BMS397

LABORATORY RESEARCH PROJECT

SEMESTER 1, 2020-21

Research Performance

+ Lab Book

Please fill in your details below for the coversheet

Name: Harry Petch-Smith

Title: Identification of Cyp21a2-dependent, glucocorticoid regulated target genes in zebrafish larvae

Supervisor: Vincent Cunliffe

By submitting this work, I agree to abide by the University of Sheffield rules on plagiarism, collusion and the use of unfair means in assessment.

https://www.sheffield.ac.uk/ssid/unfair-means/index

Identification of Cyp21a2-dependent, glucocorticoid regulated target genes in zebrafish larvae

I am using an R markdown file as my lab book as it will allow me to store my code, figures and notes all in one place. I will outline the steps of my analysis before I perform it and note down any observations. When I am done, I will convert this to a word document using the knit to word function for submission.

I've separated the markdown into headings which give the dates that I did each part of the analysis.

Clear the global environment

I've included this becuase my code wasnt working and this command fixed it. It could be that a previous operation is confliting with what I was trying to do, runnning this console command resets all variables, allowing you to run the whole thing again from fresh without having to restart R studio. I've left it at the top for convenience.

```
rm(list = ls())
```

Documentation for packages

This section is updated every time I use a new package. It will include a list of links to useful resources on how to use each package in case i come across any problems in my analysis

• readr: https://www.rdocumentation.org/packages/readr/versions/1.3.1

• Deseq2: https://www.bioconductor.org/packages/devel/bioc/vignettes/DESeq2/inst/doc/DESeq2.html

• ggplot2: https://ggplot2.tidyverse.org/ Note: not technically documentation but does contain a useful cheat sheet

•Pheatmap:https://www.rdocumentation.org/packages/pheatmap/versions/1.0.12/topics/pheatmap

• Dplyr: https://www.rdocumentation.org/packages/dplyr/versions/0.7.8

16/10/2020 Preprocessing of the count data

I retrieved the count data from my supervisor in the form of multiple csv files, each containing the raw count data for each sample and the ensembl ids of the genes. The filenames where as follows:

readspergene_cyp21_2.out.xlsx readspergene_cyp21_1.out.xlsx
readspergene_cyp21_3.out.xlsx readspergene_cyp21HC_2.out.xlsx
readspergene_cyp21HC_3.out.xlsx readspergene_cyp21HC_1.out.xlsx
readspergene_WT_1.out.xlsx readspergene_WT_2.out.xlsx *readspergene_WT_4.out.xlsx*
readspergene_WTHC_1.out.xlsx *readspergene_WT_3.out.xlsx*
readspergene_WTHC_2.out.xlsx *readspergene_WTHC_3.out.xlsx

I copied and pasted the count data from each of these files into a new excel file which was named counts.xlsx. I then converted this file to .txt format so that it could be read into R as a table.

Finally I created a metadata file containing the genotype and treatment of each sample which I named metadata.txt. The data is now ready to be imported into R.

17/10/2020 Importing the Data into R

Today I am importing the data into R studio for analysis, and creating a dataset using deseq. I will use the readr package to read in the count data and metadata from the .txt files mentioned above. I have added the readr documnetation to my list above. The deseq package will then transform and combine this data into one single object, which saves a lot of time and effort. I stored the raw counts data in a data frame called counts and my

metadata in a dataframe called sampleinfo. Once again i have added the deseq package documentation to my list above.

```
#Initialise the readr library
library(readr)

#Read in the count data
counts = read.table("Raw data/counts.txt", header = TRUE)



#Read in the metadata
sampleinfo = read.table("Raw data/metadata.txt", header = TRUE)
```

Creating the Deseq dataset

To allow for analysis my count data and metadata must be combined into a single object. This could be done using base R however the process would be very long and painful. Luckily there is an R package offered by bioconductor that allows for the analysis of RNAseq data quickly. This package is known as DEseq2 and can be used to combine my data into a single object, containing the count data and sample information from the study.

The DESeqDataSetFromMatrix() function takes the count matrix (the count data table) and the metadata (information about each sample) and combines them into one dataset. The design is set to the experimental design of the study, i.e., what condition we are comparing between samples. I set it to genotype as i wish to compare the mutant and wild type samples first.

In order to check that my sample information had been assigned correctly i used the colData function to check that each sample had the correct treatment and genotype assigned to them.

```
#Initialise the deseq2 package
library(DESeq2)
```

```
#Combine metadata and count data into single dds object
dds <- DESeqDataSetFromMatrix(countData = counts,colData = sampleinfo, design
= ~Genotype)

#Checking that metadata has been correctly assigned
colData(dds)

## DataFrame with 13 rows and 2 columns
##                Genotype   Treatment
##                <factor> <character>
## WT1                 WT    untreated
## WT2                 WT    untreated
## WT3                 WT    untreated
## WT4                 WT    untreated
## cyp21_MUT1          MUT   untreated
## ...                 ...         ...
## cyp21_MUT.HC2       MUT     treated
## cyp21_MUT.HC3       MUT     treated
## WT.HC1              WT      treated
## WT.HC2              WT      treated
## WT.HC3              WT      treated
```

After inspecting the colData property of the dds object it is apparent that the sample id column did not have a name. This is due to the fact that deseq automatically imports the sample names from the csv file as rownames in the deseq dataset. This may make analysis difficult if I wish to select samples based on their name, so I wrote a code block that creates a sample id column, with all the sample ids and adds this to the deseq dataset.This uses the cbind function which allows you to create a new column and assign values to the subsequent rows. I learned how to do this through reading the R documentation for cbind(): https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/cbind

```
#Create a new column in the colData item corrosponding to the rownames
colData(dds) <- cbind(Name = rownames(colData(dds)), colData(dds))
```

```
#Inspect results
colData(dds)

## DataFrame with 13 rows and 3 columns
##              Name Genotype   Treatment
##       <character> <factor> <character>
## 1             WT1      WT    untreated
## 2             WT2      WT    untreated
## 3             WT3      WT    untreated
## 4             WT4      WT    untreated
## 5      cyp21_MUT1      MUT   untreated
## ...           ...      ...         ...
## 9   cyp21_MUT.HC2      MUT     treated
## 10  cyp21_MUT.HC3      MUT     treated
## 11         WT.HC1      WT      treated
## 12         WT.HC2      WT      treated
## 13         WT.HC3      WT      treated
```

The rownames are now numbers which I can't get rid of however I will be able to use the Name column to select samples, therefor the row numbers are irrelevant and shouldnt interfere with any analysis (this remains to be seen).

Future research has revealed that I could've just used the rownames() function to select the samples however I prefer to have them in a named column as it makes the data easier to read in my opinion.

21/10/2020 Quality analysis of the data

In order to check that samples are labelled correctly and that the count data follows a normal distribution I must first conduct quality analysis of the count data.

In order to visualise the distribution of count data it must first be transformed to account for large differences in counts that might skew the data. deseq2 has two inbuilt functions to do this: rlog() and vst(). Both of these do essentially the same thing; they both conduct a

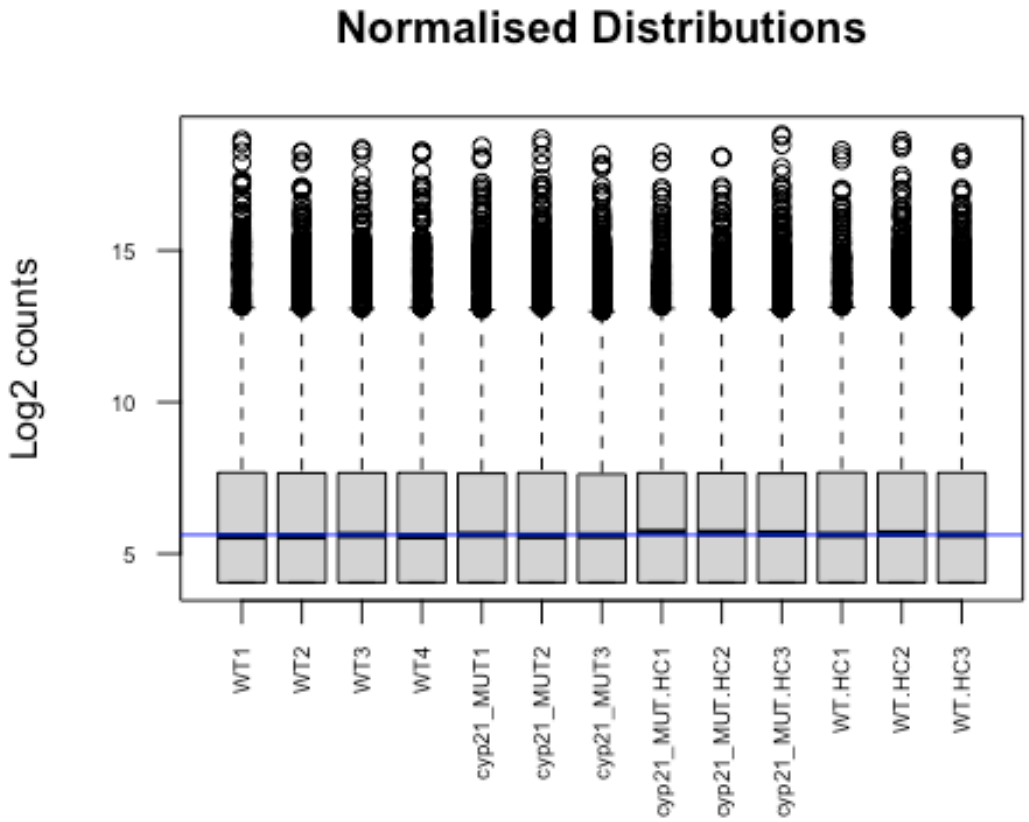
log2 transformation on the data. I've seen a lot of people in online threads saying that vst is better and faster for large datasets, so i will use that.

```
#Transform the data
vsd = vst(dds, blind = TRUE)
```

Boxplot of normalised count data

I used a boxplot to assess the distribution of the transformed count data. The blue line represents where the medians pf each sample lines up. Initially the sample labels on the x axis were cut off so I had to resize the figure to compensate for this. This blog proved a very useful resource when learning how to do this: http://zevross.com/blog/2017/06/19/tips-and-tricks-for-working-with-images-and-figures-in-r-markdown-documents/

```
par(cex.axis=0.6)
boxplot(assay(vsd), ylab="Log2 counts",las=2,main="Normalised Distributions",
names = vsd$Name)

# Added a blue horizontal line that corresponds to the median logCPM
abline(h=median(assay(vsd)), col="blue")
```



The medians of all the data appear to line up aside from some slight variability in the mutant treated samples, two of which show medians slightly above the line. as a result I can conclude that these data have been normalized successfully.

