Differential Expression, Hansen et al.

Zachary Weber

5/19/2022

Contents

1 R Environment Setup **1**

2 Loading and Formatting the Counts Data **2**

3 Filtering and Normalizing Reads **4**

4 Estimating Dispersion among our Tags **6**

5 Fitting Models of differential Expression **7**

5.1 Cis-regulatory Differential Expression 8

5.2 Cis-Chromosomal Differential Expression 12

5.3 Global Differential Expression 13

1 R Environment Setup

For the differential expression analysis we're going to make use of a number of packages, some of which we'll need to download using BioConductor. For the statistical analyses of the RNA-Seq data, we're going to be using edgeR, a popular package which Directly models RNA-Seq counts data using negative binomial models. We're also going to download the package qvalue which makes doing FDR procedures for multiple-test corrections easy.

```
# install BiocManager
if (!require("BiocManager", quietly=TRUE)) {
  install.packages("BiocManager")
  BiocManager::install(version="3.15")
}

# install BioConductor packages
bioc_packages <- c("edgeR", "Gviz", "GenomicRanges", "qvalue")
BiocManager::install(bioc_packages)
library(edgeR)
library(Gviz)
library(GenomicRanges)
library(qvalue)
```

We'll also make use of some R packages found in CRAN, these are dplyr for data wrangling, ggplot2 for advanced graphics construction, and ggrepel for better text labels on graphics. We utilize stringr for convenient

grep-style operations. Finally, we make use of RColorBrewer and wesanderson for expanded color palletes in plotting.

Each of these packages, except ggrepel and RColorBrewer, is contained in the super-package tidyverse.

```
# install CRAN packages
library(tidyverse)
library(ggrepel)
library(RColorBrewer)
library(wesanderson)
```

2 Loading and Formatting the Counts Data

The experimental information from the pipeline is stored in a sample metadata file called **sample_align_metadata.csv**. We can read in this information, and use it to load our counts data and their corresponding annotations. From the sample metadata, we will collect the sample name, the SNP-centered enhancer target, its deletion genotype (0 -> WT, 1 -> D/WT, 2-> D/D), as well as the name of the counts table file.

```
# load the sample metadata
sampledata <- read.csv("counts_data/sample_align_metadata.csv",
                       header=TRUE, stringsAsFactors=FALSE, row.names="run_name"
                       ) %>% select(.,del_GT,SNP_roi,counts_table) %>%
                          mutate(., experiment_num=str_match(rownames(.),"^[1-9]+")) %>%
                            mutate_at(., c("counts_table"), basename) %>%
                              mutate_at(.,c("experiment_num","SNP_roi"), factor)

# show the sample data layout
head(sampledata)
```

```
##          del_GT   SNP_roi                  counts_table experiment_num
## 1-1a_S10      2 rs4955430 1-1a_S10.ReadsPerGene.out.tab              1
## 1-1b_S11      2 rs4955430 1-1b_S11.ReadsPerGene.out.tab              1
## 1-1c_S12      2 rs4955430 1-1c_S12.ReadsPerGene.out.tab              1
## 1-4a_S13      1 rs4955430 1-4a_S13.ReadsPerGene.out.tab              1
## 1-4b_S14      1 rs4955430 1-4b_S14.ReadsPerGene.out.tab              1
## 1-4c_S15      1 rs4955430 1-4c_S15.ReadsPerGene.out.tab              1
```

Next, we use the sample data to create experimental group labels and read in the transcript counts data for each sample. We also will need to remove the meta_tags from each count table. These summary tags begin with the pattern `^N_*` in the htseq-counts tables. These meta_tabs aggregate information particularly on reads which could not be cleanly assigned to specific tags (i.e. multi-mapping reads, unmapped reads, and ambiguous reads).

```
# load the raw counts data into an edgeR DGEList
counts_raw <- readDGE(files=sampledata$counts_table,
                      path="counts_data/",
                      labels=rownames(sampledata)
                      )
```

```r
counts_raw$samples$del_GT <- sampledata$del_GT
counts_raw$samples$enhancer <- sampledata$SNP_roi

# remove meta-tags summarizing multi-mapping, ambiguous, and un-mapped reads
meta_tags <- grep("^N_", rownames(counts_raw))
counts_raw <- counts_raw[-meta_tags, ]
```

