Introduction to Bulk RNAseq data analysis

Annotation and Visualisation of Differential Expression Results

Last modified: 23 Apr 2021

Contents


```
library(AnnotationHub)
library(AnnotationDbi)
library(DESeq2)
library(tidyverse)
```

Before starting this section, we will make sure we have all the relevant objects from the Differential Expression analysis.

```
ddsObj.interaction <- readRDS("RObjects/DESeqDataSet.interaction.rds")
results.interaction.11 <- readRDS("RObjects/DESeqResults.interaction_d11.rds")
results.interaction.33 <- readRDS("RObjects/DESeqResults.interaction_d33.rds")
```

Overview

- Getting annotation
- Visualising DE results

Adding annotation to the DESeq2 results

We have a list of significantly differentially expressed genes, but the only annotation we can see is the Ensembl Gene ID, which is not very informative.

There are a number of ways to add annotation. One method is to do this using a Bioconductor annotation package. These packages which are re-built every periodically with the latest annotations. These packages are listed on the annotation section of the Bioconductor, and are installed in the same way as regular Bioconductor packages.

An another approach is to use `biomaRt`, an interface to the BioMart resource. Using BioMart ensures that you are able to get the latest annotations for the GeneIDs, and can match the version of the gene annotation that was used for read counting.

A third method is to use `AnnotationHub`, this is like the bioconductor packages but in an online database like `bioMaRt`. They keep them slightly more up to date than the standard bioconductor packages and each time you use them the results are cached on your machine.

Today we will use the `AnnotationHub` method. A workflow for annotation with biomaRt is included in the extended materials section acessible on the course website.

Query the database

First we need to get the correct database from `AnnotationHub`. We make the instance (the first time we do this it will create a local cache on your machine so that repeat queries are very quick).

As you can see `ah` contains huge amounts of information and it is constantly changing. This is why it gives us the snapshot date so we know when our cached version is from. The `ah` object actually online contains pointers to where all the information is online and we don't want to download all of them as it would take a very long time and we don't need all of it.

This object is a vector and you can get information about a single resource by indexing with a single bracket `[` or download a resource with a double bracket `[[`.

```
# create an annotationhub instance
ah <- AnnotationHub()
ah
```

```
## AnnotationHub with 57231 records
## # snapshotDate(): 2020-10-27
## # $dataprovider: Ensembl, BroadInstitute, UCSC, ftp://ftp.ncbi.nlm.nih.gov/g...
## # $species: Homo sapiens, Mus musculus, Drosophila melanogaster, Bos taurus,...
## # $rdataclass: GRanges, TwoBitFile, BigWigFile, EnsDb, Rle, OrgDb, ChainFile...
## # additional mcols(): taxonomyid, genome, description,
## #   coordinate_1_based, maintainer, rdatadateadded, preparerclass, tags,
## #   rdatapath, sourceurl, sourcetype
## # retrieve records with, e.g., 'object[["AH5012"]]'
##
##            title
##   AH5012  | Chromosome Band
##   AH5013  | STS Markers
##   AH5014  | FISH Clones
##   AH5015  | Recomb Rate
##   AH5016  | ENCODE Pilot
##   ...       ...
##   AH91566 | Zonotrichia_albicollis.Zonotrichia_albicollis-1.0.1.ncrna.2bit
```

```
##   AH91567 | Zosterops_lateralis_melanops.ASM128173v1.cdna.all.2bit
##   AH91568 | Zosterops_lateralis_melanops.ASM128173v1.dna_rm.toplevel.2bit
##   AH91569 | Zosterops_lateralis_melanops.ASM128173v1.dna_sm.toplevel.2bit
##   AH91570 | Zosterops_lateralis_melanops.ASM128173v1.ncrna.2bit
```

```
ah[1]
```

```
## AnnotationHub with 1 record
## # snapshotDate(): 2020-10-27
## # names(): AH5012
## # $dataprovider: UCSC
## # $species: Homo sapiens
## # $rdataclass: GRanges
## # $rdatadateadded: 2013-03-26
## # $title: Chromosome Band
## # $description: GRanges object from UCSC track 'Chromosome Band'
## # $taxonomyid: 9606
## # $genome: hg19
## # $sourcetype: UCSC track
## # $sourceurl: rtracklayer://hgdownload.cse.ucsc.edu/goldenpath/hg19/database...
## # $sourcesize: NA
## # $tags: c("cytoBand", "UCSC", "track", "Gene", "Transcript",
## #   "Annotation")
## # retrieve record with 'object[["AH5012"]]'
```

```
# Download the database we want to use
OrgDb <- query(ah, c("OrgDb", "Mus musculus"))[[1]]
```

We use the select function to query the database. Now we need to set up a query. This requires us to tell it what we want and what we have. For this we need to specify three things:

(a) What type of information we are going to search the dataset on - called **keytypes**. In our case this is Ensembl Gene IDs
(b) A vector of the **keys** for our filter - the Ensembl Gene IDs from our DE results table
(c) What columns (**columns**) of the dataset we want returned.

```
# what can we search for? 'columns'
columns(OrgDb)
```

```
##  [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"  "ENSEMBLTRANS"
##  [6] "ENTREZID"     "ENZYME"       "EVIDENCE"     "EVIDENCEALL"  "GENENAME"
## [11] "GO"           "GOALL"        "IPI"          "MGI"          "ONTOLOGY"
## [16] "ONTOLOGYALL"  "PATH"         "PFAM"         "PMID"         "PROSITE"
## [21] "REFSEQ"       "SYMBOL"       "UNIGENE"      "UNIPROT"
```

```
# what can we search with? 'keytypes'
keytypes(OrgDb)
```

```
##  [1] "ACCNUM"       "ALIAS"        "ENSEMBL"      "ENSEMBLPROT"  "ENSEMBLTRANS"
##  [6] "ENTREZID"     "ENZYME"       "EVIDENCE"     "EVIDENCEALL"  "GENENAME"
## [11] "GO"           "GOALL"        "IPI"          "MGI"          "ONTOLOGY"
## [16] "ONTOLOGYALL"  "PATH"         "PFAM"         "PMID"         "PROSITE"
## [21] "REFSEQ"       "SYMBOL"       "UNIGENE"      "UNIPROT"
```

```r
# lets set it up
ourCols <- c("SYMBOL", "ENSEMBL", "ENTREZID")
ourKeys <- rownames(results.interaction.11)[1:1000]

# run the query
annot <- AnnotationDbi::select(OrgDb,
                keys=ourKeys,
                columns=ourCols,
                keytype="ENSEMBL")
```

