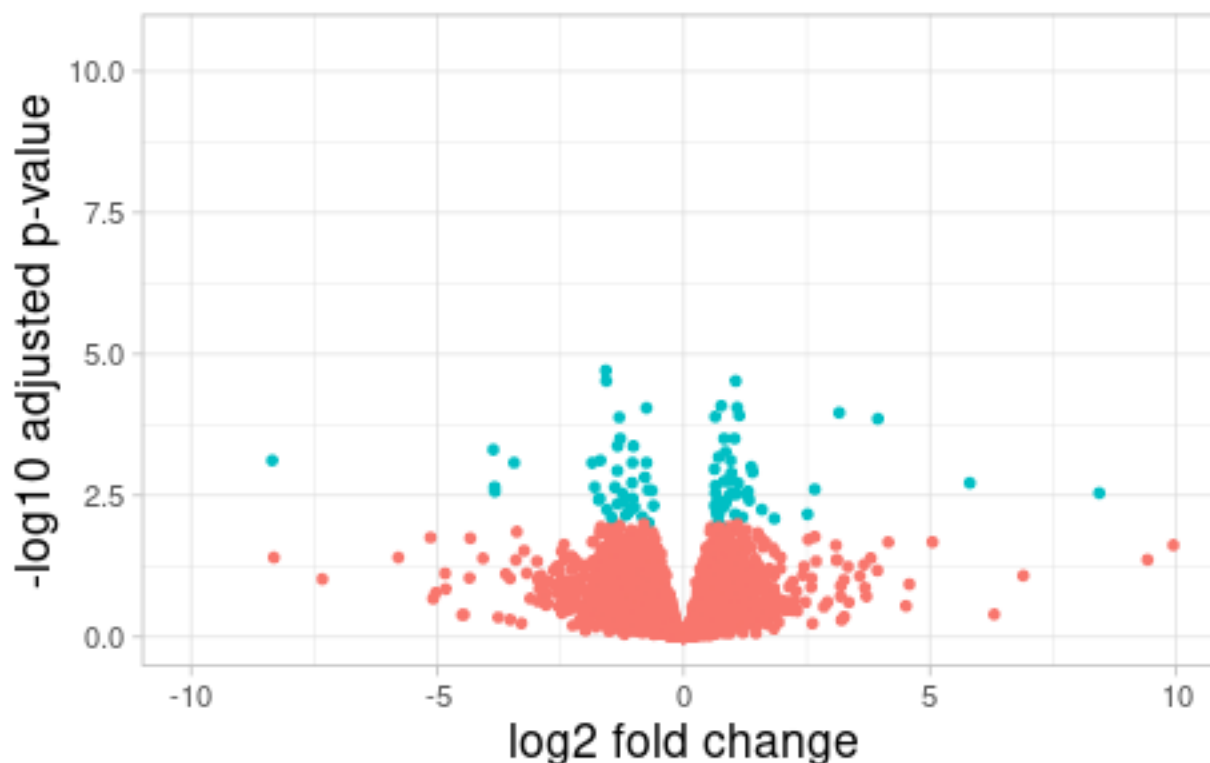
Tumor_percentage_high: High vs Low



```
# Add a column for significant genes
resD0_tb <- resD0_tb %>% mutate(threshold = padj < 0.01)

ggplot(resD0_tb) +
  geom_point(aes(x = log2FoldChange, y = -log10(padj), colour = threshold)) +
  ggtitle("Dafe of: 20180323 vs 20180228") +
  xlab("log2 fold change") +
  ylab("-log10 adjusted p-value") +
  scale_x_continuous(limits = c(-10,10)) +
  theme(legend.position = "none",
        plot.title = element_text(size = rel(1.5), hjust = 0.5),
        axis.title = element_text(size = rel(1.25)))
```

Date of: 20180323 vs 20180228



Heatmaps

Create a matrix of normalized expression

```
sig_up <- resResponse_tb_significant %>% arrange(-log2FoldChange) %>% head(50) %>% pull(gene)
sig_down <- resResponse_tb_significant %>% arrange(log2FoldChange) %>% head(50) %>% pull(gene)
sig <- c(sig_up, sig_down)
```

```
row_annotation <- gene_symbol %>%
  as_tibble() %>%
  dplyr::filter(gene_id %in% sig)
```

```
plotmat <- txi$abundance[c(sig_up, sig_down),] %>% as.data.frame() %>%
  rownames_to_column(var = "ensembl_gene_id") %>%
  left_join(gene_symbol, by = c("ensembl_gene_id" = "gene_id")) %>%
  drop_na(symbol)
```

```
plotmat$ensembl_gene_id <- NULL
```

```
plotmat <- plotmat %>% column_to_rownames(var = "symbol") %>% as.matrix()
```