Finally, we generate a design matrix that describes the relevant batches and experimental factors present in the data. This design matrix will be used downstream in our regression to estimate deletion effect sizes.

```r
# define design matrix specific to enhancer and genotype

# build a matrix called 'conditions' to encode the deletion genotypes
conditions <- matrix(0,nrow=nrow(counts_raw$samples), ncol=3)
rownames(conditions) <- rownames(counts_raw$samples)
colnames(conditions) <- c("rs4955430_KO", "rs6446275_KO", "rs10947532_KO")

conditions[c(1:9),1] <- 1
conditions[c(10:18),2] <- 1
conditions[c(19:24),3] <- 1
conditions <- conditions * counts_raw$samples$del_GT

head(conditions) # show the coding for the number of Deletion Alleles
```

```
##          rs4955430_KO rs6446275_KO rs10947532_KO
## 1-1a_S10            2            0             0
## 1-1b_S11            2            0             0
## 1-1c_S12            2            0             0
## 1-4a_S13            1            0             0
## 1-4b_S14            1            0             0
## 1-4c_S15            1            0             0
```

```r
# append these conditions to the sample metadata for ease of downstream use
counts_raw$samples <- cbind.data.frame(counts_raw$samples,
                                       as.data.frame(conditions)
                                       )

# form model matrix (here called design) for the deletion genotypes and batch
fct <- as.formula("~0 + enhancer + rs4955430_KO + rs6446275_KO + rs10947532_KO")
fct_names <- c("enhancer_rs10947532", "enhancer_rs4955430",
               "enhancer_rs6446275", "rs4955430_KO", "rs6446275_KO",
               "rs10947532_KO")

design <- model.matrix(fct, data=counts_raw$samples)
colnames(design) <- fct_names
print(colnames(design))
```

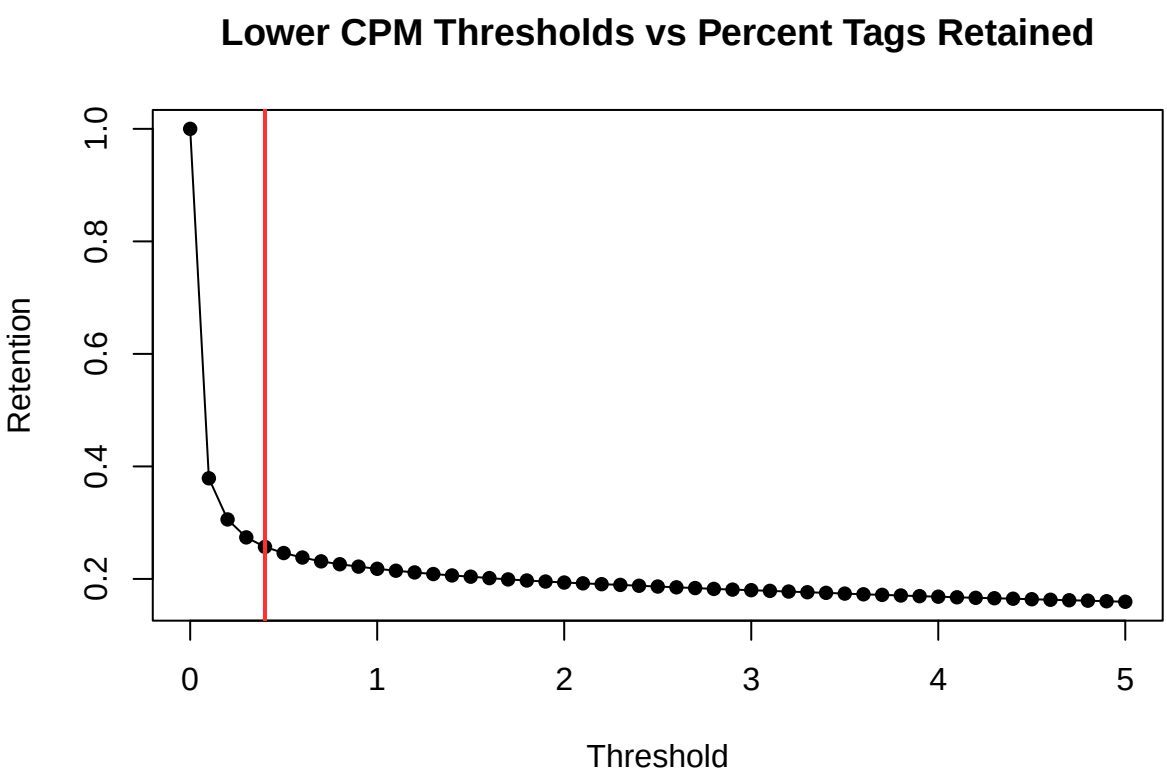
```
## [1] "enhancer_rs10947532" "enhancer_rs4955430"  "enhancer_rs6446275"
## [4] "rs4955430_KO"        "rs6446275_KO"        "rs10947532_KO"
```