One-to-many relationships

Let's inspect the annotation.

```r
head(annot)
```

```
##              ENSEMBL SYMBOL ENTREZID
## 1 ENSMUSG00000000001  Gnai3    14679
## 2 ENSMUSG00000000028  Cdc45    12544
## 3 ENSMUSG00000000037  Scml2   107815
## 4 ENSMUSG00000000049   Apoh    11818
## 5 ENSMUSG00000000056   Narf    67608
## 6 ENSMUSG00000000058   Cav2    12390
```

```r
length(unique(annot$ENTREZID))
```

```
## [1] 1008
```

```r
sum(is.na(annot$ENTREZID)) # Why are there NAs in the ENTREZID column?
```

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

```r
dim(annot) # why are there more than 1000 rows?
```

```
## [1] 1008    3
```

```r
# find all rows containing duplicated ensembl ids
annot %>%
    add_count(ENSEMBL) %>%
    dplyr::filter(n>1)
```

```
##               ENSEMBL        SYMBOL  ENTREZID n
## 1  ENSMUSG00000000486       Septin1     54204 2
## 2  ENSMUSG00000000486        Gm4532 100043580 2
## 3  ENSMUSG00000000562        Adora3     11542 2
## 4  ENSMUSG00000000562        Tmigd3     69296 2
## 5  ENSMUSG00000001768          Rin2     74030 2
## 6  ENSMUSG00000001768      BC039771    408057 2
## 7  ENSMUSG00000002250         Ppard     19015 2
```

```
## 8  ENSMUSG00000002250 1810013A23Rik     69050 2
## 9  ENSMUSG00000003271       Sult2b1     54200 2
## 10 ENSMUSG00000003271        Gm5897    545963 2
## 11 ENSMUSG00000003812       Dnase2a     13423 2
## 12 ENSMUSG00000003812       Gm38426 100503676 2
## 13 ENSMUSG00000004455        Ppp1cc     19047 2
## 14 ENSMUSG00000004455       Ppp1ccb    434233 2
## 15 ENSMUSG00000005983 1700037C18Rik     73261 2
## 16 ENSMUSG00000005983       Gm41410 105246059 2
```

There is one Ensembl IDs with no EntrezID. These gene ids has no corresponding Entrez ID in the `OrgDb` database package. The Ensembl and Entrez databases don't match on a 1:1 level although they have started taking steps towards consolidating in recent years.

There are some genes that have multiple entries in the retrieved annotation. This is because there are multiple Entrez IDs for a single Ensembl gene. These one-to-many relationships come up frequently in genomic databases, it is important to be aware of them and check when necessary.

We will need to do a little work to account for these one-to-many relationships before adding the annotation to our results table. We could decide that the mappings are ambiguous and elect to discard both of the Entrez ID mappings. We could concatenate the Entrez IDs so that we don't lose information. Alternatively, we could spend some time manually comparing the details of the genes on the Ensembl and NCBI websites and make a decision as to which Entrez ID to keep, e.g. Rpl13:

- Ensembl: ENSMUSG00000000740
- Entrez ID: 270106
- Entrez ID: 100040416

In this case it would perhaps makes most sense to discard the Entrez ID **100040416**, which is for a pseudogene.

Exercise 1 - Retrieve the full annotation

> So far we have retrieved the annotation for just 1000 genes, but we need annotations for the entire results table.
>
> A reminder of the code we have used so far:

```
# lets set it up
ourCols <- c("ENSEMBL", "SYMBOL", "ENTREZID")
ourKeys <- rownames(results.interaction.11)[1:1000]

# run the query
annot <- AnnotationDbi::select(OrgDb,
                keys=ourKeys,
                columns=ourCols,
                keytype="ENSEMBL")
```

(a) Run the same query using all of the genes in our results table (`results.interaction.33`), and this time include the descriptive name of the genes too. Hint: You can find the name of the column for this by running `columns(OrgDb)`

(b) How many Ensembl genes have multipe Entrez IDs associated with them?

(c) Are all of the Ensembl gene IDs annotated? If not, why do you think this is?

In this case the problems with the annotation aren't due to the versions as our snapshot was made 27-10-2020 which is the same date as version 102 (we used version 102 for counting reads in Day 1) was released. It is good to keep an eye on these things so you don't miss releases.

The is another set of databases within `AnnotationHub` which you can call instead called `EnsDb` and for these you can specify which release you prefer if so some reason you were not using the latest version.

A curated annotation - one we prepared earlier

Dealing with all the one-to-many annotation mappings requires some manual curation of your annotation table.

To save time we have created an annotation table in which we have modified the column names and dealt with the one-to-many/missing issues for Entrez IDs.

The code we used for doing this is available in the extended materials section.

```
ensemblAnnot <- readRDS("RObjects/Ensembl_annotations.rds")
colnames(ensemblAnnot)
```

```
## [1] "GeneID"      "Entrez"      "Symbol"      "Description" "Biotype"
## [6] "Chr"         "Start"       "End"         "Strand"
```

```
annot.interaction.11 <- as.data.frame(results.interaction.11) %>%
    rownames_to_column("GeneID") %>%
    left_join(ensemblAnnot, "GeneID") %>%
    rename(logFC=log2FoldChange, FDR=padj)
```

Finally we can output the annotation DE results using `write_tsv`.