Since my data is normalised I can conduct the other quality analysis tomorrow.

22/10/2020 Principle componant analysis

In order to visualise the relative similarity of samples I can use a PCA plot. This compares samples based on their two principle components, and plots their similarity on a two dimensional grid, relative to all other samples in the study. Samples that are similar will likely form clusters together within a quadrant of the graph. IN a perfect world we should expect significant overlap between results of the same group however there is likely to be

some biological variability that impacts this. As long as there's no distinct crossover between groups then I can conclude the samples are correctly labelled.

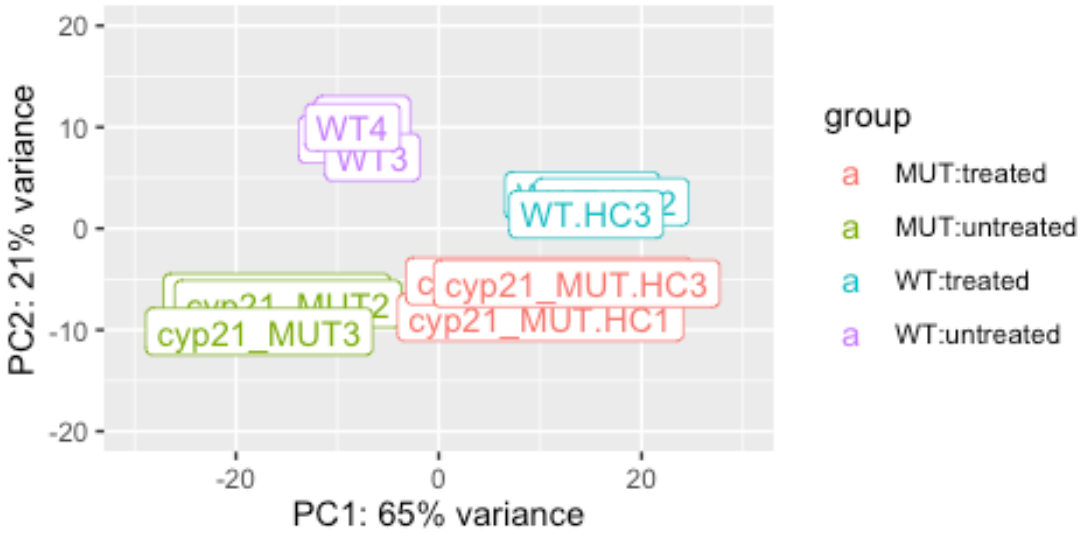
In order to create the plot I am using two packages: deseq2 and ggplot. deseq2 has a function dedicated to making pca plots of genetic data, which i will use to make the plot itself. I can then use ggplot to modify this graph and make modifications to make it easier to visualise the data.

My supervisor taught me how to make a pca plot for one of my comparisons, however after looking at the documentation from deseq2 it appears that I can define an interest group as a vector that covers all sample conditions defined in the colData (metadata). My pca plot will therefor be one plot that covers all samples in the study.

I generated the plot with the interest groups being treatment and genotype in order to gain an overview of how each group of samples was different. I limited the X and Y scales after experimenting to see which settings gave the best sized graph where all the data was visible. I used the figure.heading and figure.height parameters in the chunk options to resize the graph to a suitable size ensuring visibility.

gglpot was used to turn each data into a sample label.

```
#Initialise the ggplot2 library
library(ggplot2)



#Create the pca plot, intgroup is a vector with genotype and treatment
plot = plotPCA(vsd, intgroup = c("Genotype","Treatment"))

#print the plot with sample information and scale modifications using ggplot2
plot + geom_label(aes(label = vsd$Name)) +  coord_fixed(xlim = c(-30,30),
ylim = c(-20,20))

## Coordinate system already present. Adding new coordinate system, which
will replace the existing one.
```

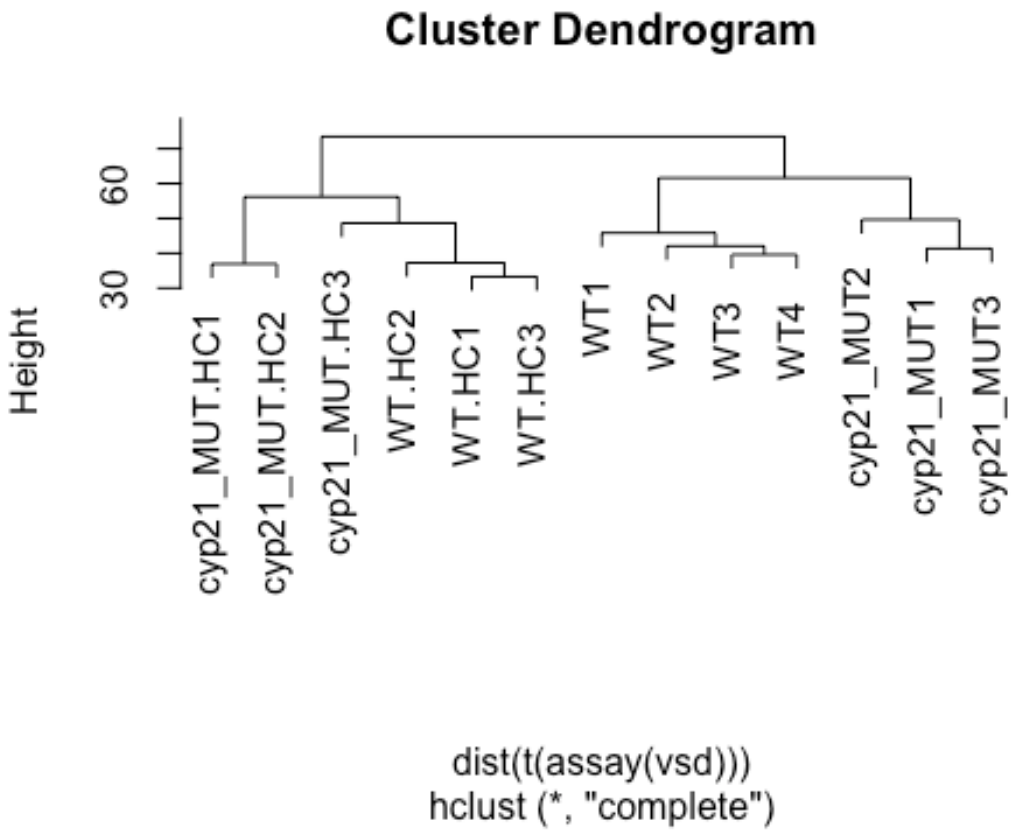


The PCA plot shows distinct sample clustering based on genotype and treatment group. It appears the wildtype treated and mutant treated samples are more similar to each other than their non treated counterparts. There appears to be most variation within the mutant treated group, with CYP_MUT.HC1 being seperated from the other two samples. These small differences are likely due to biological variability within the samples and are unlikely to effect results.

cluster dendorograms are another method of assessing similarity. These provide a tree with branches connecting samples that are related i.e more similar. Hopefully the samples will branch based on their genotype and treatment group, indicating differences in gene expression between these factors.

Sample clustering analysis

```
#PLot Dendogram
plot(hclust(dist(t(assay(vsd)))), labels = vsd$Name)
```



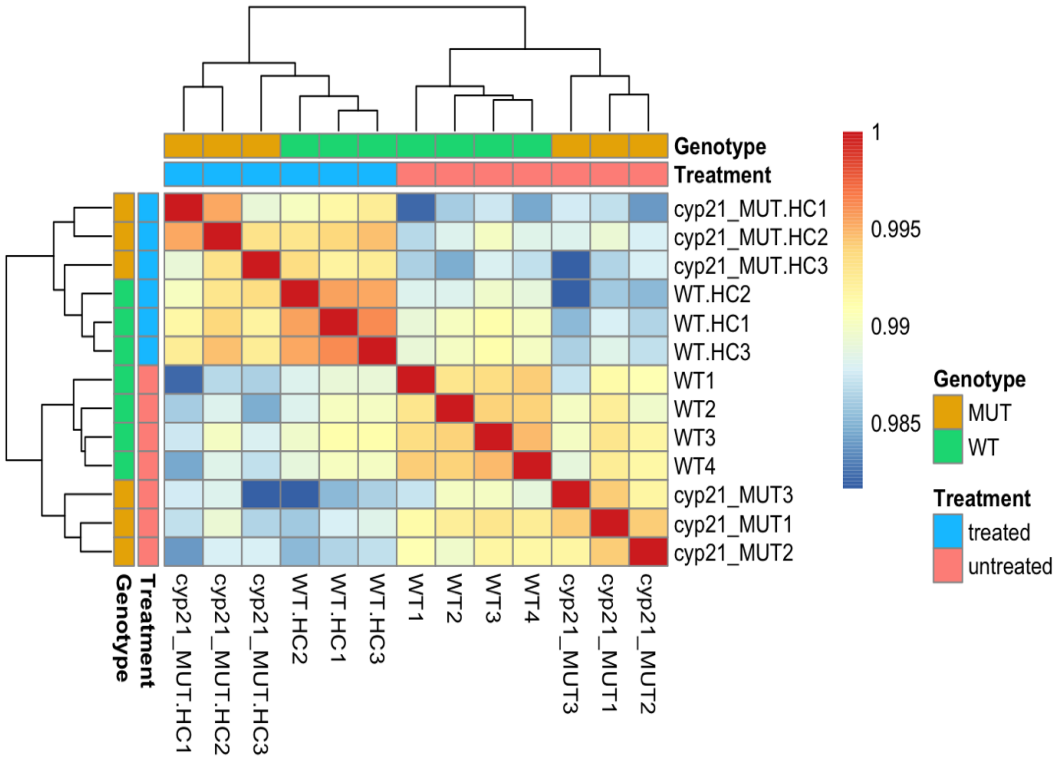
The cluster dendogram shows that there is a large amount of similarity within groups apart from the mutant treated samples where CYP_MUT.HC3 is shown to be slightly more similar to the wild type treated samples.