3 Filtering and Normalizing Reads

Before we conduct differential expression analysis we need to filter out genes with exceedingly low expression values, whose differential expression detection power is close to zero. Let us first compute the curve representing the proportion of our tags that retained (1 - the proportion removed by filtering) for a given threshold of average sample abundance.

```
# define a function to return our retention fraction for a set of thresholds
test_filter_thresholds <- function(counts_matrix, thresholds) {
  kept <- sapply(thresholds, function(t) sum(rowMeans(counts_matrix) >= t))
  proportion.retained <- kept / nrow(counts_matrix)
  return(proportion.retained)
}

# run the filtering computation on CPM of the raw counts
thresholds <- seq(from=0, to=5, by=0.1)
retention <- test_filter_thresholds(counts_matrix=cpm(counts_raw),
                                    thresholds=thresholds
                                    )
```



From the above curve it seems like a threshold at 0.4 Average CPM (marked above by the red reference line) characterizes the elbow well and will give us the most efficient set of Tags on which to perform downstream analysis. Our retention at this point is about 26%, so we're keeping genes in roughly the top quartile of expression in these cells.

```
print(sprintf("Lower Threshold: %0.1g, Proportion of Tags Retained: %0.3g",
      thresholds[5], retention[5])
      )
```

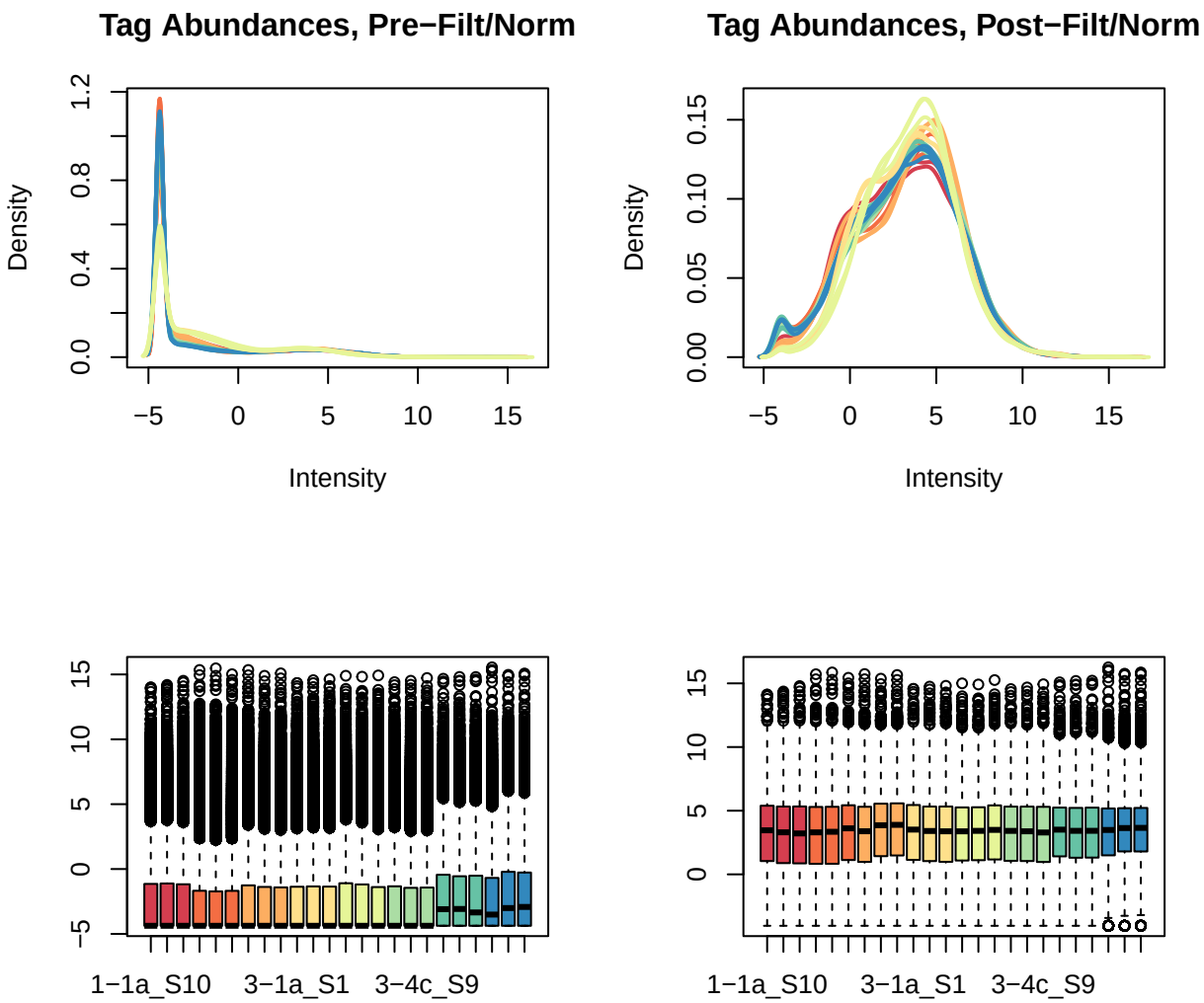
```
## [1] "Lower Threshold: 0.4, Proportion of Tags Retained: 0.257"
```