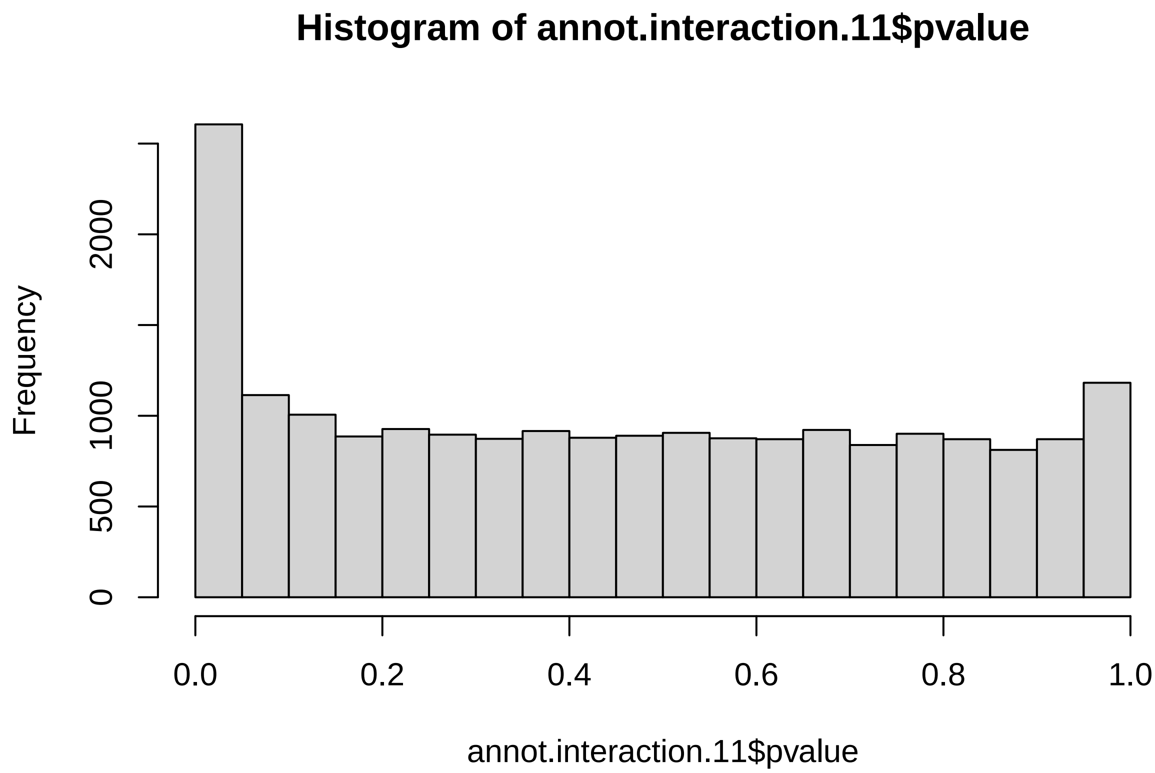
```
write_tsv(annot.interaction.11, "results/Interaction.11_Results_Annotated.txt")
```

Visualisation

P-value histogram

A quick and easy "sanity check" for our DE results is to generate a p-value histogram. What we should see is a high bar at `0 - 0.05` and then a roughly uniform tail to the right of this. There is a nice explanation of other possible patterns in the histogram and what to do when you see them in this post.

```
hist(annot.interaction.11$pvalue)
```



Shrinking the log2FoldChange

`DESeq2` provides a functon called `lfcShrink` that shrinks log-Fold Change (LFC) estimates towards zero using and empirical Bayes procedure. The reason for doing this is that there is high variance in the LFC estimates when counts are low and this results in lowly expressed genes appearing to show greater differences between groups than highly expressed genes. The `lfcShrink` method compensates for this and allows better visualisation and ranking of genes. We will use it for our visualisation of the data.

```
ddsShrink.11 <- lfcShrink(ddsObj.interaction,
                       res = results.interaction.11,
                       type = "ashr")
```

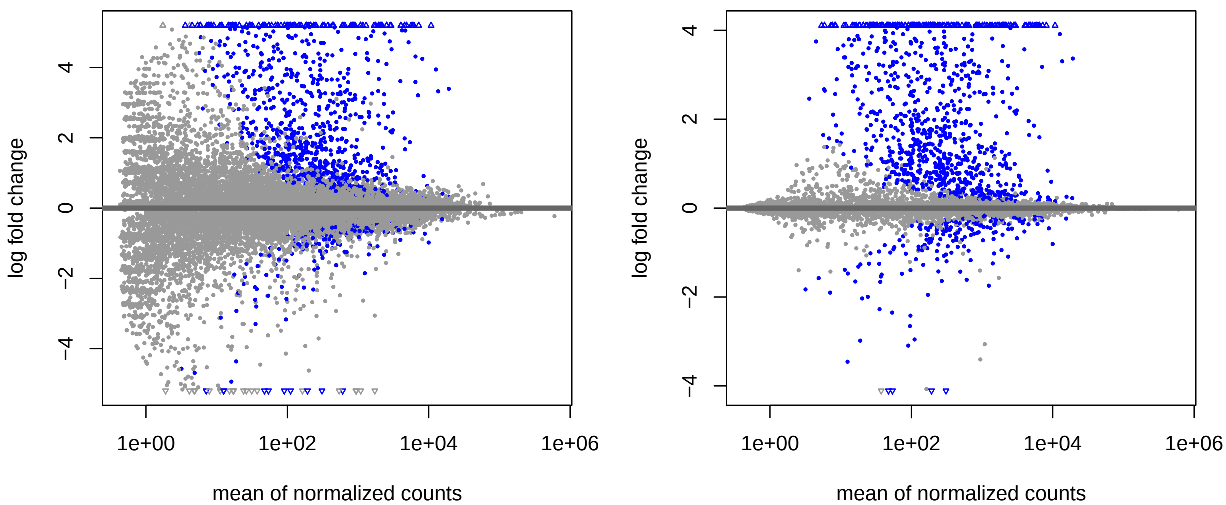
```
## using 'ashr' for LFC shrinkage. If used in published research, please cite:
##     Stephens, M. (2016) False discovery rates: a new deal. Biostatistics, 18:2.
##     https://doi.org/10.1093/biostatistics/kxw041
```

```
shrinkTab.11 <- as.data.frame(ddsShrink.11) %>%
    rownames_to_column("GeneID") %>%
    left_join(ensemblAnnot, "GeneID") %>%
    rename(logFC=log2FoldChange, FDR=padj)
```

MA plots

MA plots are a common way to visualize the results of a differential analysis. We met them briefly towards the end of the DESeq2 session. This plot shows the log-Fold Change for each gene against its average expression across all samples in the two conditions being contrasted. `DESeq2` has a handy function for plotting this. Let's use it too compare the shrunk and un-shrunk fold changes.

```
par(mfrow=c(1,2))
plotMA(results.interaction.11, alpha=0.05)
plotMA(ddsShrink.11, alpha=0.05)
```



The DESeq2 in `plotMA` function is fine for a quick look, but these inbuilt functions aren't easy to customise, make changes to the way it looks or add things such as gene labels. For this we would recommend using the ggplot package.