Color palette

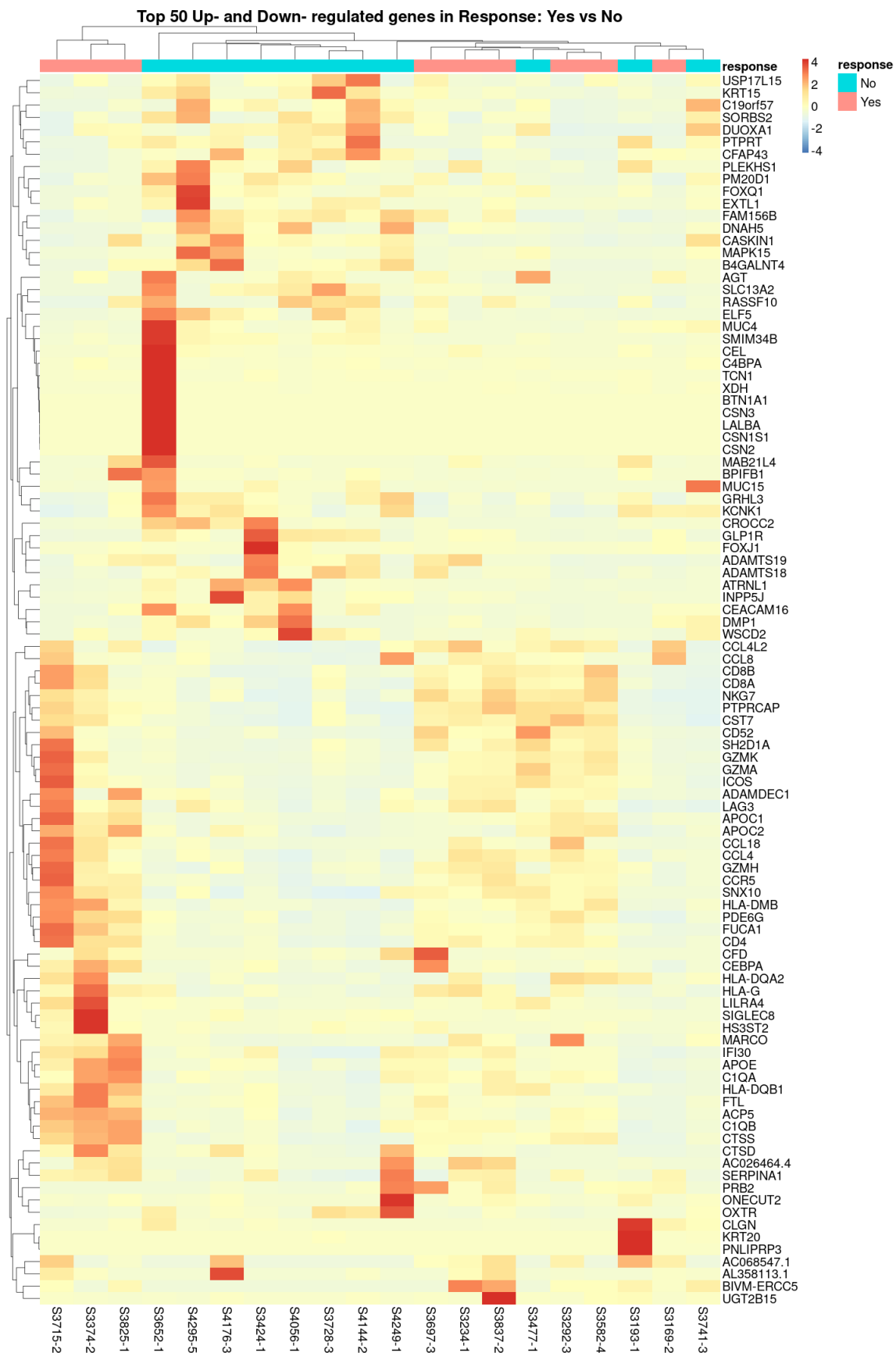
```
heat.colors <- brewer.pal(6, "YlOrRd")
```