We can now filter the counts matrix using the above thresholds, and calculate the library sizes and normalization factors for our data.

```
# filter the reads
tokeep <- (rowMeans(cpm(counts_raw)) >= thresholds[5])
counts_filt <- counts_raw[tokeep, , keep.lib.sizes=FALSE]

# calculate the read normalization factors to use by applying TMM
counts_norm <- calcNormFactors(counts_filt, method="TMM")
```

Visualizing the filtering and normalization process on our abundance distributions:

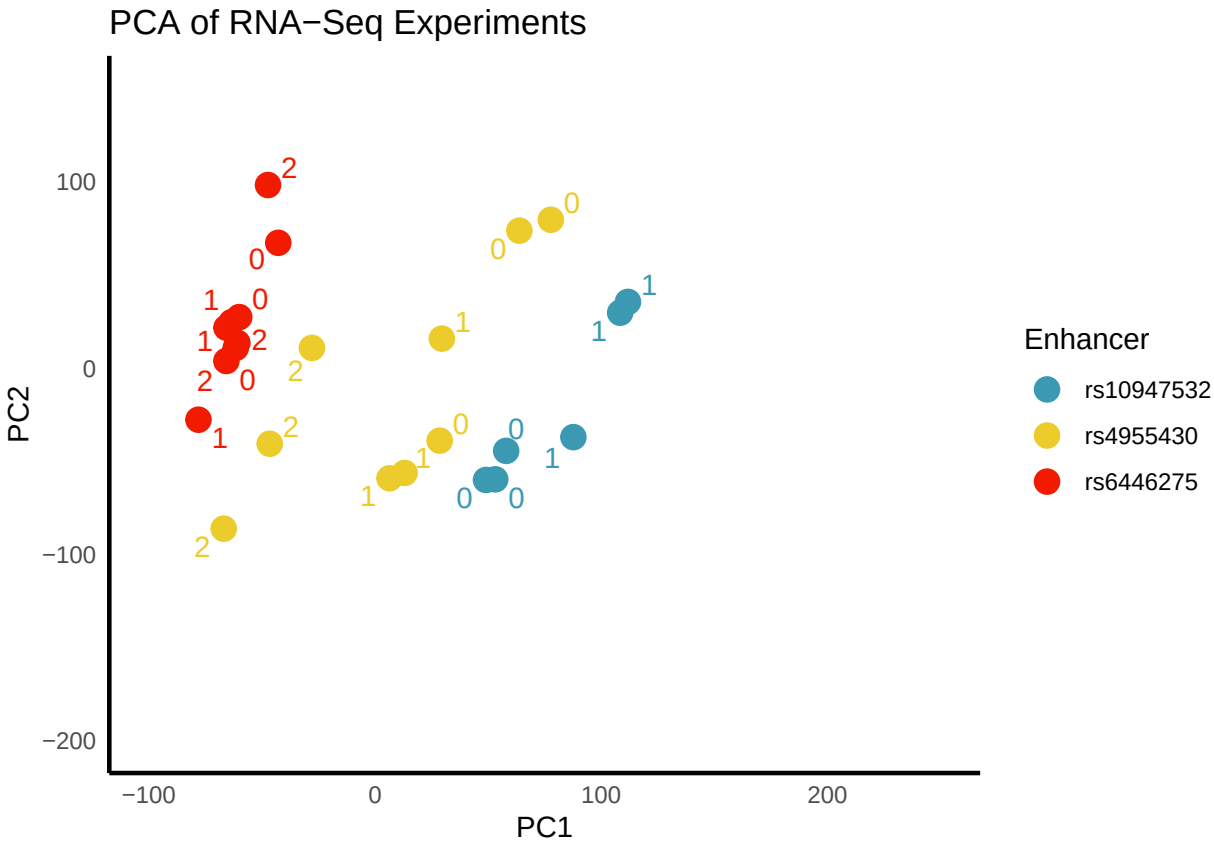