Volcano Plots

Another common visualisation is the *volcano plot* which displays a measure of significance on the y-axis and fold-change on the x-axis. We will use ggplot to create this.

A Brief Introduction to `ggplot2`

The `ggplot2` package has emerged as an attractive alternative to the traditional plots provided by base R. A full overview of all capabilities of the package is available from the cheatsheet.

In brief:-

- `shrinkTab.11` is our data frame containing the variables we wish to plot
- `aes` creates a mapping between the variables in our data frame to the ***aes***thetic properties of the plot:
    - the x-axis will be mapped to `logFC`

    – the y-axis will be mapped to the `-log10(pvalue)`

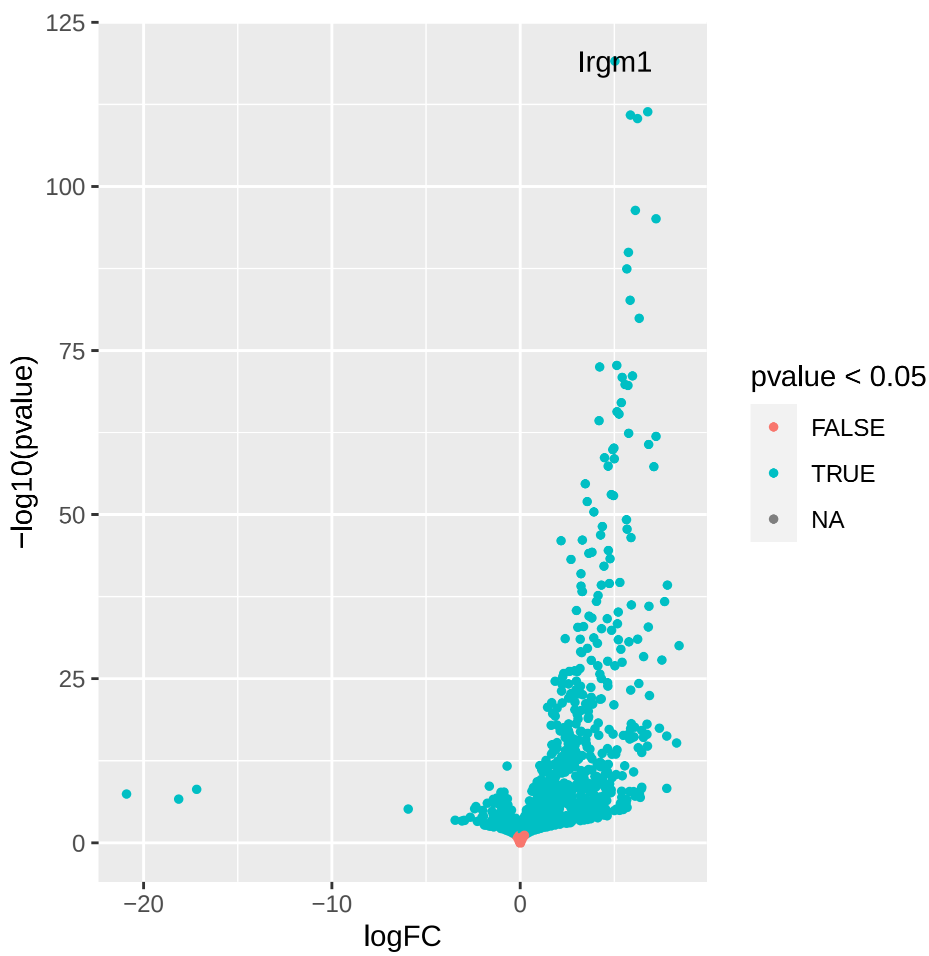
- `geom_point` specifies the particular type of plot we want (in this case a scatter plot)
- `geom_text` allows us to add labels to some or all of the points
    – see the cheatsheet for other plot types

The real advantage of `ggplot2` is the ability to change the appearance of our plot by mapping other variables to aspects of the plot. For example, we could colour the points based on the sample group. To do this we can add metadata from the `sampleinfo` table to the data. The colours are automatically chosen by `ggplot2`, but we can specifiy particular values. For the volcano plot we will colour according whether the gene has a pvalue below 0.05. We use a `-log10` transformation for the y-axis; it's commonly used for p-values as it means that more significant genes have a higher scale.

```
volcanoTab.11 <- shrinkTab.11 %>%
    mutate(`-log10(pvalue)` = -log10(pvalue))

ggplot(volcanoTab.11, aes(x = logFC, y=`-log10(pvalue)`)) +
    geom_point(aes(colour=pvalue < 0.05), size=1) +
    geom_text(data=~top_n(.x, 1, wt=-FDR), aes(label=Symbol))
```

```
## Warning: Removed 47 rows containing missing values (geom_point).
```



Exercise 2 - Volcano plot for 33 days

Now it's your turn! We just made the volcano plot for the 11 days contrast, you will make the one for the 33 days contrast.