Cluster heatmap

In order to get a more quantitative overview of sample similarity a heatmap can be generated from the correlations in the count data. In order to do this I used the pheatmap library which allows for generation of heatmaps. The heatmap will compare each sample to one another, with the degree of redness of each tile showing sample simlarity. I should expect to see deep red where samples of the same group are compared and deep blue where samples of differing groups are compared. As a result the heatmap should display clusters of red and blue. I used the metadata mentioned earlier to annotate the columns and rows of the heatmap based on the genotype an treatment of each sample.

```r
#Initialise the pheatmap library
library(pheatmap)

#Retrieve the metadata for annotation
sampleinfo = as.data.frame(colData(dds)[,c('Treatment','Genotype')])

#Retrieve normalised count data
assay = assay(vsd)

#Calculate correlations
assay_cor <- cor(assay)



# 1. Open jpeg file
png("Figures/qa heatamp", width = 700)
# 2. Create the plot
#Plot correlations on annotated heatmap
pheatmap(assay_cor,
         annotation = sampleinfo,
         annotation_row = sampleinfo,
         labels_row = vsd$Name,
         labels_col = vsd$Name)
# 3. Close the file
dev.off()
```

```
## quartz_off_screen
##                 3
```



The heatmap shows clustering where similar groups are compared. It appears there is a large amount of similarity between wild type and mutant treated samples, and a greater difference between the two genotypes when untreated. There also appears to be a large amount of variability within the mutant untreated group, which indicates biological differences in gene expression between the samples.

27/10/2020 Differential expression analysis: experimental design

The aims of the study are to identify cyp21a2 dependant glucocorticoid regulated gened and to assess the effects of hydrocortsione on the mutant phenotype. In order to assess this I will make multiple pair wise gene expression comparisions:

WT V MUT to analyse the success of mutation and its effect on gene transcription

MUT treated V MUT untreated to assess the effects of the treatment

WT V MUT treated to assess how different the treated mutants are from the norm

WT treated V WT untreated to assess the efffects of elevated glucocorticoid signalling on gene transcription

I then aim on visualising this data with methods such as heatmaps and volcano plots.

4 comparisons may be too many to talk about in my final report and if that's the case then i will pick out the most interesting ones.

In order to analyse this data i will need to make subsets of the dds object i created earlier corrsponding to each of the comparisons that i wish to make. Once i have figured out how to do this i can conduct my analysis and visualise the data

###Subsetting the deseq dataset After reading through the documentation for the deseq2 package It seems i can subset the data by gnerating different datasets corrosponding to each pairwise analysis from my original dds object. It is a simple case of retrieving the counts for each sample in the group that i wish to comapre and creating a new deseqdataset containing just that information. Once i have this I can run the deseq() function which will transform and analyse my data, calculating the foldchange in gene expression and assigning statistical significance.

If i cant get the subsetting to work an alternative method would be to maunally copy and paste the pairwise data into individual csv files, and then read that data into R seperately, which was recommended by dr cunliffe. However I wish to try subsetting the data in R first as it will ensure that no human error occured through munual copying and pasting of the data. If i cant get this to work then i will manually copy and paste the data into excel files.

WT untreated vs mutant untreated

The following should successfully create my first subset (WT V MUT)

```
#Retrieve counts for untreated samples
dds_genotype = dds[,which(dds$Treatment == "untreated")]

#Set the design to compare expression between genotypes
design(dds_genotype) = ~Genotype

#Use DESeq2 to analyse data
de_genotype = DESeq(dds_genotype)
```

```
## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

#Generate results, using MUT as baseline for log fold change
genotype_results = results(de_genotype, c('Genotype','MUT','WT'))
```

The deseq function calculates log fold change in gene expression. in this case since i set the comparison to mutant vs wt it will calculate the change in mutant to wild type. A positive fold change indicates a greater expression in the mutant, wilst a negative fold change indicates decreased expression. Deseq also calculates the adjusted p value, which gives indication of statistical significance. The lower this is the more significant the result.

Since this successfully subsetted the data, and i now have an object corresponding to just the mutant and wildtype untreated samples, I will now create subsets for the rest of the data.

Mutant treated vs mutant untreated

When changing the design to compare between treatment groups I came accross a problem. When deseq read in the sample info it assigned the treatment column to be a character vector as opposed to a factor. In order to use the Treatment as a design I need to convert it to a factor which I can do using the as.factor() function which i learned how to use here:
https://www.rdocumentation.org/packages/h2o/versions/2.4.3.11/topics/as.factor

```
#Convert the Treatment character vector to factor
dds$Treatment = as.factor(dds$Treatment)
```

I can now subset the data, this time using Treatment as the design.

```
#Retrieve counts for mutant samples
dds_muttreatment = dds[,which(dds$Genotype == "MUT")]

#Set the design to compare expression between treatments
design(dds_muttreatment) = ~Treatment

#Use DESeq2 to analyse data
de_muttreatment = DESeq(dds_muttreatment)

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

#Store results
mutreatment_results = results(de_muttreatment, contrast =
c('Treatment','treated','untreated'))
```

Mutant treated vs WT

Subsetting this data is a bit more tricky. My which() function needs logic to determine which samples are both mutant and treated and samples that are WT and untreated. In order to do this I created two vectors, one selecting mutant treated samples and one containing WT untreated samples. This was done using the and (&) and or (|) logical operators.

```
#Retrieve counts for mutant treated and wildtype untreated samples
dds_mutreatedwt = dds[,which(c(dds$Genotype == "MUT" & dds$Treatment ==
'treated') | c(dds$Genotype == 'WT' & dds$Treatment == 'untreated'))]
```

```
#Set the design to compare expression between treatments
design(dds_mutreatedwt) = ~Treatment


#Use DESeq2 to analyse data
de_mutreatedwt = DESeq(dds_mutreatedwt)

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

#Store results
muttreatedwt_results = results(de_mutreatedwt,
c('Treatment','treated','untreated'))
```

WT treated vs WT unntreated

This is just a simple case of seperating the WT data into a subset as i did before with the mutant samples

```
#Retrieve counts for mutant samples
dds_wt = dds[,which(dds$Genotype == "WT")]


#Set the design to compare expression between treatments
design(dds_wt) = ~Treatment


#Use DESeq2 to analyse data
de_wt = DESeq(dds_wt)

## estimating size factors
```

```
## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing

#Store results
wt_results = results(de_wt, contrast = c('Treatment','treated','untreated'))
```

Now that I've subsetted the data i can move on to visualisation, which i will do another day.

28/10/2020 Processing results for visualisation

Now that ive generated my results theres a few things I need to do before graphing them. First I need to assign external gene names to my results to make differentailly expressed genes easily identifiable by their name as opposed to a long ensembl id, which makes it easier to label them (and google them).

In order to add gene symbols I need a database containting gene symbols matched to ensembl ids. For this i used the biomart package, which is a library of all gene names matched to ensembl id for many different organsims. The gene symbols will need to be matched to the ensembl ids in my results and joined to each results table.

First i will retrieve the gene symbols from biomaRt:

```
#Initialise database with gene symbols
library(biomaRt)


#retrieve data for zebrafish
ensembl = useMart("ensembl",dataset="drerio_gene_ensembl")
```

Each of my results objects does not contain a column for ensembl id. instead the ensembl id of each gene is a row name. This is a problem as i wont be able to easily select the ensembl

ids of each gene to match with those in the external database. I will have to therefor assign a new column to each of my results tables containing the ensembl ids of the genes. I did this using the sapply function to create a new column named “ensembl” that contained the rownames of the table and therefor the ensembl ids of each gene.

```
#Create new columns for ensmbl id in each results table
genotype_results$ensembl <- sapply( strsplit( rownames(genotype_results),
split="\\+" ), "[", 1 )
mutreatment_results$ensembl <- sapply( strsplit(
rownames(mutreatment_results), split="\\+" ), "[", 1 )
muttreatedwt_results$ensembl <- sapply( strsplit(
rownames(muttreatedwt_results), split="\\+" ), "[", 1 )
wt_results$ensembl <- sapply( strsplit( rownames(wt_results), split="\\+" ),
"[", 1 )
```

All the gene symbols and ensembl ids are stored in the ensembl object. In order to match these to my results I need to create a genemap object that retrieves the ensembl id and external gene names that match with the ensembl ids in my results. since all subsets tested for the same genes i can use any of them to match the emsmbl ids and create the gene map.

```
#Generate genemap with ensembl ids and external gene names.
genemap = getBM (attributes = c("ensembl_gene_id","external_gene_name"),
                 filters = "ensembl_gene_id",
                 values = wt_results$ensembl,
                 mart = ensembl)
```

The genemap object now contains a table with the ensembl id from my results and the corrosponding external gene id from the bioMart database. I now need to join this to my original results tables individually.

```
#Match gene symbols with ensembl ids from results
idx = match(wt_results$ensembl, genemap$ensembl_gene_id)

#Join the gene id table to each of my results tables
wt_results$external_gene_name = genemap$external_gene_name[idx]
```

```
genotype_results$external_gene_name = genemap$external_gene_name[idx]
mutreatment_results$external_gene_name = genemap$external_gene_name[idx]
muttreatedwt_results$external_gene_name = genemap$external_gene_name[idx]
```

Now that I have my results for each pairwise comparison I can save them to csv files to store the data and make sure I dont have to run all of the above code again in the future. Instead i can simply read the data from these files whenever i need it. Furthermore it means i have a permanant copy of my results, reducing the chances of me losing them.

In order to convert the data to a csv file i can use the readr write.csv() function. My data has to be in the form of a data frame in order to do this so I converted it using as.data.frame(). I also arranged the data by padj in order to see which results where the most significant.