Plot heatmap

```
pheatmap(plotmat,
  scale = "row",
  show_rownames = TRUE,
```



```
border = FALSE,  
annotation = meta[, c("response"), drop = FALSE],  
main = "Top 50 Up- and Down- regulated genes in Response: Yes vs No",  
fontsize = 20)
```



```

# Create a matrix of normalized expression
sig_up <- resER_tb_significant %>% arrange(-log2FoldChange) %>% head(50) %>% pull(gene)
sig_down <- resER_tb_significant %>% arrange(log2FoldChange) %>% head(50) %>% pull(gene)
sig <- c(sig_up, sig_down)

row_annotation <- gene_symbol %>%
  as_tibble() %>%
  dplyr::filter(gene_id %in% sig)

plotmat <- txi$abundance[c(sig_up, sig_down),] %>% as.data.frame() %>%
  rownames_to_column(var = "ensembl_gene_id") %>%
  left_join(gene_symbol, by = c("ensembl_gene_id" = "gene_id")) %>%
  drop_na(symbol)

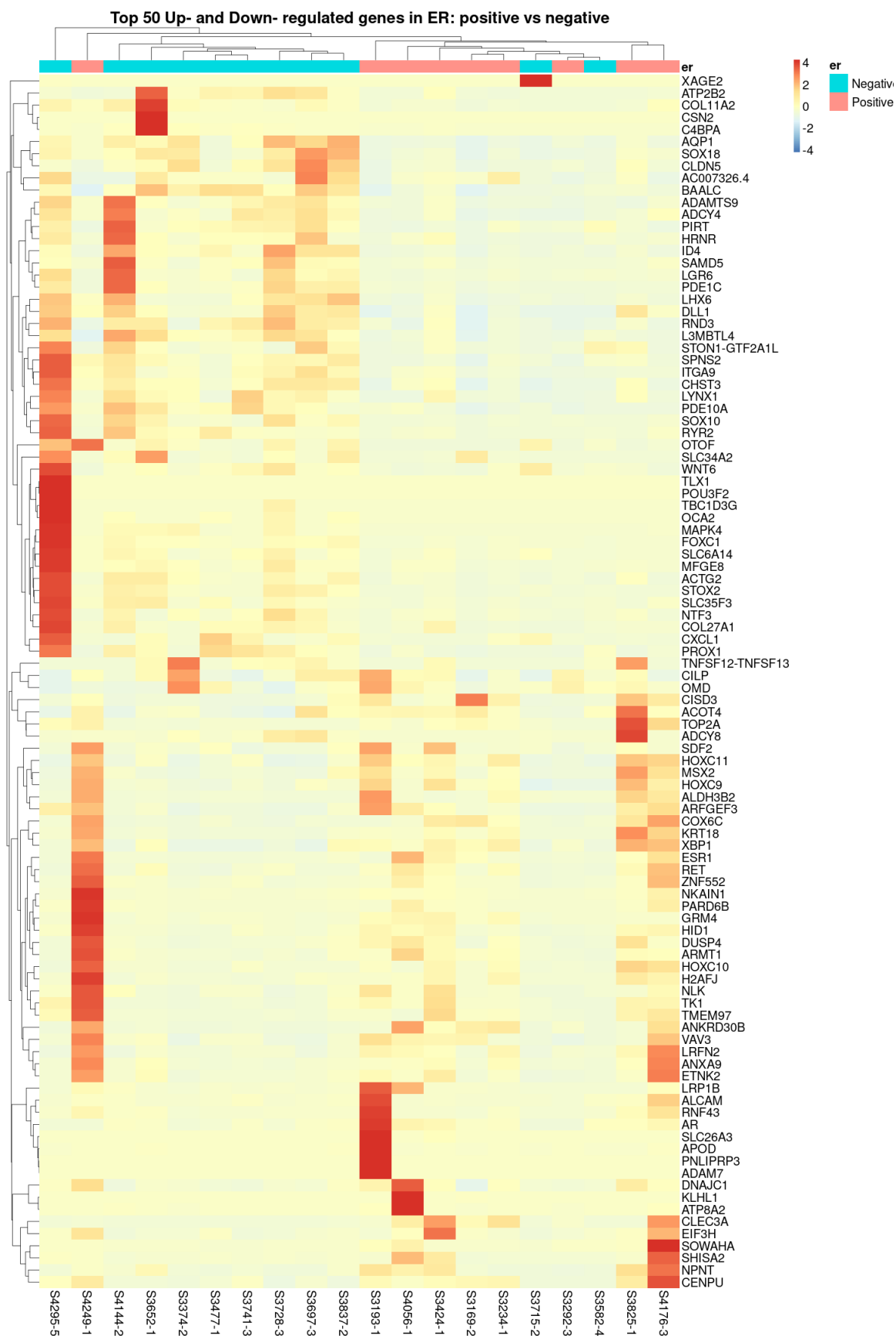
plotmat$ensembl_gene_id <- NULL

plotmat <- plotmat %>% column_to_rownames(var = "symbol") %>% as.matrix()

# Color palette
heat.colors <- brewer.pal(6, "YlOrRd")

# Plot heatmap
pheatmap(plotmat,
  scale = "row",
  show_rownames = TRUE,
  border = FALSE,
  annotation = meta[, c("er"), drop = FALSE],
  main = "Top 50 Up- and Down- regulated genes in ER: positive vs negative",
  fontsize = 20)