If you haven't already make sure you load in our data and annotation. You can copy and paste the code below.

```
# First load data and annotations
results.interaction.33 <- readRDS("RObjects/DESeqResults.interaction_d33.rds")
ensemblAnnot <- readRDS("RObjects/Ensembl_annotations.rds")
```

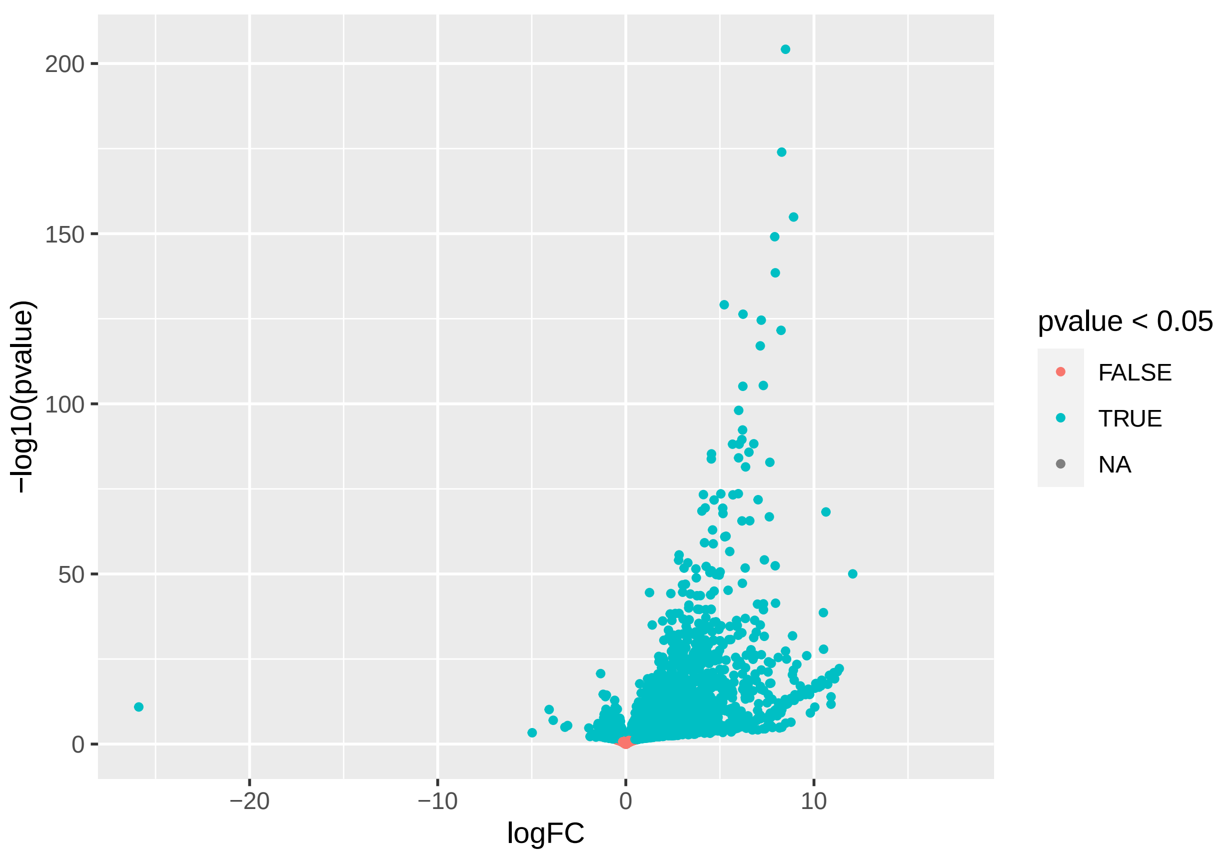
(a) Shrink the results for the 33 days contrast.

```
## using 'ashr' for LFC shrinkage. If used in published research, please cite:
##     Stephens, M. (2016) False discovery rates: a new deal. Biostatistics, 18:2.
##     https://doi.org/10.1093/biostatistics/kxw041
```

(b) Create a new column of -log10(pvalue) values in your shrinkTab for 33 days.

(c) Create a plot with points coloured by P-value < 0.05 similar to how we did in the first volcano plot

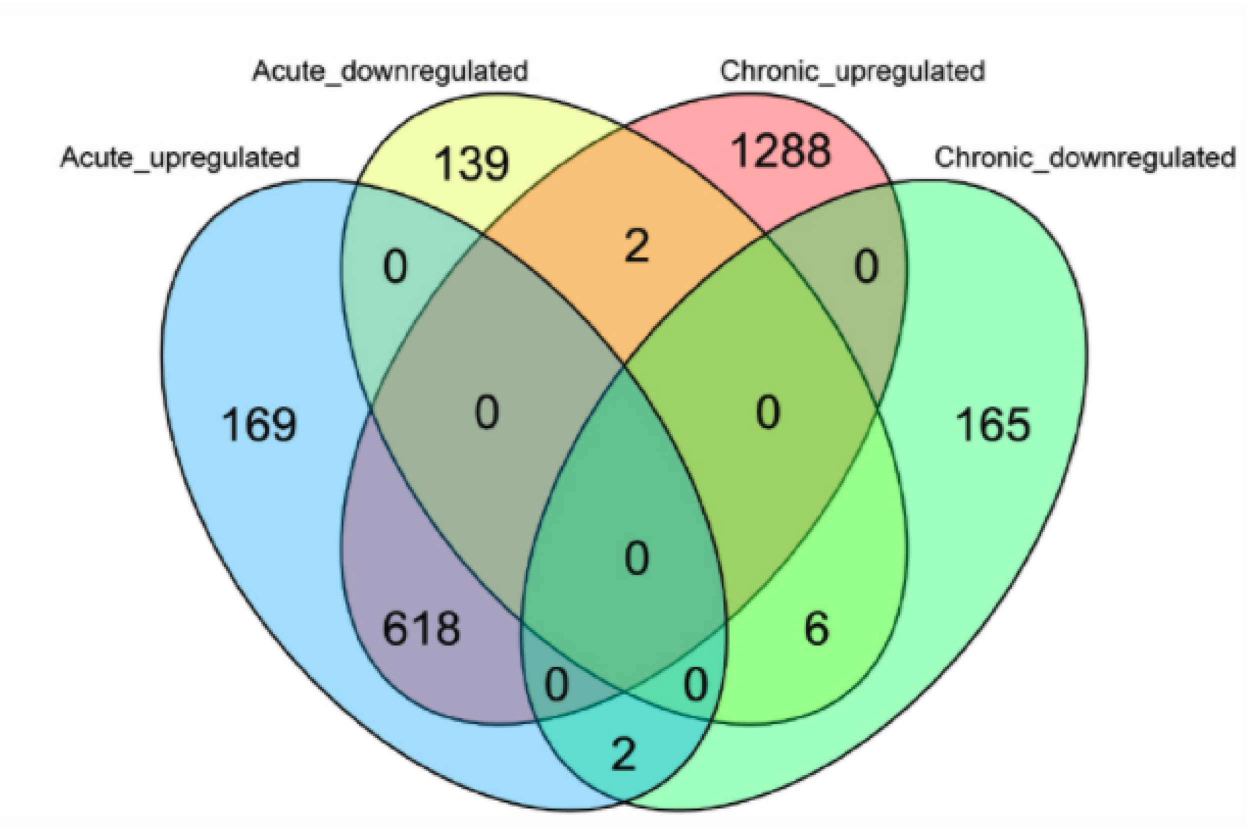
```
## Warning: Removed 47 rows containing missing values (geom_point).
```



(d) Compare these two volcano plots, what differences can you see between the two contrasts?