It can also be helpful to visualize our data on PCA:

```
# run PCA
pca.expr <- prcomp(t(cpm(counts_norm)), center=TRUE, scale=TRUE)
pct.var.explained <- (pca.expr$sdev)^2 / sum((pca.expr$sdev)^2)
```

```
# print the percent of variance explained by PCs 1-10
print(sprintf("Percent Variance Explained by PCs 1-10: %s",
                    round(sum(pct.var.explained[1:5]),3)))
```

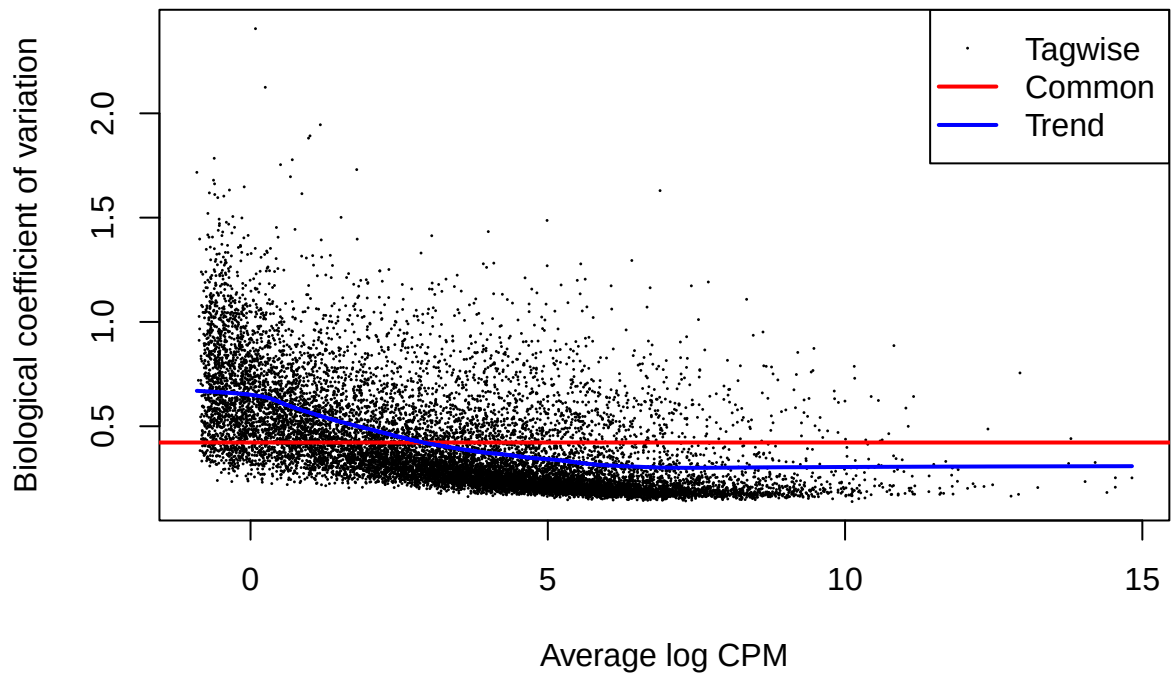
```
## [1] "Percent Variance Explained by PCs 1-10: 0.708"
```