```



```

# Create a matrix of normalized expression
sig_up <- resTP_tb_significant %>% arrange(-log2FoldChange) %>% head(50) %>% pull(gene)
sig_down <- resTP_tb_significant %>% arrange(log2FoldChange) %>% head(50) %>% pull(gene)
sig <- c(sig_up, sig_down)

row_annotation <- gene_symbol %>%
  as_tibble() %>%
  dplyr::filter(gene_id %in% sig)

plotmat <- txi$abundance[c(sig_up, sig_down),] %>% as.data.frame() %>%
  rownames_to_column(var = "ensembl_gene_id") %>%
  left_join(gene_symbol, by = c("ensembl_gene_id" = "gene_id")) %>%
  drop_na(symbol)

plotmat$ensembl_gene_id <- NULL

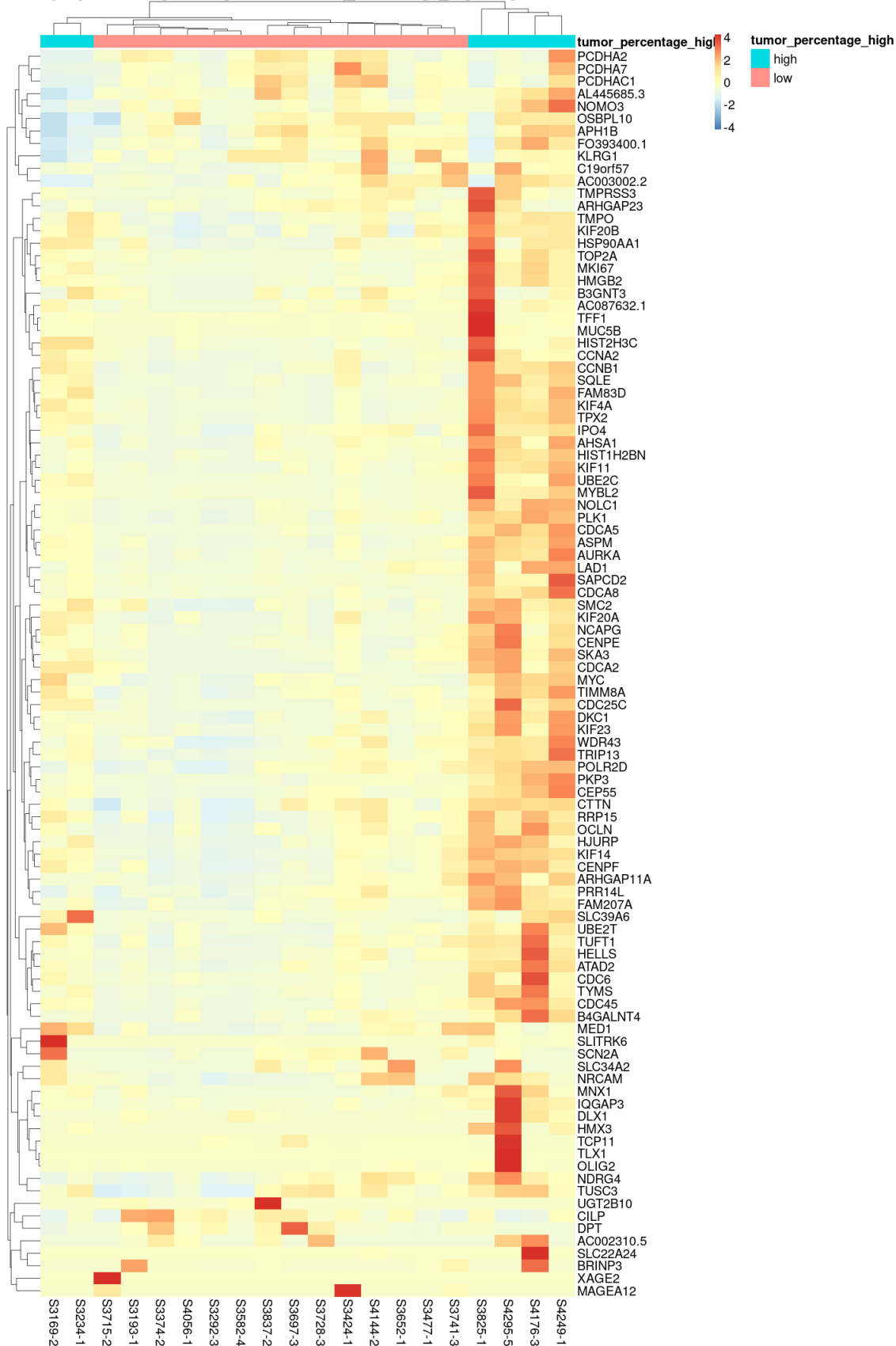
plotmat <- plotmat %>% column_to_rownames(var = "symbol") %>% as.matrix()

# Color palette
heat.colors <- brewer.pal(6, "YlOrRd")

# Plot heatmap
pheatmap(plotmat,
  scale = "row",
  show_rownames = TRUE,
  border = FALSE,
  annotation = meta[, c("tumor_percentage_high"), drop = FALSE],
  main = "Top Up/Down-regulated genes in Tumor_percentage_high: high vs low",
  fontsize = 20)