Venn Diagram

In the paper you may notice they have presented a Venn diagram of the results.



We will recreate it with our analysis. To do this we are using the package `ggvenn` which is an extension to `ggplot` from Linlin Yan.

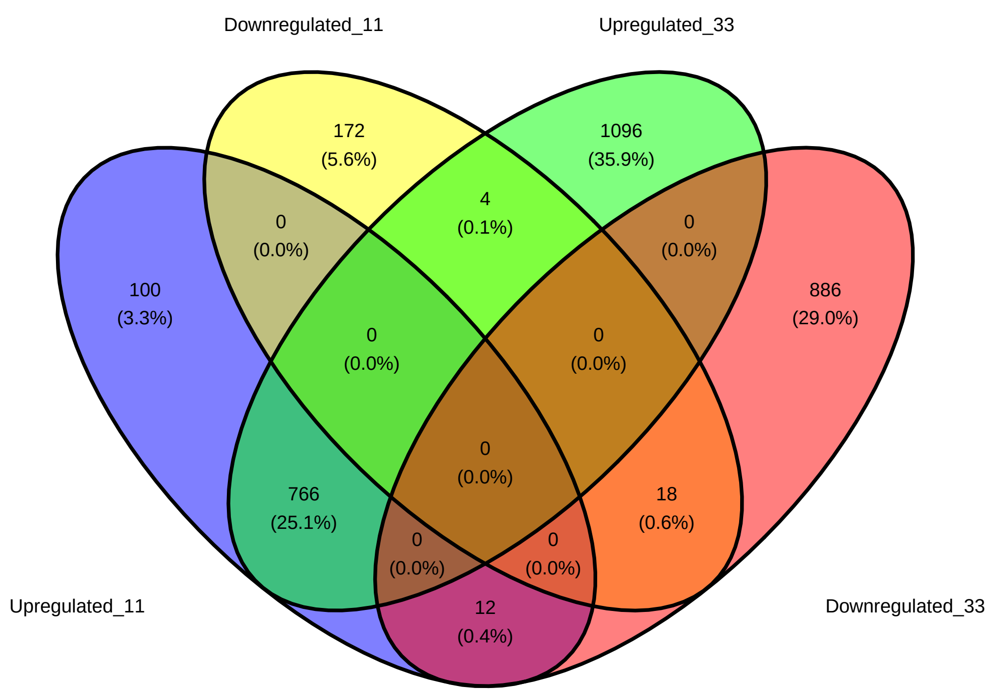
```
library(ggvenn)
```

```
## Loading required package: grid
```

First we have to prepare the data with a column for each set we want in the Venn.

```
vennDat <- tibble(Geneid=rownames(results.interaction.11)) %>%
  mutate(Upregulated_11 = results.interaction.11$padj < 0.05 & !is.na(results.interaction.11$padj) & res
  mutate(Downregulated_11 = results.interaction.11$padj < 0.05 & !is.na(results.interaction.11$padj) & r
  mutate(Upregulated_33 = results.interaction.33$padj < 0.05 & !is.na(results.interaction.33$padj) & res
  mutate(Downregulated_33 = results.interaction.33$padj < 0.05 & !is.na(results.interaction.33$padj) & r

ggvenn(vennDat, set_name_size = 4)
```



Heatmap

We're going to use the package `ComplexHeatmap` (**???**). We'll also use `circlize` to generate a colour scale (**???**).

```
library(ComplexHeatmap)
library(circlize)
```

We can't plot the entire data set, let's just select the top 300 by FDR. We'll want to use normalised expression values, so we'll use the `vst` function.

```
# get the top genes
sigGenes <- shrinkTab.11 %>%
    top_n(300, wt=-FDR) %>%
    pull("GeneID")

# filter the data for the top 300 by padj
plotDat <- vst(ddsObj.interaction)[sigGenes,] %>%
  assay()
```