The directories used to store the results can be seen in the code below.

```
library(dplyr)

#Write wt results to a file
write.csv(arrange(as.data.frame(wt_results), padj),file =
"Results/wt_results.csv")


#Write genotype results to a file
write.csv(arrange(as.data.frame(genotype_results), padj),file =
"Results/genotype_results.csv")

#Write the mutated treatment results to a file
write.csv(arrange(as.data.frame(mutreatment_results), padj),file =
"Results/Mutant_treatment_results.csv")

#Write the mutant treated vs wt results to a file
write.csv(arrange(as.data.frame(muttreatedwt_results), padj),file =
"Results/Mutant_treated_vs_wt_results.csv")
```

29/10/2020 Visualising the data

Now i need to visualise the data for each of my pairwise analysis

29/10/2020 Mutant untreated vs wildtype untreated samples

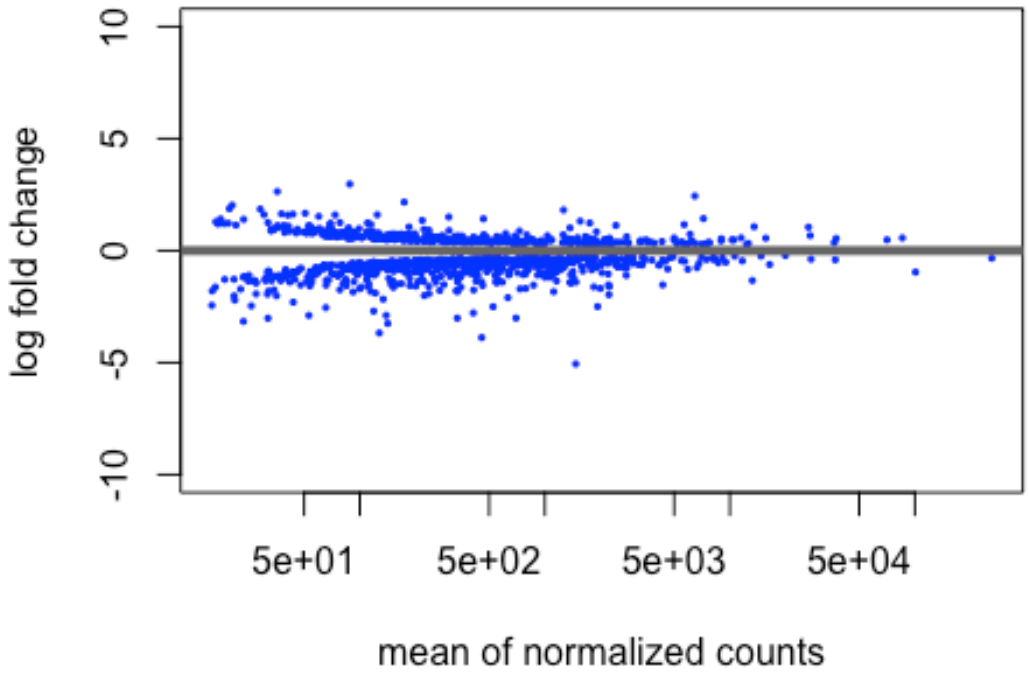
First i need to shrink the log fold changes of my results in order to properly visualizes the data without any extreme values interfering. In order to do this i will use the deseq2 lfcshrink function.

```
#Shrink the log fold changes
shrink_genotype = lfcShrink(de_genotype, contrast = c('Genotype','MUT','WT'),
res = genotype_results, type = 'normal')

## using 'normal' for LFC shrinkage, the Normal prior from Love et al (2014).
##
## Note that type='apeglm' and type='ashr' have shown to have less bias than
type='normal'.
## See ?lfcShrink for more details on shrinkage type, and the DESeq2
vignette.
## Reference: https://doi.org/10.1093/bioinformatics/bty895
```

In order to gain an overview of my data i can generate an ma plot. This gives an overview of the distrvution of my data, i.e whether there is more overall upregulation or downregulation of genes.

Ma plot of mutant vs wildtype

```
sig_genotype = subset(genotype_results, padj < 0.05)


plotMA(sig_genotype,ylim = c(-10,10))
```



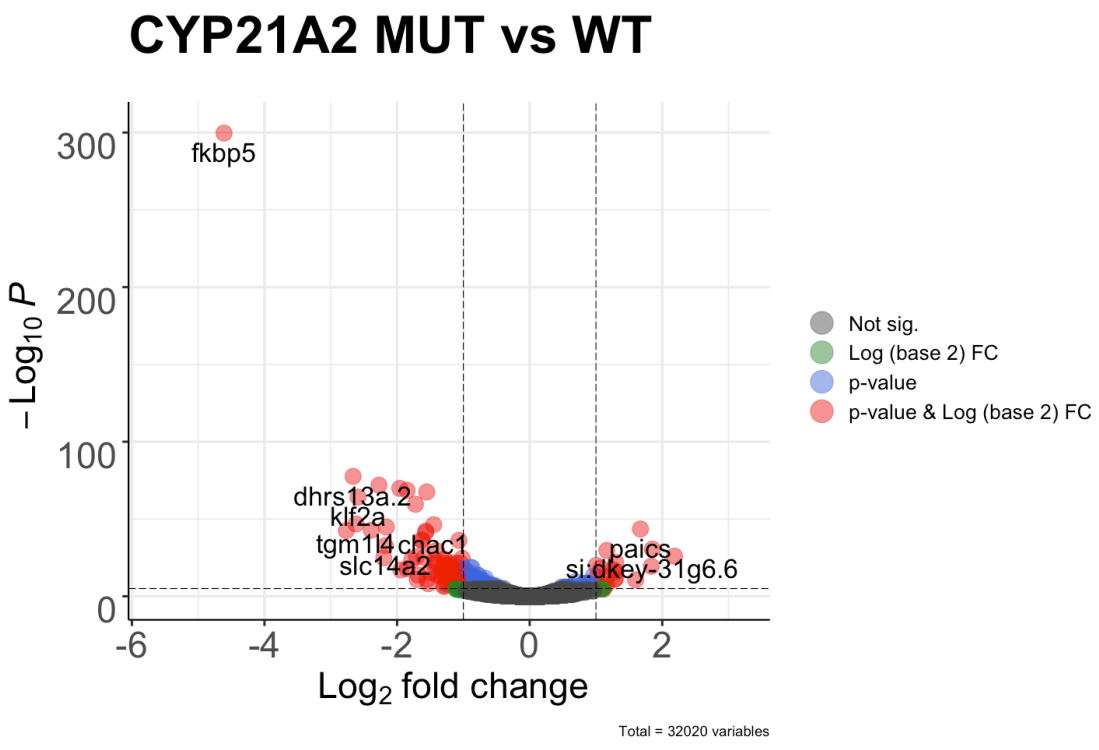
The Ma plot gives me an overview of the differential expression. the ma plot shows that more genes where Downregulated in the wildtype compared to the mutants

Next I created a volcano plot which plots the log fold change against the p value of each gene. Genes with a pvalue of less than 0.05 and a fold change greater than +1 or -1 are highlighhted in red. I set the code to label the top 30 most significant genes where possible, however some of the datapoints are so close that a distinct label cannot be applied. Once again i resized the figure using the chunk settings.

Volcano plot of mutant vs wildtype

```
shrink_genotype = arrange(as.data.frame(shrink_genotype),padj)
symb = shrink_genotype$external_gene_name[1:30]
library(EnhancedVolcano)

## Loading required package: ggrepel
```

```
EnhancedVolcano(shrink_genotype,
                lab = shrink_genotype$external_gene_name,
                selectLab = c(symb),
                labSize = 9.0,
                pointSize = 7.0,
                title = "CYP21A2 MUT vs WT",
                axisLabSize = 35,
                titleLabSize = 50,
                subtitle = " ",
                x ='log2FoldChange',
                y ='padj',
                legendLabels=c('Not sig.','Log (base 2) FC','p-value',
                               'p-value & Log (base 2) FC'),
                legendPosition = 'right',
                legendLabSize = 20,
                legendIconSize = 10)
```



The plot shows that more genes where downregulated in the mutant compared to upregulated. The

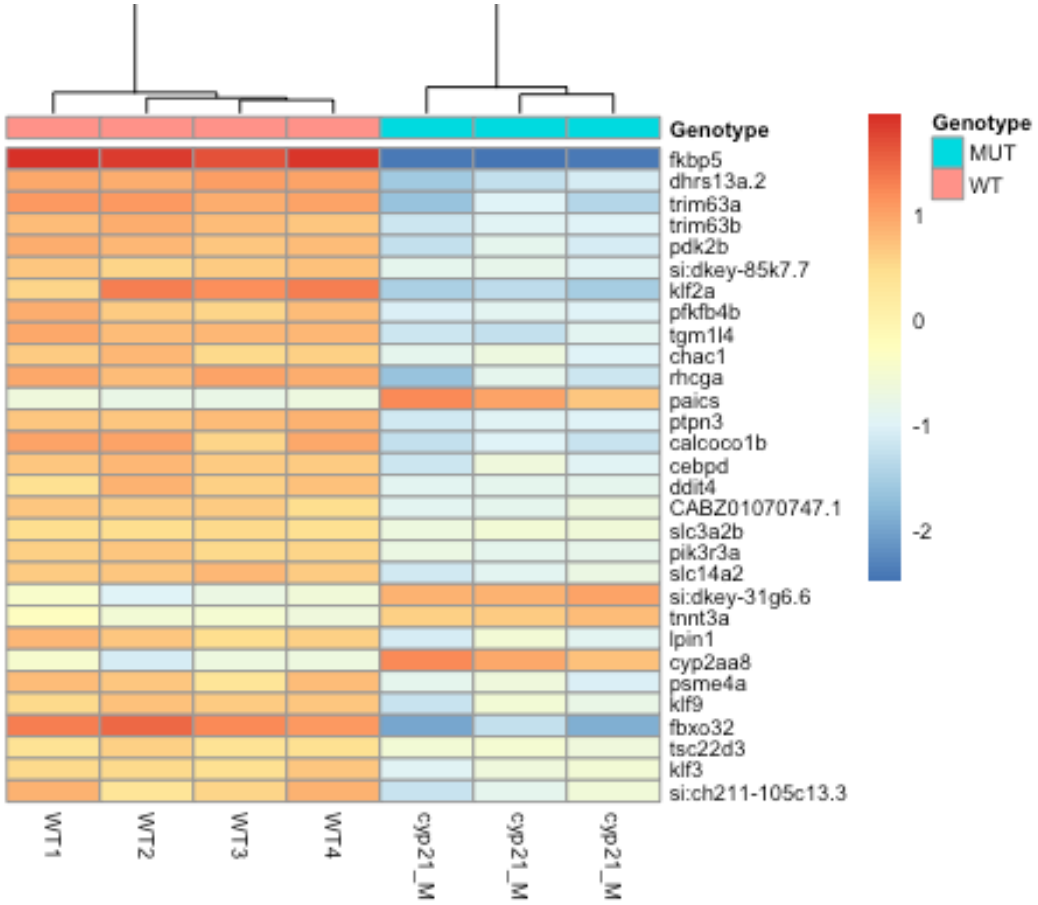
gene fkbp5 is clearly the most downregulated gene and shows the highest statistical significance. Paics was the most significant upregulated gene in the mutants.