```

Top Up/Down-regulated genes in Tumor_percentage_high: high vs low



```

# Create a matrix of normalized expression
sig_up <- resD0_tb_significant %>% arrange(-log2FoldChange) %>% head(50) %>% pull(gene)
sig_down <- resD0_tb_significant %>% arrange(log2FoldChange) %>% head(50) %>% pull(gene)
sig <- c(sig_up, sig_down)

row_annotation <- gene_symbol %>%
  as_tibble() %>%
  dplyr::filter(gene_id %in% sig)

plotmat <- txi$abundance[c(sig_up, sig_down),] %>% as.data.frame() %>%
  rownames_to_column(var = "ensembl_gene_id") %>%
  left_join(gene_symbol, by = c("ensembl_gene_id" = "gene_id")) %>%
  drop_na(symbol)

plotmat$ensembl_gene_id <- NULL

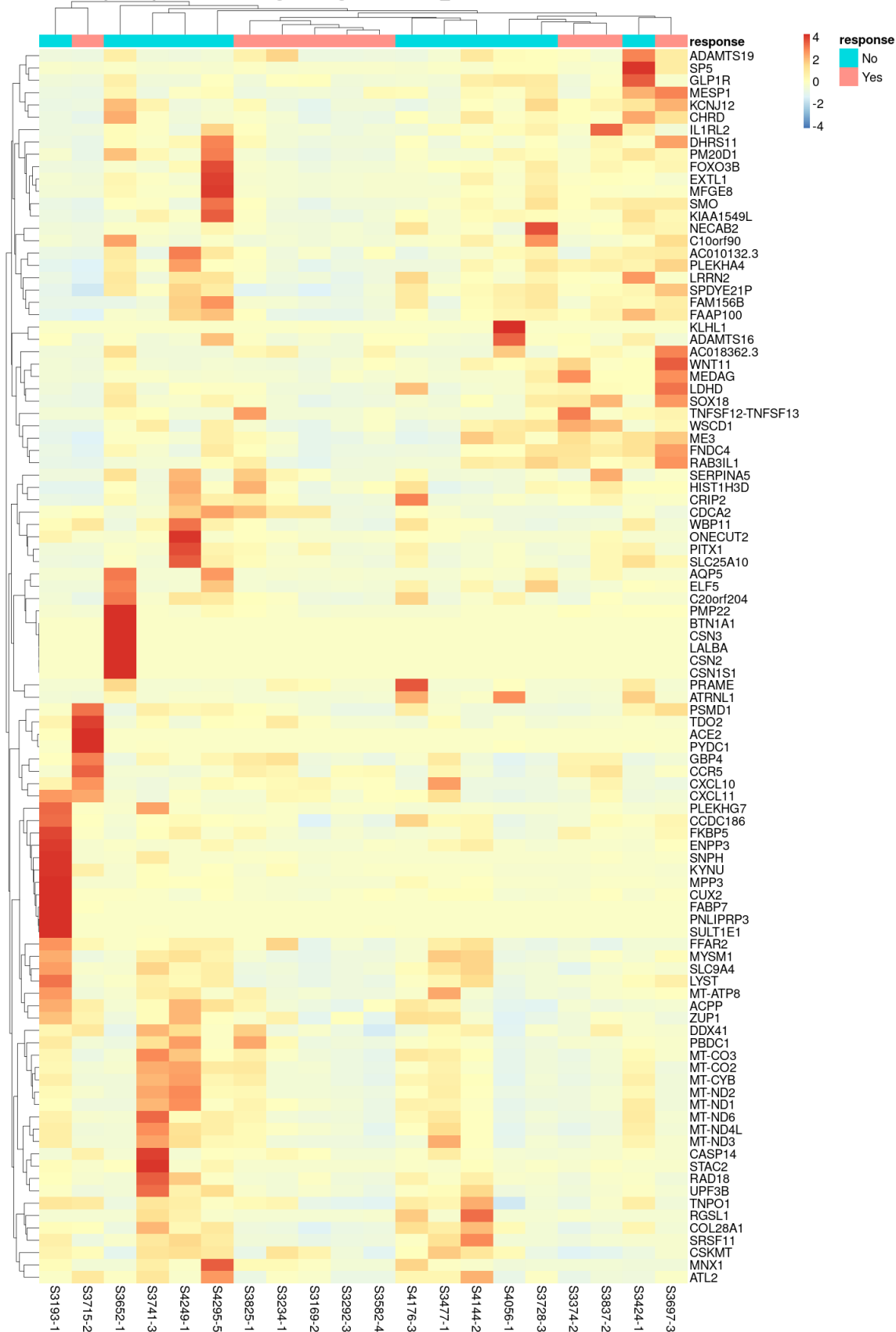
plotmat <- plotmat %>% column_to_rownames(var = "symbol") %>% as.matrix()

# Color palette
heat.colors <- brewer.pal(6, "YlOrRd")

# Plot heatmap
pheatmap(plotmat,
  scale = "row",
  show_rownames = TRUE,
  border = FALSE,
  annotation = meta[, c("response"), drop = FALSE],
  main = "Top 50 Up- and Down- regulated genes in date_of: 20180323 vs 20180228",
  fontsize = 20)