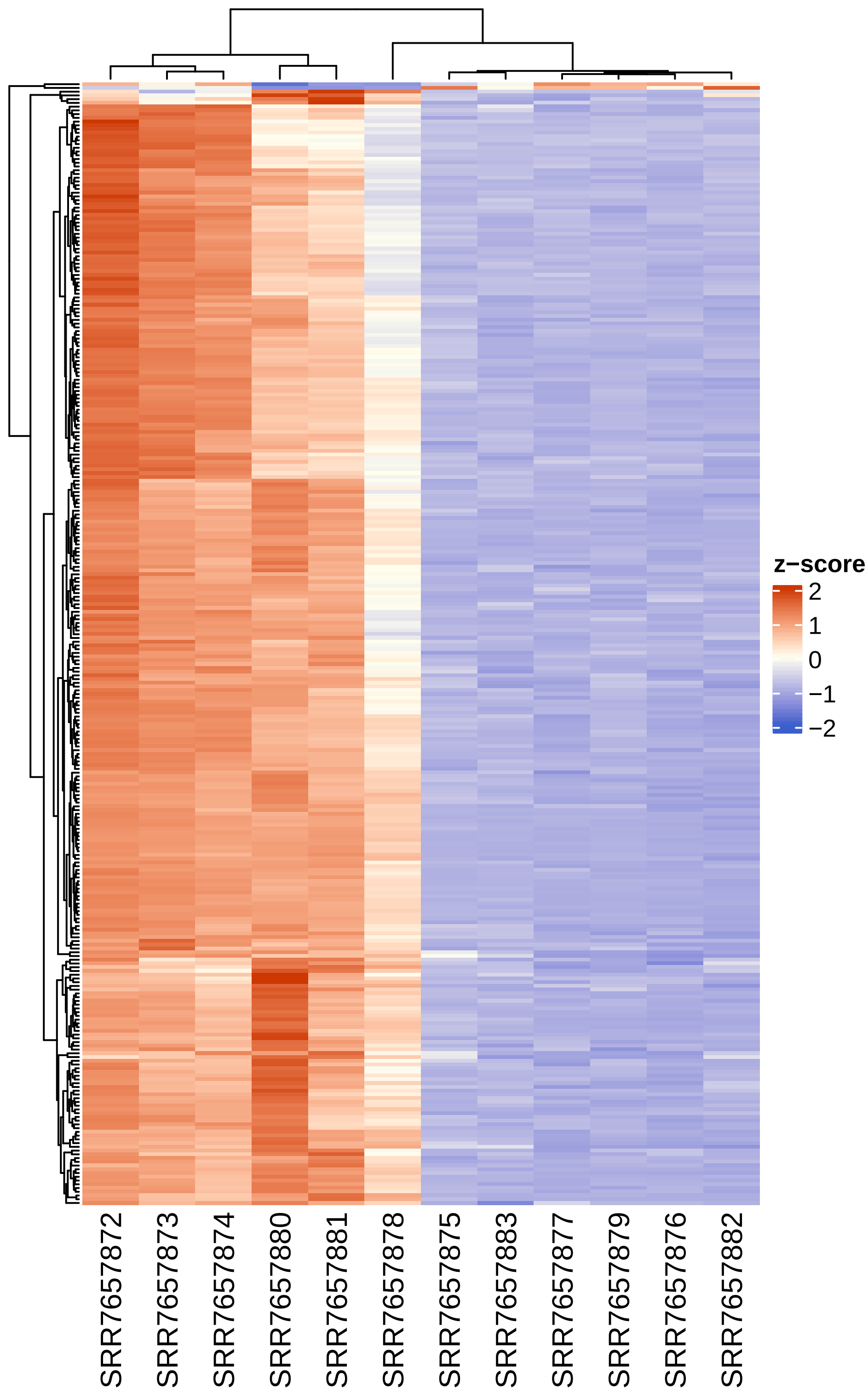
The range expression values for different genes can vary widely. Some genes will have very high expression. Our heatmap is going to be coloured according to gene expression. If we used a colour scale from 0 (no expression) to the maximum expression, the scale will be dominated by our most extreme genes and it will be difficult to discern any difference between most of the genes.

To overcome this we will z-scale the counts. This scaling method results in values for each that show the number of standard deviations the gene expression is from the mean for that gene across all the sample - the mean will be '0', '1' means 1 standard deviation higher than the mean, '-1' means 1 standard deviation lower than the mean.

```
z.mat <- t(scale(t(plotDat), center=TRUE, scale=TRUE))
```

```
# colour palette
myPalette <- c("royalblue3", "ivory", "orangered3")
myRamp <- colorRamp2(c(-2, 0, 2), myPalette)
```

```
Heatmap(z.mat, name = "z-score",
        col = myRamp,
        show_row_names = FALSE)
```