4 Estimating Dispersion among our Tags

Estimating dispersion among the tags is key to properly parameterizing the linear models for differential expression analysis. The code below estimates dispersion, contingent on our experimental design, and gives us three options for dispersion parameters: (1) Tag-wise dispersion –> each gene gets its own sample variance which is derived empirically from the data. (2) Common dispersion –> each gene gets the same sample variance, based on the average abundance i the sample. Common dispersion assumes counts are Poisson distributed. Finally, (3) Trended Dispersion –> assumes a Common dispersion for each level of mean abundance, and fits a curve to the data using LOESS regression to find an appropriate paramter. Trended dispersion generally fits with count distribution models which are Negative-Binomial or Gamma-Poisson, which are often more appropriate. In this analysis we will be using Trended dispersion, the `edgeR` default.

```
# model dispersion and plot the results
counts_disp <- estimateDisp(counts_norm, design=design, trend.method="loess", robust=TRUE)
plotBCV(counts_disp)
```



Visualizing our dispersion across the samples, we can see that the trended dispersion best models the data. In some places the estimated value appears a bit high, due to outliers, but this bias skews our results toward more conservative testing in most cases. There are more aggressive methods available to drawing trended dispersion lines, but this should be fine.

5 Fitting Models of differential Expression

Now that we've estimated dispersion we're ready to conduct differential expression analysis. We have three sets of analyses that we're potentially interested in:

1. Genes in the cis-regulatory region of the enhancer deletion (1 MB window)
2. Genes in cis chromosome-wide (all genes on the same chromosome as the deletion)
3. All available genes (global analysis of DE)

For convenience, We have gene sets for the first two analyses in a file named *cisregulatory-genesets.txt* (1) and *chrom-genesets.txt* (2). Below we read them into a list and define a convenience function to access them from their comma-delimmited strings.

```
# load gene sets
gene_sets <- list(
  "cisreg"=read.delim("cisregulatory-genesets.txt",
            header=F, row.names=1, col.names=c("SNP","Genes")),
  "chrom"=read.delim("chrom-genesets.txt",
            header=F, row.names=1, col.names=c("SNP","Genes"))
)


# define accessor function
get_genes <- function(gene_list, enhancer, type) {
  genes <- gene_list[[type]][enhancer,]
  return(strsplit(genes, ",")[[1]])
}
```

Now we can fit our models (This step can be done independent of gene sets in the test)

```
# fit the model using Quasi-Likelihood Fit
fit <- glmQLFit(counts_disp, design=design)
```

Next we define a function that takes as input our fit model, gene set of interest, and our coefficient of interest, and outputs the results of the differential expression tests. We also define a function that can concatenate some high level statistics on the DE tests into a table:

```
# define a function to test DE
test_diff_exp <-function(fit, genes, coefficient) {
  fct_names <- colnames(fit$coefficients)
  DE <- glmQLFTest(fit, coef=match(coefficient, fct_names))
  DE$table <- DE$table[rownames(DE$table) %in% genes,]
  return(DE)
}

# summarize outputs of multiple DE tests
summarize_outputs <- function(de_outputs, exp_labels) {
  res <- sapply(de_outputs, function(x) summary(decideTests(x))) %>%
    cbind.data.frame(.)
  dimnames(res) <- list(c("Down","NS", "Up"), exp_labels)
  return(res)
}
```