```

Top 50 Up- and Down-regulated genes in date_of: 20180323 vs 20180228




```

# prepares an expression profile for GSEA
# http://software.broadinstitute.org/cancer/software/gsea/wiki/index.php/Data_formats#Expression_Data_
# for GSEA it is important to report all genes - genome wide
# hopefully cpms are better than logcpms
counts <- counts[rowSums(counts)>0,]
result_file <- paste0("tables/8day.4gsea.txt")

counts_gsea <- counts %>% as.data.frame() %>%
  rownames_to_column(var = "ensembl_gene_id") %>%
  left_join(gene_symbol, by = c("ensembl_gene_id" = "gene_id")) %>%
  dplyr::relocate(symbol)
#%>%
# dplyr::relocate(ensembl_gene_id)

colnames(counts_gsea)[1:2] <- c("NAME", "DESCRIPTION")

d <- duplicated(counts_gsea$NAME)
o <- order(rowSums(counts_gsea[,rownames(meta)]),decreasing = T)
counts_gsea <- counts_gsea[o, ]
counts_gsea <- counts_gsea[!d, ]

samples_yes <- meta %>% dplyr::filter(response == "Yes") %>% row.names()
samples_no <- meta %>% dplyr::filter(response == "No") %>% row.names()

counts_gsea <- counts_gsea[,c("NAME", "DESCRIPTION", samples_yes, samples_no)]
# gsea now supports ENSEMBL_IDS
write_tsv(counts_gsea, result_file)

```

Functional analysis

Biological Process (BP)