I can also create a heatmap based on the read data for each gene in order to give a closer look at the differences between expression. I created a matrix containing the top 30 most significantly expressed genes which I then created the heatmap from. I used the sampleinfo object to label the genotypes of the samples and annotated each row with the gene symbols.

I used the symb variable from earlier to label each row with the gene id.

```
#Initialise pheatmap if not done previously
library(pheatmap)

#log2 transformation
vsd_genotype = vst(dds_genotype)

#Create matrix containing the top 30 most significant genes
genotype_mat = assay(vsd_genotype)[ head(order(genotype_results$padj),30), ]

#Subtract the mean reads for each gene
genotype_mat = genotype_mat - rowMeans(genotype_mat)

#Read in genotypes for annotation
sampleinfo = as.data.frame(colData(dds)[c('Genotype')])



#Plot heatmap
pheatmap(genotype_mat,
         fontsize = 7,
         cellwidth = 30,
         cellheight = 7,
         labels_row = symb,
         annotation = sampleinfo,
```

```
        cluster_rows = F,
        clustering_distance_rows = F,
        labels_col = dds_genotype$Name)
```



The heatmap provides more insight into the results behind the volcano plot. It shows a significant difference between reads for fkbp5 in the WT and Mut samples. Of the top 30 genes only 4 where upregulated in the mutants, the rest all show quite s significant downregulation.

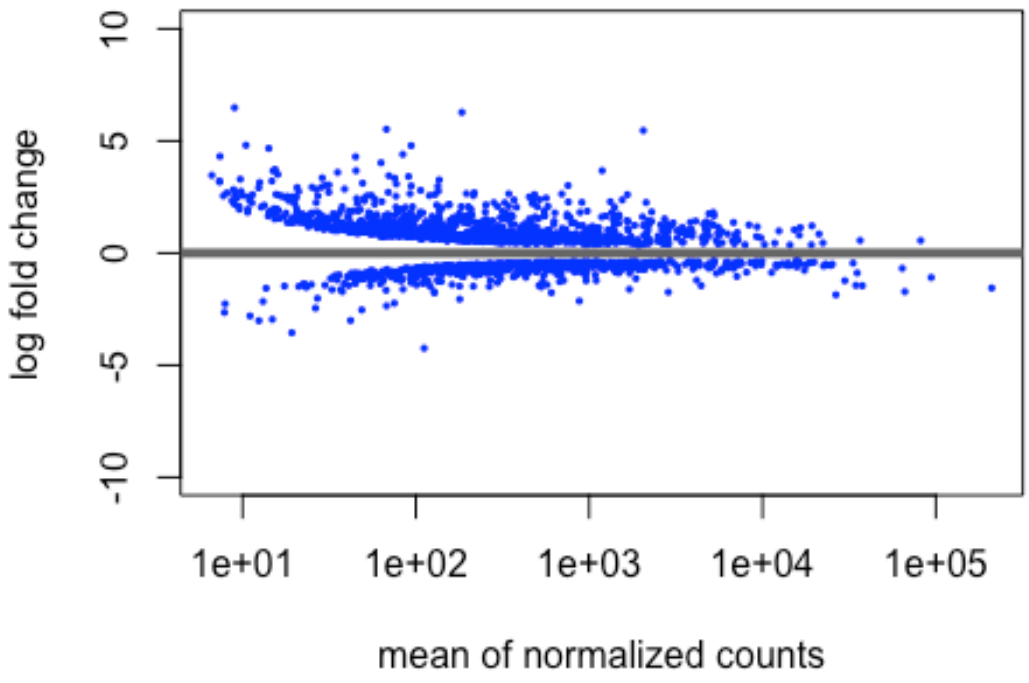
Now that I have a basic set of graphs i can repeat this process for the other comparisons.

01/11/2020 Mutant treated vs untreated

Ma plot of mutant treated vs untreated

The Ma plot shows treated vs untreated mutant samples. those with a negative fold change are underexpressed in the treated group and those with a positive fold change where upregulted in the treated group. The plot shows that more genes where upregulated than downregulated in the treated group.

```
#Subset top 30 most sig results
sig_muttreatment = subset(mutreatment_results, padj < 0.05)

#create plot
plotMA(sig_muttreatment, ylim = c(-10, 10))
```



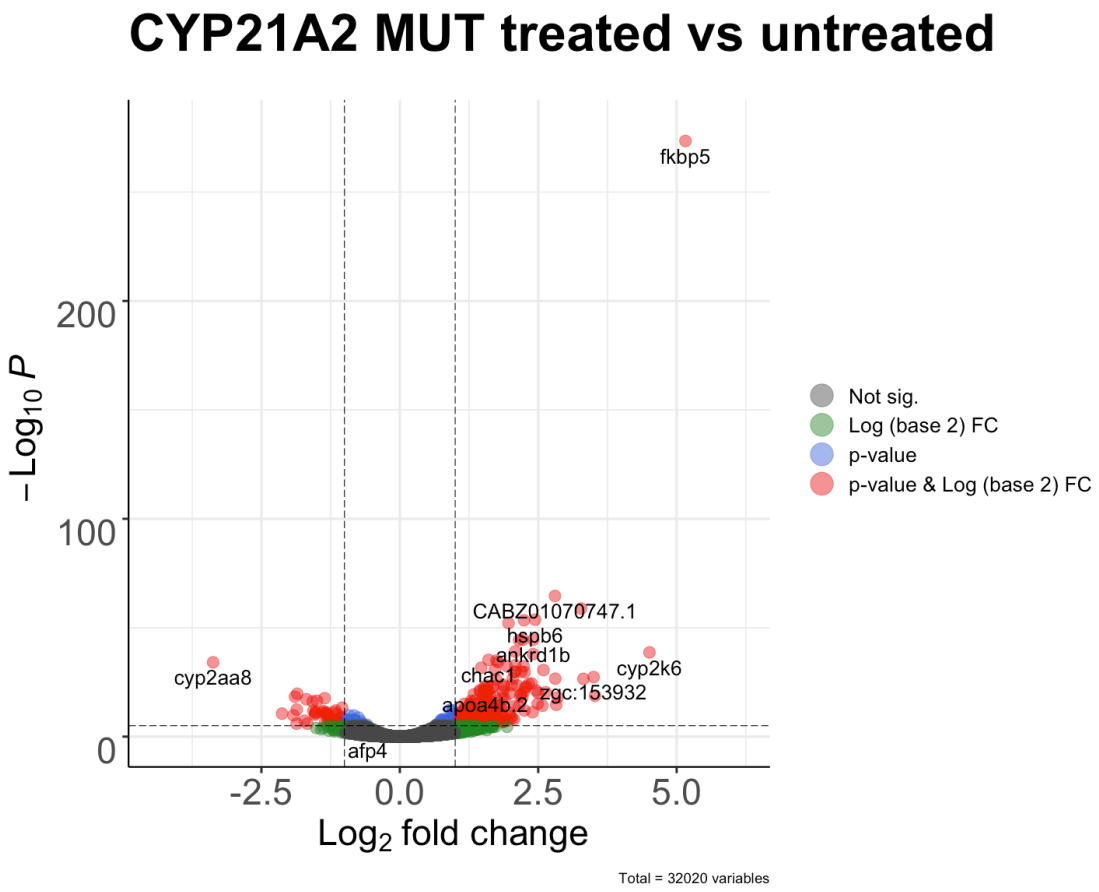
Next i produced a volano plot. As before i performed an lfc shrink on the data

```
#Shrink lfc
shrink_muttreatment = lfcShrink(de_muttreatment, contrast =
c('Treatment','treated','untreated'), res = mutreatment_results, type =
'normal')

## using 'normal' for LFC shrinkage, the Normal prior from Love et al (2014).
##
## Note that type='apeglm' and type='ashr' have shown to have less bias than
type='normal'.
## See ?lfcShrink for more details on shrinkage type, and the DESeq2
vignette.
## Reference: https://doi.org/10.1093/bioinformatics/bty895
```

I created the volcano plot as previously described, except this time using the muttreatment dataset.

```
#Arrange by p value
shrink_muttreatment = arrange(as.data.frame(shrink_muttreatment),padj)


#Retrieve symbols of top 30 genes
symb = shrink_muttreatment$external_gene_name[1:30]


library(EnhancedVolcano)


#Create plot
EnhancedVolcano(shrink_muttreatment,
                lab = shrink_muttreatment$external_gene_name,
                labSize = 7.0,
                selectLab = c(symb),
                pointSize = 5.0,
                title = "CYP21A2 MUT treated vs untreated",
                axisLabSize = 35,
                titleLabSize = 50,
                subtitle = " ",
```

```
                x ='log2FoldChange',
                y ='padj',
                legendLabels=c('Not sig.','Log (base 2) FC','p-value',
                               'p-value & Log (base 2) FC'),
                legendPosition = 'right',
                legendLabSize = 20,
                legendIconSize = 10)
```