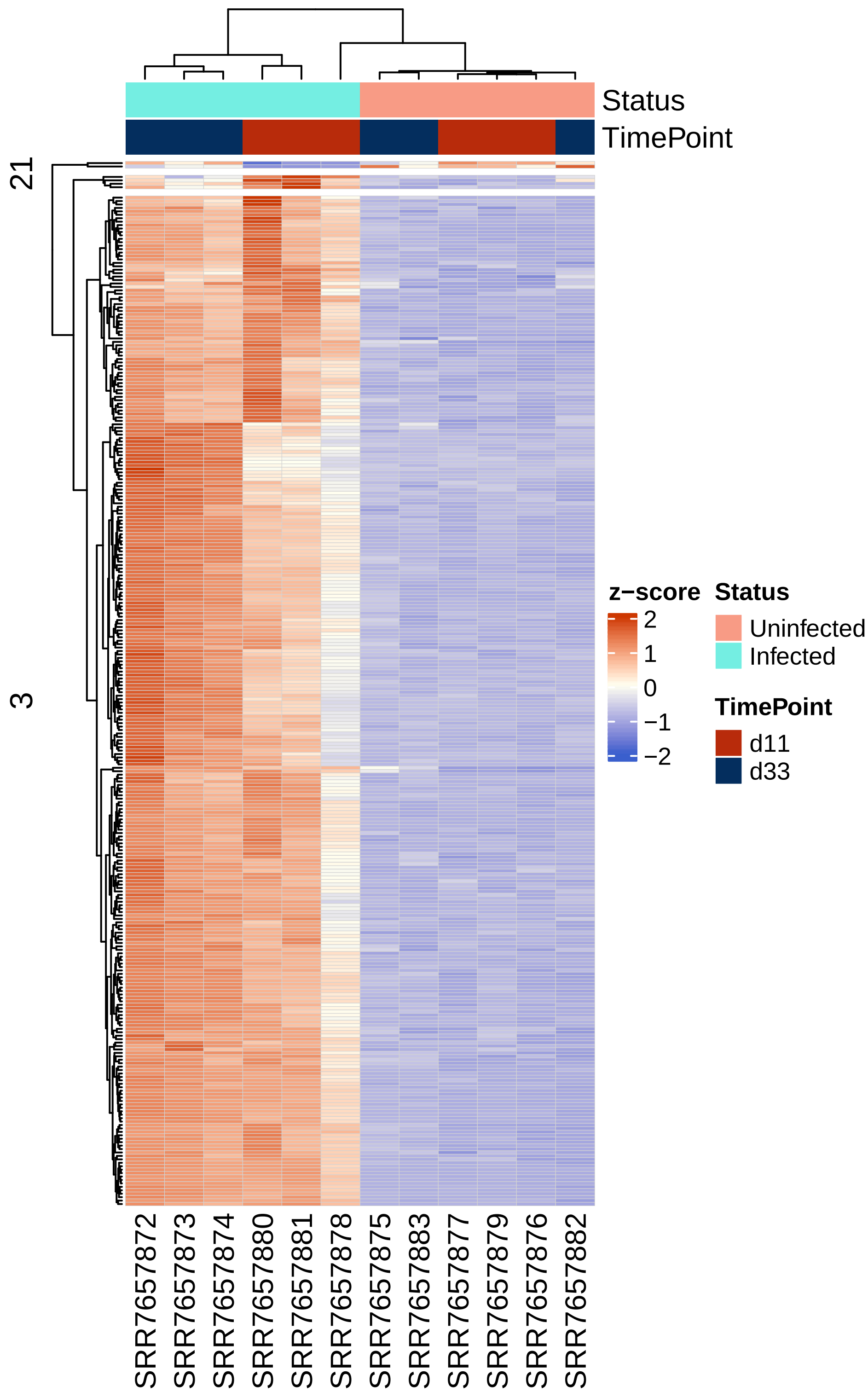


we can also split the heat map into clusters and add some annotation.

```
ha1 = HeatmapAnnotation(df = colData(ddsObj.interaction)[,c("Status", "TimePoint")])

Heatmap(z.mat, name = "z-score",
```

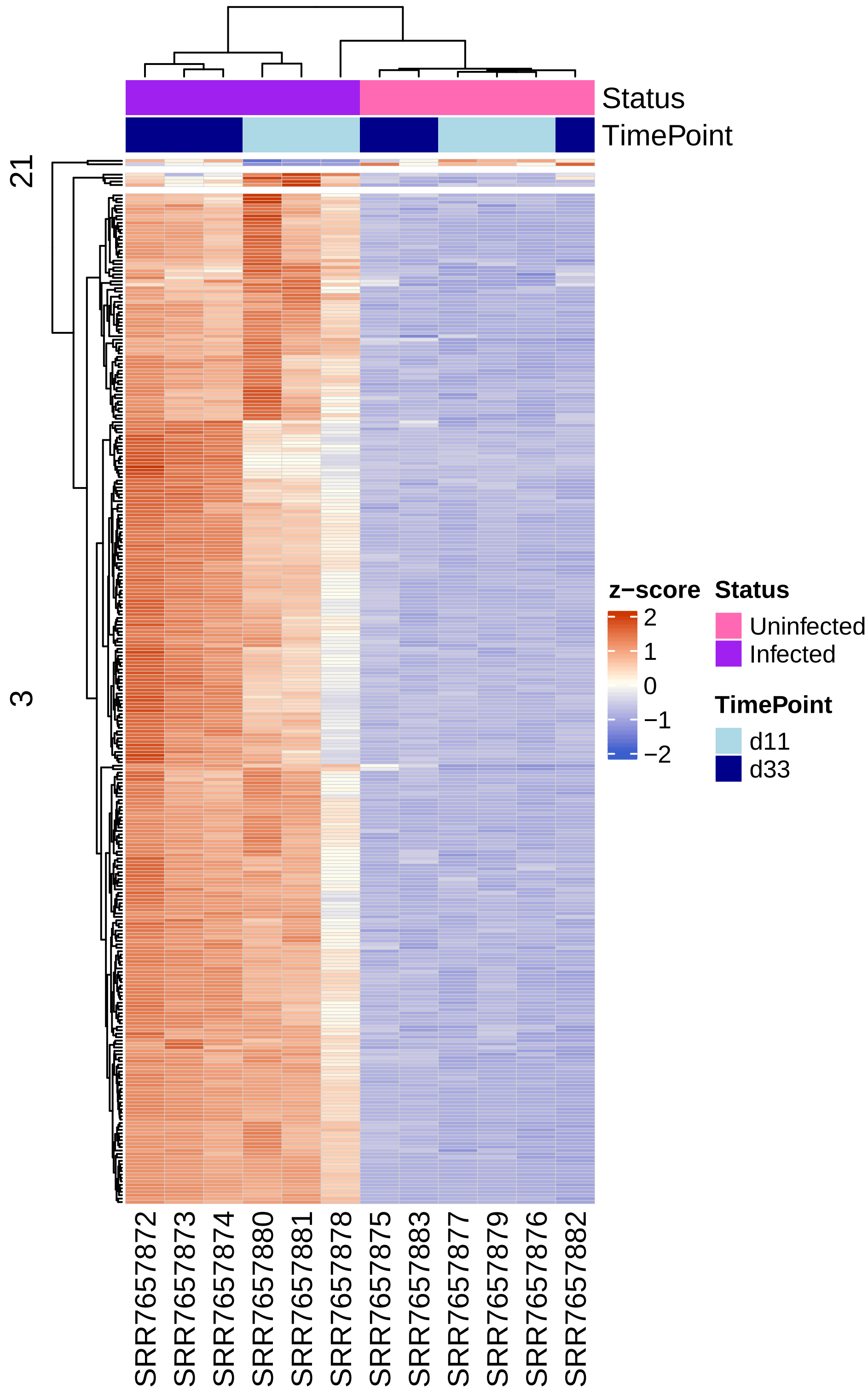
```
        col = myRamp,
        show_row_name = FALSE,
        split=3,
        rect_gp = gpar(col = "lightgrey", lwd=0.3),
        top_annotation = ha1)
```



Whenever we teach this session several student always ask how to set the colours of the bars at the top of the heatmap. This is shown below.

```
ha1 = HeatmapAnnotation(df = colData(ddsObj.interaction)[,c("Status", "TimePoint")], col = list(Status =
```

```r
Heatmap(z.mat, name = "z-score",
        col = myRamp,
        show_row_name = FALSE,
        split=3,
        rect_gp = gpar(col = "lightgrey", lwd=0.3),
        top_annotation = ha1)
```




```
saveRDS(annot.interaction.11, file="results/Annotated_Results.d11.rds")
saveRDS(shrinkTab.11, file="results/Shrunk_Results.d11.rds")
saveRDS(annot.interaction.33, file="results/Annotated_Results.d33.rds")
saveRDS(shrinkTab.33, file="results/Shrunk_Results.d33.rds")
```