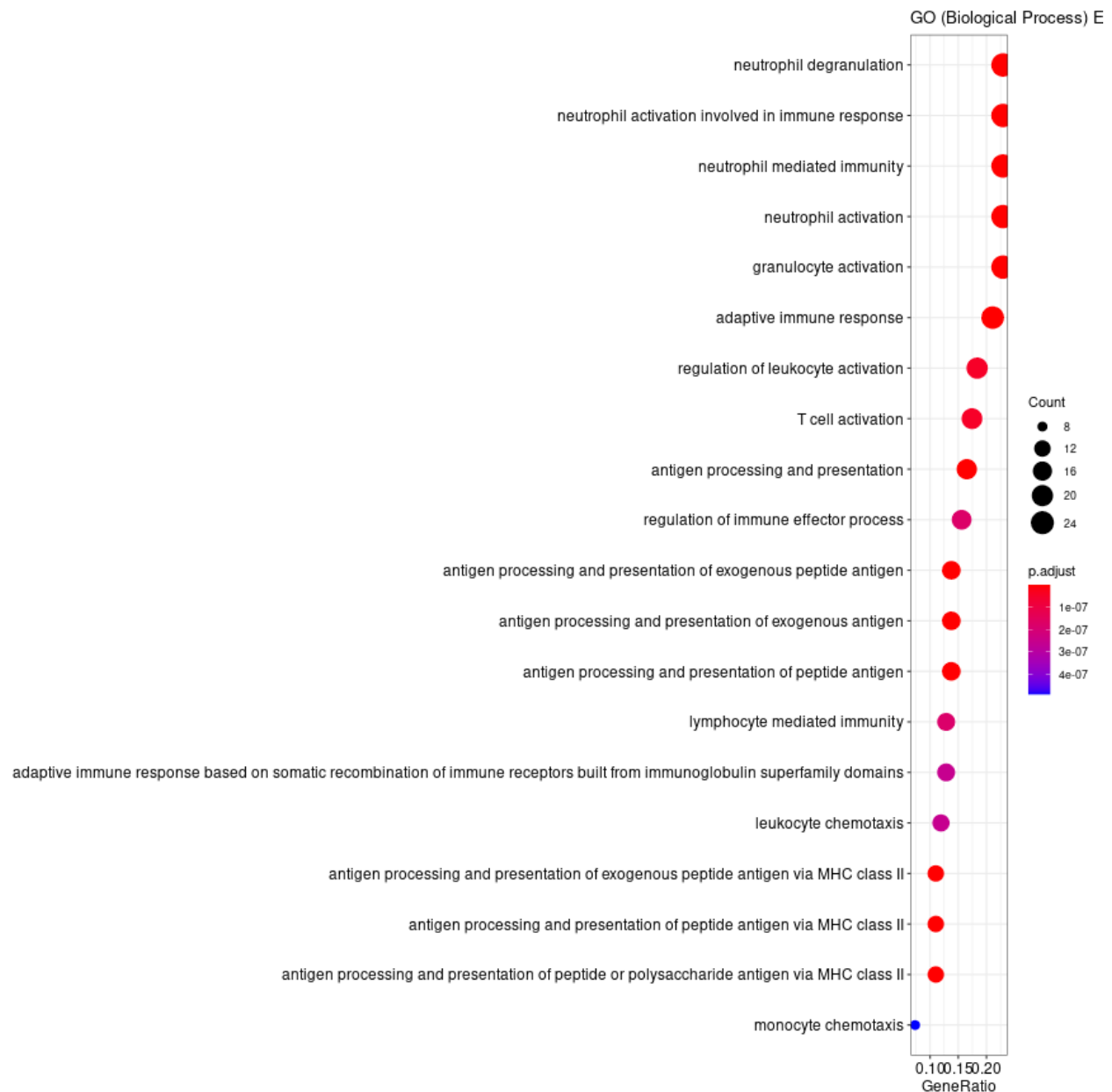
```

bg_genes <- rownames(resResponse)

## Run GO enrichment analysis
compGO <- enrichGO(gene = sigResponse_up,
  universe = bg_genes,
  keyType = "ENSEMBL",
  OrgDb = "org.Hs.eg.db",
  ont = "BP",
  qvalueCutoff = 0.05,
  pAdjustMethod = "BH")

dotplot(compGO,
  showCategory = 20,
  title = "GO (Biological Process) Enrichment Analysis for UP in Treatment (POST)")

```



```
# image pdf 12 x 12
```

```
## Output results from GO analysis to a table
```

```
results_up <- data.frame(compGO@result) %>% dplyr::filter(p.adjust < 0.05)
```

R session

```
sessionInfo()
```

```
## R version 4.0.3 (2020-10-10)
## Platform: x86_64-redhat-linux-gnu (64-bit)
## Running under: Fedora 32 (Workstation Edition)
##
```

```

## Matrix products: default
## BLAS/LAPACK: /usr/lib64/libopenblas-r0.3.12.so
##
## locale:
## [1] LC_CTYPE=en_CA.UTF-8      LC_NUMERIC=C
## [3] LC_TIME=en_CA.UTF-8      LC_COLLATE=en_CA.UTF-8
## [5] LC_MONETARY=en_CA.UTF-8  LC_MESSAGES=en_CA.UTF-8
## [7] LC_PAPER=en_CA.UTF-8     LC_NAME=C
## [9] LC_ADDRESS=C             LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_CA.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel stats4      stats      graphics  grDevices  utils      datasets
## [8] methods    base
##
## other attached packages:
## [1] clusterProfiler_3.18.1      org.Hs.eg.db_3.12.0
## [3] ensemblDb_2.14.0           AnnotationFilter_1.14.0
## [5] GenomicFeatures_1.42.1     AnnotationDbi_1.52.0
## [7] AnnotationHub_2.22.0       BiocFileCache_1.14.0
## [9] dbplyr_2.1.0               knitr_1.30
## [11] ggrepel_0.9.1              tximport_1.18.0
## [13] DEGreport_1.26.0           pheatmap_1.0.12
## [15] RColorBrewer_1.1-2         forcats_0.5.1
## [17] stringr_1.4.0              dplyr_1.0.5
## [19] purrr_0.3.4                readr_1.4.0
## [21] tidyr_1.1.3                tibble_3.1.0
## [23] ggplot2_3.3.3              tidyverse_1.3.0
## [25] DESeq2_1.30.1              SummarizedExperiment_1.20.0
## [27] Biobase_2.50.0             MatrixGenerics_1.2.1
## [29] matrixStats_0.58.0         GenomicRanges_1.42.0
## [31] GenomeInfoDb_1.26.2        IRanges_2.24.1
## [33] S4Vectors_0.28.1          BiocGenerics_0.36.0
##
## loaded via a namespace (and not attached):
## [1] utf8_1.1.4                  tidyselect_1.1.0
## [3] RSQLite_2.2.3               grid_4.0.3
## [5] BiocParallel_1.24.1         scatterpie_0.1.5
## [7] munsell_0.5.0               withr_2.4.1
## [9] colorspace_2.0-0            GOSemSim_2.16.1
## [11] rstudioapi_0.13             DOSE_3.16.0
## [13] labeling_0.4.2              lasso2_1.2-21.1
## [15] GenomeInfoDbData_1.2.4      polyclip_1.10-0
## [17] mnormt_2.0.2                farver_2.1.0
## [19] bit64_4.0.5                 downloader_0.4
## [21] vctrs_0.3.6                 generics_0.1.0
## [23] xfun_0.19                   R6_2.5.0
## [25] graphlayouts_0.7.1          clue_0.3-58
## [27] locfit_1.5-9.4              bitops_1.0-6
## [29] cachem_1.0.4                reshape_0.8.8
## [31] fgsea_1.16.0                DelayedArray_0.16.2
## [33] assertthat_0.2.1            promises_1.2.0.1
## [35] scales_1.1.1                ggraph_2.0.5
## [37] enrichplot_1.10.2           gtable_0.3.0