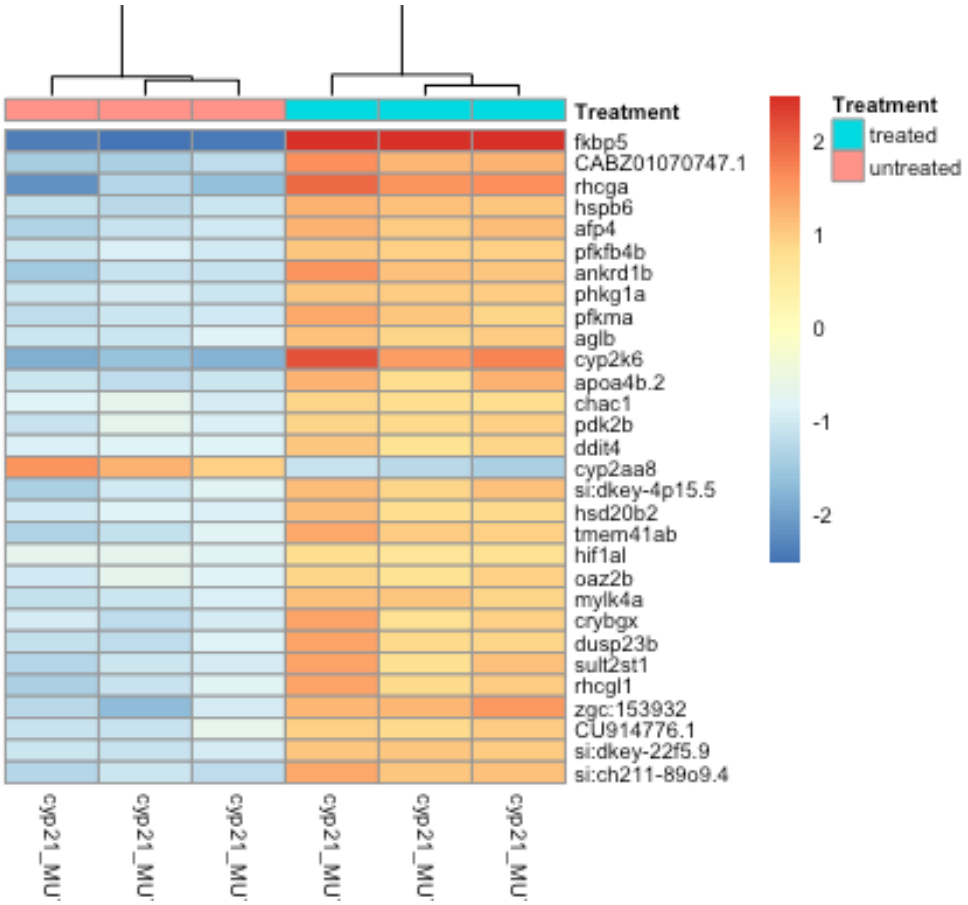


Interestingly the plot shows an almost complete reversal in gene expression. most noticabile is the change in fkbp5 which is singicantly upregulated in the treated group.

I then created a heatmap as described before

```
#Initialise pheatmap if not done previously
library(pheatmap)


#log2 transformation
```

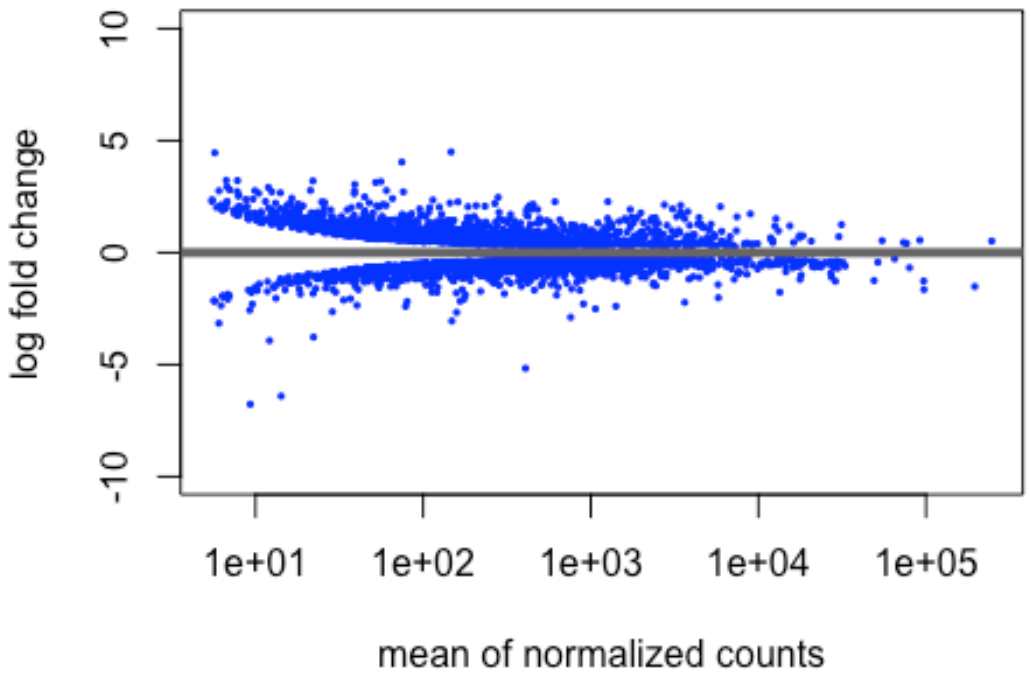
```
vsd_muttreated = vst(dds_muttreatment)

#Create matrix containing the top 30 most significant genes
muttreated_mat = assay(vsd_muttreated)[
head(order(mutreatment_results$padj),30), ]

#Subtract the mean reads for each gene
muttreated_mat = muttreated_mat - rowMeans(muttreated_mat)

#Read in genotypes for annotation
sampleinfo = as.data.frame(colData(dds_muttreatment)['Treatment'])



#Plot heatmap
pheatmap(muttreated_mat,
         fontsize = 7,
         cellwidth = 30,
         cellheight = 7,
         labels_row = symb,
         annotation = sampleinfo,
         cluster_rows = F,
         clustering_distance_rows = F,
         labels_col = dds_muttreatment$Name)
```



Once again it appears that the expression is the opposite when the mutants are treated. Only one gene was downregulated, whilst all the reat inlcuding fkbp5 where significantly upregulated

#01/11/2020 Mutant treated vs wild type ### Ma plot of mutant treated vs WT untreated

```
sig_muttreatmentwt = subset(muttreatedwt_results, padj < 0.05)


plotMA(sig_muttreatmentwt, ylim = c(-10, 10))
```



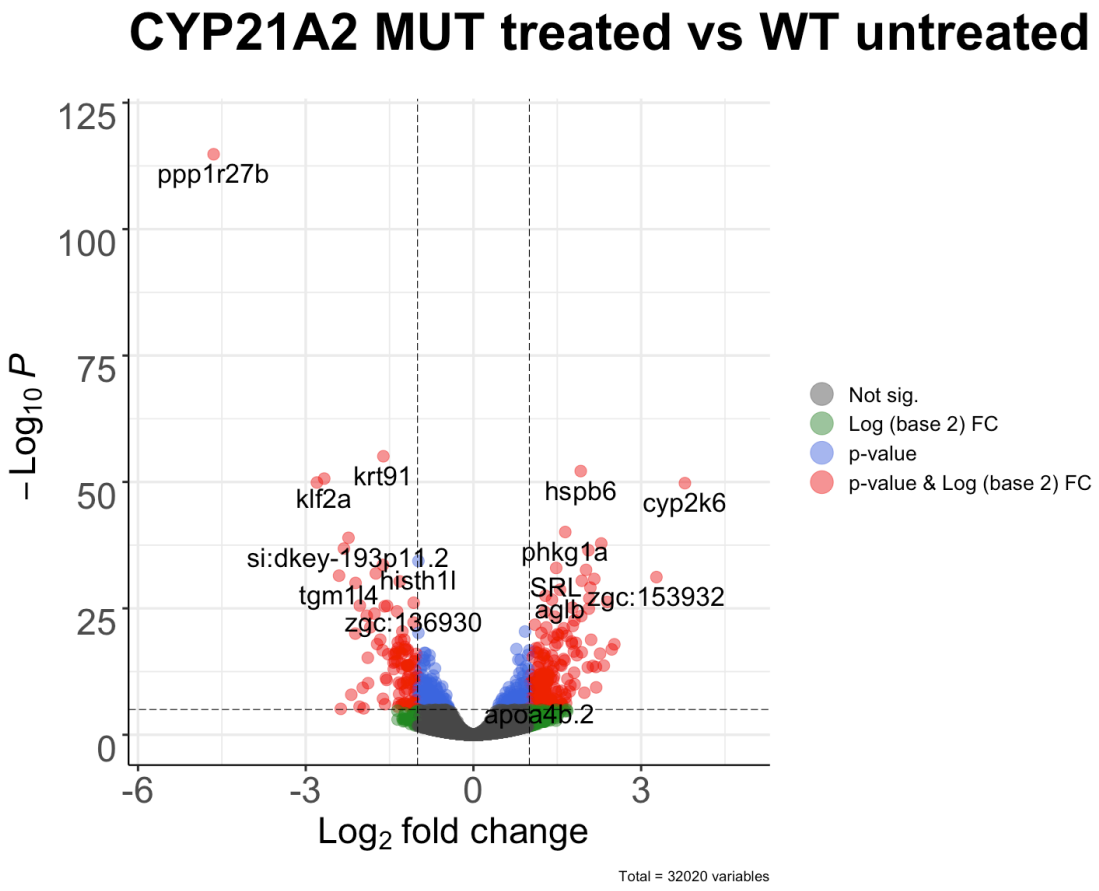
Interestingly this MA plot shows the greatest difference in expression so far. There are more significant genes on this plot than in the previous two comparisons, despite the fact that I ecprected the treated mutants to have at least some similarity to the wild types.