```

## [39] Cairo_1.5-12.2	tidygraph_1.2.0
## [41] rlang_0.4.10	genefilter_1.72.1
## [43] GlobalOptions_0.1.2	splines_4.0.3
## [45] rtracklayer_1.50.0	lazyeval_0.2.2
## [47] broom_0.7.5	BiocManager_1.30.10
## [49] yaml_2.2.1	reshape2_1.4.4
## [51] modelr_0.1.8	backports_1.2.1
## [53] httpuv_1.5.5	qvalue_2.22.0
## [55] tools_4.0.3	psych_2.0.12
## [57] logging_0.10-108	ellipsis_0.3.1
## [59] ggdendro_0.1.22	Rcpp_1.0.6
## [61] plyr_1.8.6	progress_1.2.2
## [63] zlibbioc_1.36.0	RCurl_1.98-1.2
## [65] prettyunits_1.1.1	openssl_1.4.3
## [67] viridis_0.5.1	GetoptLong_1.0.5
## [69] cowplot_1.1.1	haven_2.3.1
## [71] cluster_2.1.0	fs_1.5.0
## [73] magrittr_2.0.1	data.table_1.14.0
## [75] DO.db_2.9	circlize_0.4.12
## [77] reprex_1.0.0	tmvnsim_1.0-2
## [79] ProtGenerics_1.22.0	hms_1.0.0
## [81] mime_0.9	evaluate_0.14
## [83] xtable_1.8-4	XML_3.99-0.5
## [85] readxl_1.3.1	gridExtra_2.3
## [87] shape_1.4.5	compiler_4.0.3
## [89] biomaRt_2.46.3	shadowtext_0.0.7
## [91] crayon_1.4.1	htmltools_0.5.1.1
## [93] later_1.1.0.1	geneplotter_1.68.0
## [95] lubridate_1.7.10	DBI_1.1.1
## [97] tweenr_1.0.1	ComplexHeatmap_2.6.2
## [99] MASS_7.3-53	rappdirs_0.3.3
## [101] Matrix_1.2-18	cli_2.3.1
## [103] igraph_1.2.6	pkgconfig_2.0.3
## [105] rvcheck_0.1.8	GenomicAlignments_1.26.0
## [107] xml2_1.3.2	annotate_1.68.0
## [109] XVector_0.30.0	rvest_1.0.0
## [111] digest_0.6.27	ConsensusClusterPlus_1.54.0
## [113] Biostrings_2.58.0	rmarkdown_2.5
## [115] cellranger_1.1.0	fastmatch_1.1-0
## [117] edgeR_3.32.1	curl_4.3
## [119] shiny_1.6.0	Rsamtools_2.6.0
## [121] rjson_0.2.20	lifecycle_1.0.0
## [123] nlme_3.1-149	jsonlite_1.7.1
## [125] viridisLite_0.3.0	askpass_1.1
## [127] limma_3.46.0	fansi_0.4.2
## [129] pillar_1.5.1	lattice_0.20-41
## [131] Nozzle.R1_1.1-1	fastmap_1.1.0
## [133] httr_1.4.2	survival_3.2-7
## [135] GO.db_3.12.1	interactiveDisplayBase_1.28.0
## [137] glue_1.4.2	png_0.1-7
## [139] BiocVersion_3.12.0	bit_4.0.4
## [141] ggforce_0.3.3	stringi_1.5.3
## [143] blob_1.2.1	memoise_2.0.0