Next i produced a volano plot. As before i performed an lfc shrink on the data

```
shrink_muttreatmentwt = lfcShrink(de_mutreatedwt, contrast =
c('Treatment','treated','untreated'), res = muttreatedwt_results, type =
'normal')

## using 'normal' for LFC shrinkage, the Normal prior from Love et al (2014).
##
## Note that type='apeglm' and type='ashr' have shown to have less bias than
type='normal'.
## See ?lfcShrink for more details on shrinkage type, and the DESeq2
vignette.
## Reference: https://doi.org/10.1093/bioinformatics/bty895
```

I created the volcano plot as previously described, except this time using the muttreatmentwt dataset.

```
shrink_muttreatmentwt = arrange(as.data.frame(shrink_muttreatmentwt),padj)
symb = shrink_muttreatmentwt$external_gene_name[1:30]
library(EnhancedVolcano)
EnhancedVolcano(shrink_muttreatmentwt,
                lab = shrink_muttreatmentwt$external_gene_name,
                selectLab = c(symb),
                labSize = 9.0,
                pointSize = 5.0,
                title = "CYP21A2 MUT treated vs WT untreated",
                axisLabSize = 35,
                titleLabSize = 50,
                subtitle = " ",
                x ='log2FoldChange',
                y ='padj',
                legendLabels=c('Not sig.','Log (base 2) FC','p-value',
                               'p-value & Log (base 2) FC'),
                legendPosition = 'right',
                legendLabSize = 20,
                legendIconSize = 10)
```

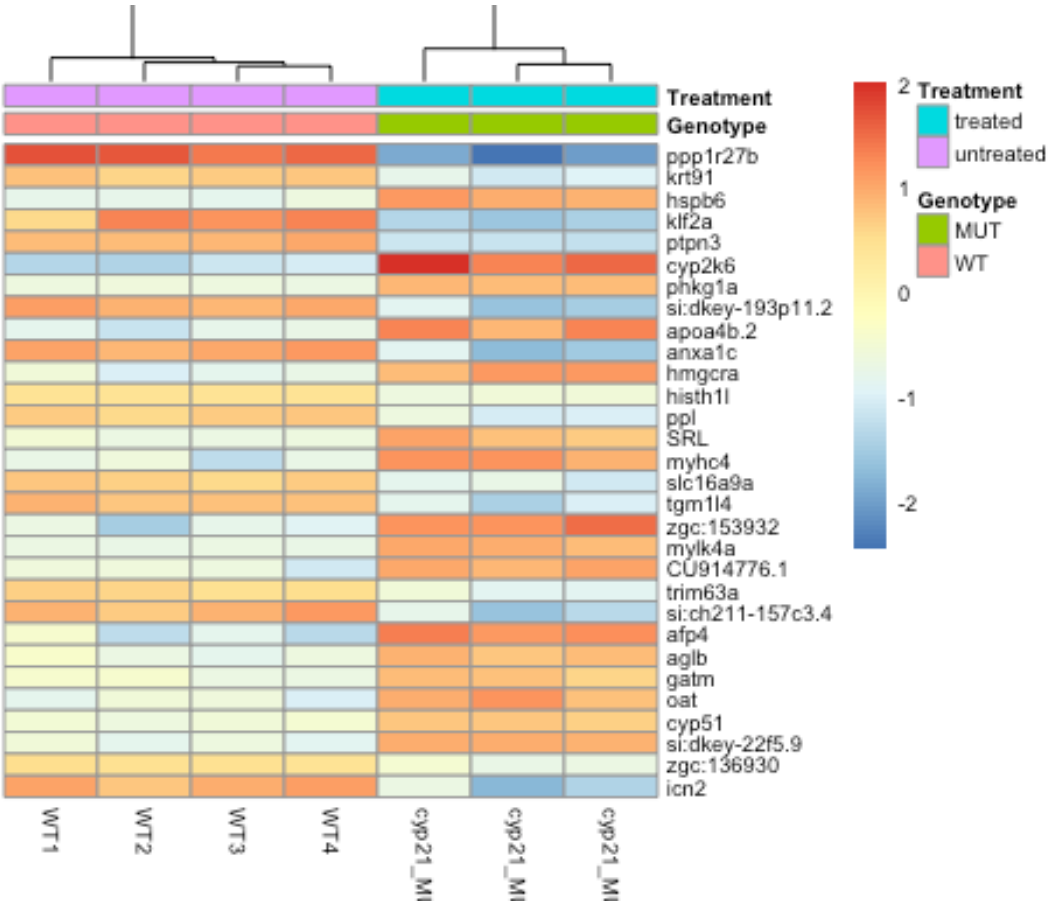


Once again this plot shows that there is a signifcant difference between the gene expression, however unlike the previous two comparisons it isnt skewed in any direction. Significanlty downregulated genes include some involved in immune regulation, such as anxa1.

I then created a heatmap as described before

```
#Initialise pheatmap if not done previously
library(pheatmap)

#log2 transformation
vsd_mutreatedwt = vst(dds_mutreatedwt)

#Create matrix containing the top 30 most significant genes
muttreatedwt_mat = assay(vsd_mutreatedwt)[
head(order(muttreatedwt_results$padj),30), ]
```

```
#Subtract the mean reads for each gene
muttreatedwt_mat = muttreatedwt_mat - rowMeans(muttreatedwt_mat)

#Read in genotypes for annotation
sampleinfo =
as.data.frame(colData(dds_mutreatedwt)[c('Genotype','Treatment')])



#Plot heatmap
pheatmap(muttreatedwt_mat,
         fontsize = 7,
         cellwidth = 30,
         cellheight = 7,
         labels_row = symb,
         annotation = sampleinfo,
         cluster_rows = F,
         clustering_distance_rows = F,
         labels_col = dds_mutreatedwt$Name)
```



The heatmap mirrors what was seen in the volcano plot, however when comparing this with the others it appears that the difference in the count data is not as pronounced as it was in the previous two comparisons. This may indicate a lower amount of overall difference, implying that although there are differences, these are less prpnounced and may have less of an effect.

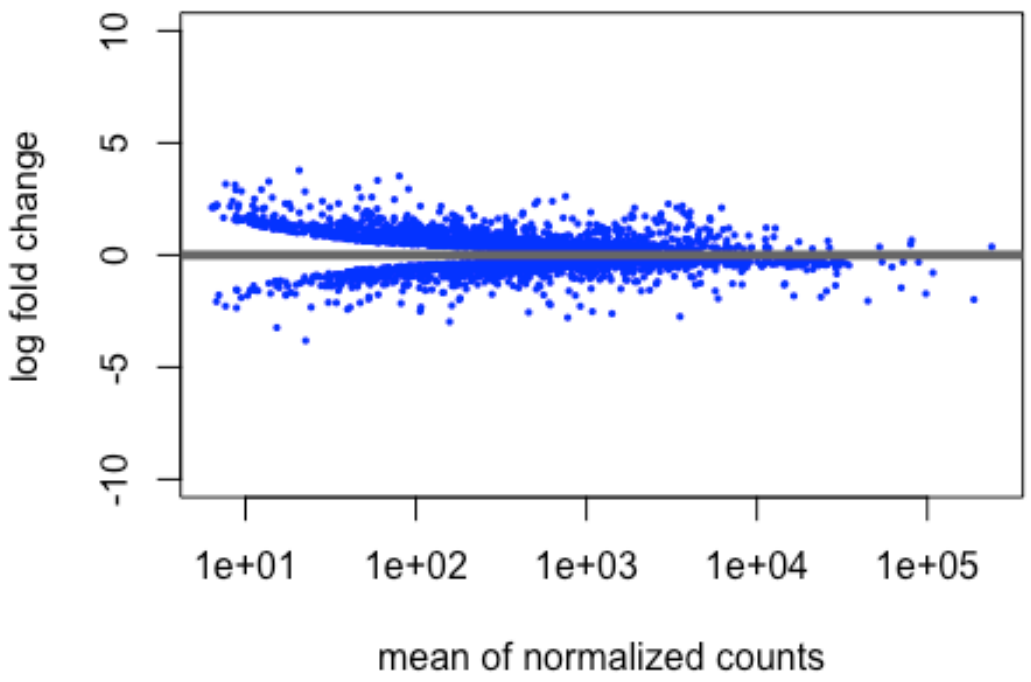
#03/11/2020 Wild type treated vs wild type untreated

Ma plot of wild type treated vs untreated

The Ma plot shows even more difference in this case. It appears that elevated glucocorticoid signalling has a significant effect on gene expression.

```
#Filter sig genes
sig_wt = subset(wt_results, padj < 0.05)
```

```
#PLot data
plotMA(sig_wt, ylim = c(-10, 10))
```

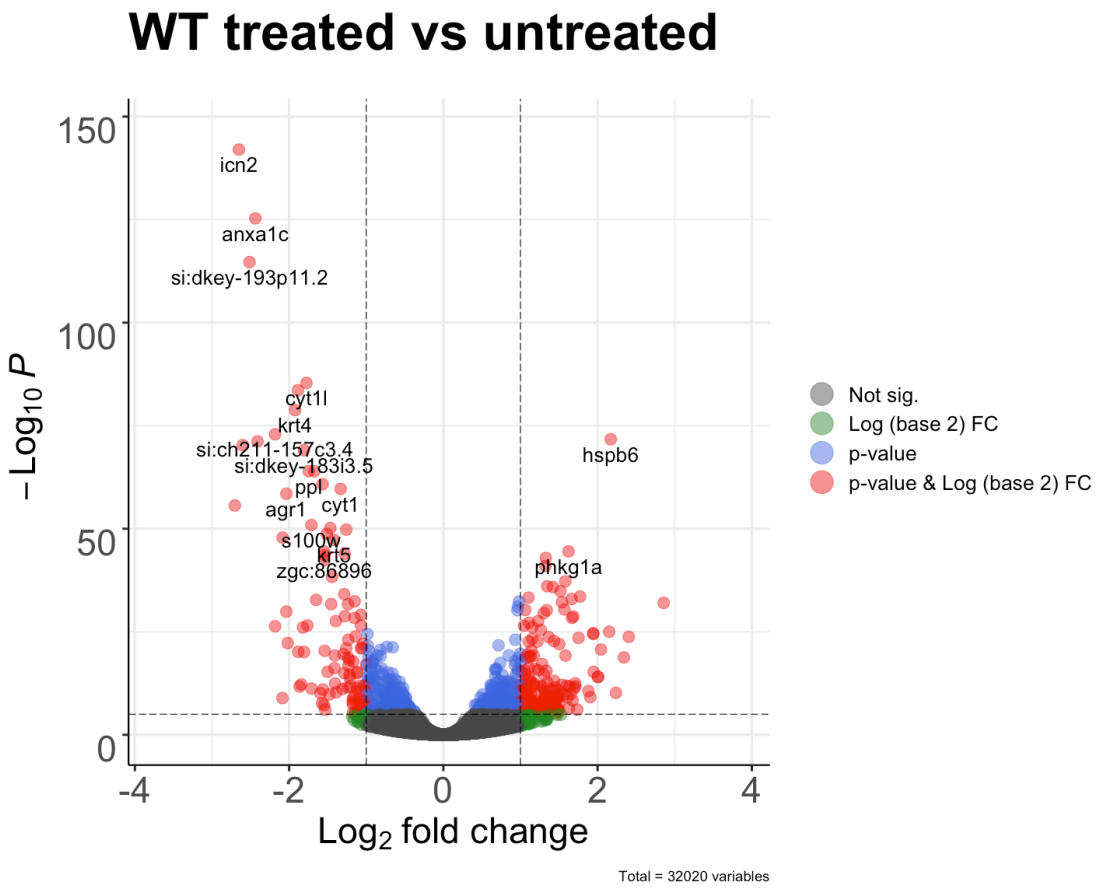


Next i produced a volano plot. As before i performed an lfc shrink on the data

```
#lfc shrink
shrink_wt = lfcShrink(de_wt, contrast = c('Treatment','treated','untreated'), res = wt_results, type = 'normal')

## using 'normal' for LFC shrinkage, the Normal prior from Love et al (2014).
##
## Note that type='apeglm' and type='ashr' have shown to have less bias than type='normal'.
## See ?lfcShrink for more details on shrinkage type, and the DESeq2 vignette.
## Reference: https://doi.org/10.1093/bioinformatics/bty895
```

I created the volcano plot as previously described, except this time using the muttreatment dataset.

```
#arrange by p value
shrink_wt = arrange(as.data.frame(shrink_wt),padj)

#Collect top 30 gene symbols
symb = shrink_wt$external_gene_name[1:30]
library(EnhancedVolcano)

#Produce plot
EnhancedVolcano(shrink_wt,
                lab = shrink_wt$external_gene_name,
                selectLab = c(symb),
                labSize = 7.0,
                pointSize = 5.0,
                title = "WT treated vs untreated",
                axisLabSize = 35,
                titleLabSize = 50,
                subtitle = " ",
                x ='log2FoldChange',
                y ='padj',
                legendLabels=c('Not sig.','Log (base 2) FC','p-value',
                               'p-value & Log (base 2) FC'),
                legendPosition = 'right',
                legendLabSize = 20,
                legendIconSize = 10)
```

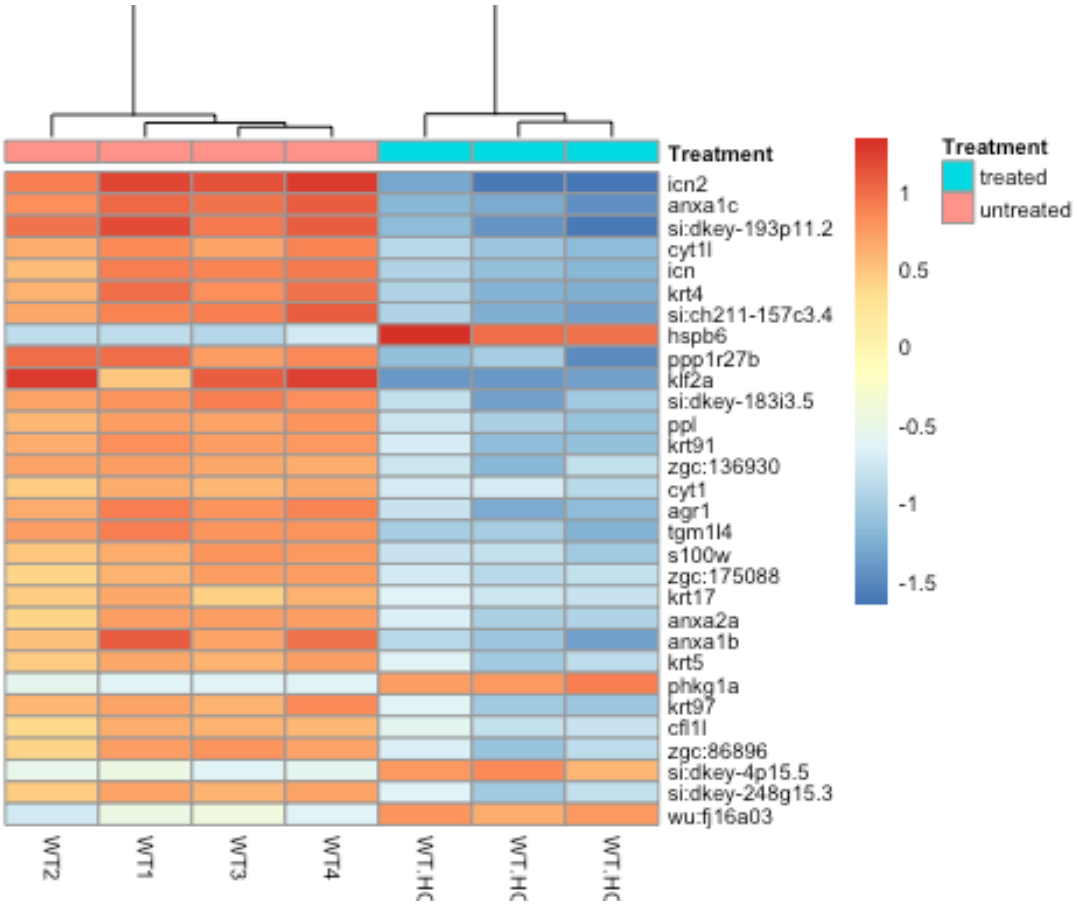


Slightly more genes where downrefulated in the treated group compared to the untreated. Once again, these genes appear to be involved in immune response with annexin making an appearance again. As before hsbp6 was also upregulated. This may indicate that treatment with hydrocortisone producing effects on gene expression that are associated with elevated glucocorticoid signalling. Is the treatment therefor an overkill?

I then created a heatmap as described before

```
#Initialise pheatmap if not done previously
library(pheatmap)

#log2 transformation
vsd_wt = vst(dds_wt)

#Create matrix containing the top 30 most significant genes
```

```
wt_mat = assay(vsd_wt)[ head(order(wt_results$padj),30), ]

#Subtract the mean reads for each gene
wt_mat = wt_mat - rowMeans(wt_mat)

#Read in genotypes for annotation
sampleinfo = as.data.frame(colData(dds_wt)['Treatment'])



#Plot heatmap
pheatmap(wt_mat,
         fontsize = 7,
         cellwidth = 30,
         cellheight = 7,
         labels_row = symb,
         annotation = sampleinfo,
         cluster_rows = F,
         clustering_distance_rows = F,
         labels_col = dds_wt$Name)
```

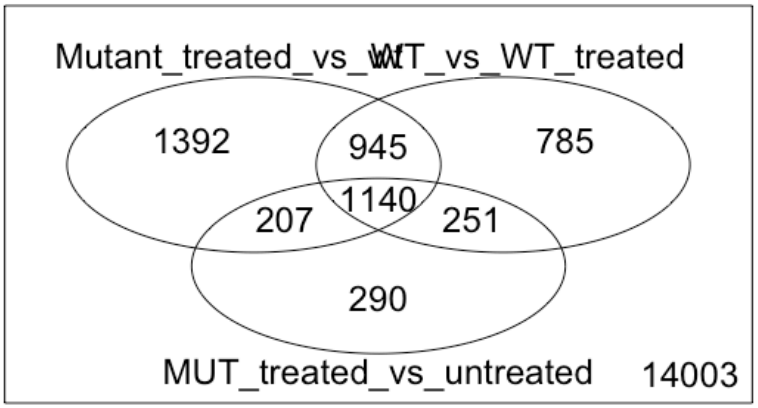


The heatmap shows significant upregulation in the top 30 genes. genes of note are once again the annexin genes, whic are involved in immune response.

#05/11/2020 Comparing De gene lists Inorder to follow up on the idea that the treatment produced similar effects on the mutants as it did in the wild type im going to compare the differentially expressed genes in both groups. I can use the limma library to make a venn diagram showing overlap between the significantly differentially expressed genes

```
#Create a data frame with two lists of differentially expressed genes to
compare
venn_data = data.frame(Mutant_treated_vs_wt = muttreatedwt_results$padj<0.05,
WT_vs_WT_treated = wt_results$padj < 0.05, MUT_treated_vs_untreated =
mutreatment_results$padj < 0.05)

#initialise the limmma library for producing the venn diagram
library(limma)
```

```
##
## Attaching package: 'limma'

## The following object is masked from 'package:DESeq2':
##
##     plotMA

## The following object is masked from 'package:BiocGenerics':
##
##     plotMA

#Plot data
vennDiagram(venn_data)
```



There appears to be signifcant overlap between the de gene lists, which may indicate that the treatment produces similar affects to those seen under elevated glucocorticoid signalling.

09/11/2020 Gene enrichment analysis

In order to assess the biological significance of these results we need to figure out the biological processes that these genes are involved in. This will involve gene ontology analysis. Over time genes have been assigned to certain GO terms (gene ontology). for example a gene involved in amino acid catabolism will have that as its GO term. Multiple genes are assigned to the same GO terms, and genes can have lots of different GO terms.

In order to assign the GO terms and visualise the resulting data i will be using two online resources: GOrilla and revigo. GOrilla allows the user to paste in a single list of genes ranked by adjusted p value, from most significant to least. It will then assing GO terms to these genes, taking into account the organism that they belong to and the significance of their appeartance in the list. A list of GO terms is produced relating to biological processes, functions or locations depending on what the suer wishes to view.

The list of GO terms can then be fed into revigo, which is an online tool allowing for the visualisation of this list. It takes into account the significance of each GO term and allows for the production of a treemap showing eah term in different sized boxes depending on how sognificantly they appear.

To produce each list i will use the csv files i created earlier. The data is already ordered by adjusted p value, so therefor i just need to paste the external_gene_name columns into GOrilla to generate the list. The resulting list of GO terms will then be fed into an R script generated by revigo to produce the treemaps

I generated four treemaps in the form of pdf files:

- MUTVWT.pdf
- MUTtreatVMUT.pdf
- MUTtreatVWT.pdf
- WTtreatVWT.pdf

My analysis is finished. If have more time or need more results i will insert all upregulated genes and all dowregulated genes into GOrilla seperately in order to identify GO terms related to them specifically, however i already have a lot of results so this may be too much