5.1 Cis-regulatory Differential Expression

Let's begin by taking a look at cis-regulatory disruption caused by our enhancer deletions:

```
exp_names <- c("1-rs4955430", "3-rs6446275", "15-rs10947532")

de_cisreg_1 <- test_diff_exp(fit=fit, coefficient="rs4955430_KO",
                genes=get_genes(gene_sets, "rs4955430", "cisreg"))
de_cisreg_3 <- test_diff_exp(fit=fit, coefficient="rs6446275_KO",
                genes=get_genes(gene_sets, "rs6446275", "cisreg"))
de_cisreg_15 <- test_diff_exp(fit=fit, coefficient="rs10947532_KO",
                get_genes(gene_sets, "rs10947532", "cisreg"))
```

We can summarize the 5% FDR-significant calls in the cis-regulatory regions below by categorizing them into down-regulated genes, not-significantly altered genes, and up-regulated genes.

```
# generate summary table
de_cisreg <- list(de_cisreg_1, de_cisreg_3, de_cisreg_15)
summarize_outputs(de_cisreg, exp_names)
```

```
##      1-rs4955430 3-rs6446275 15-rs10947532
## Down          10           1             3
## NS            37          53            17
```

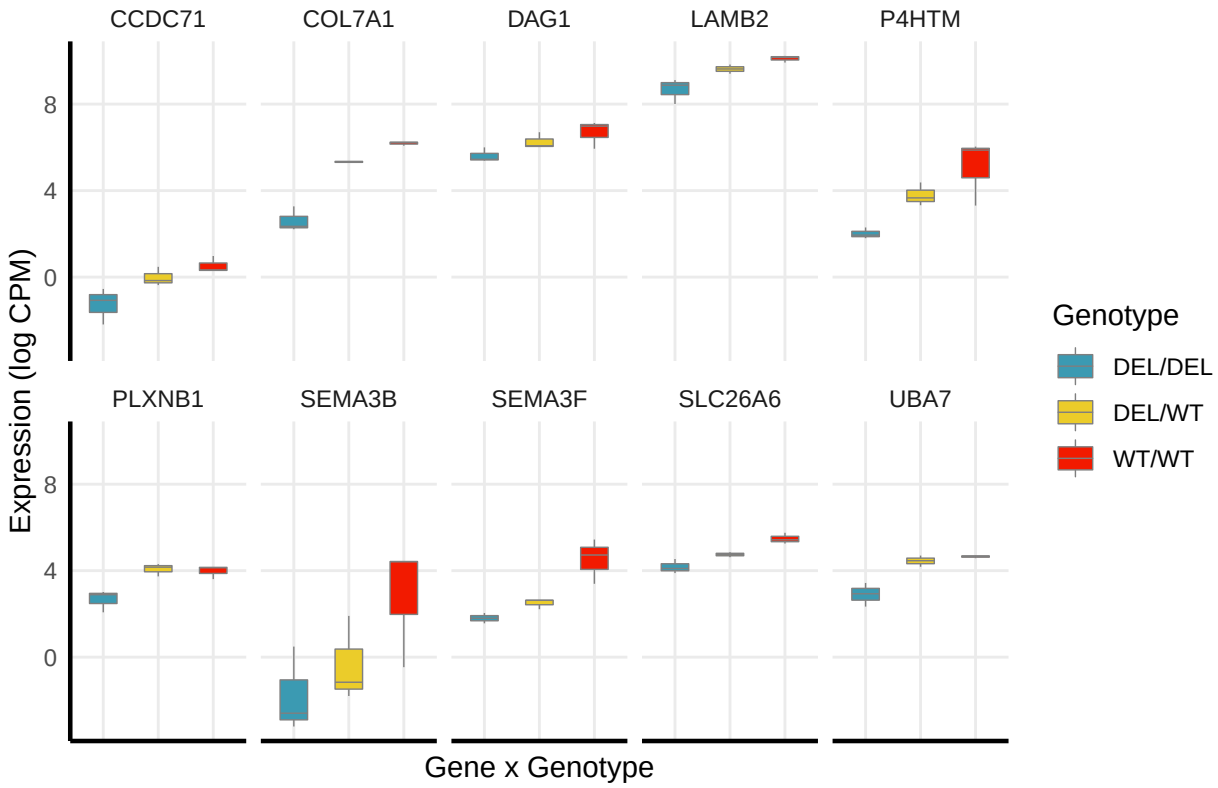
```
## Up              10            6              1
```

For each experiment, we can generate boxplots for expression of the differentially expressed genes in the cis-regulatory regions.

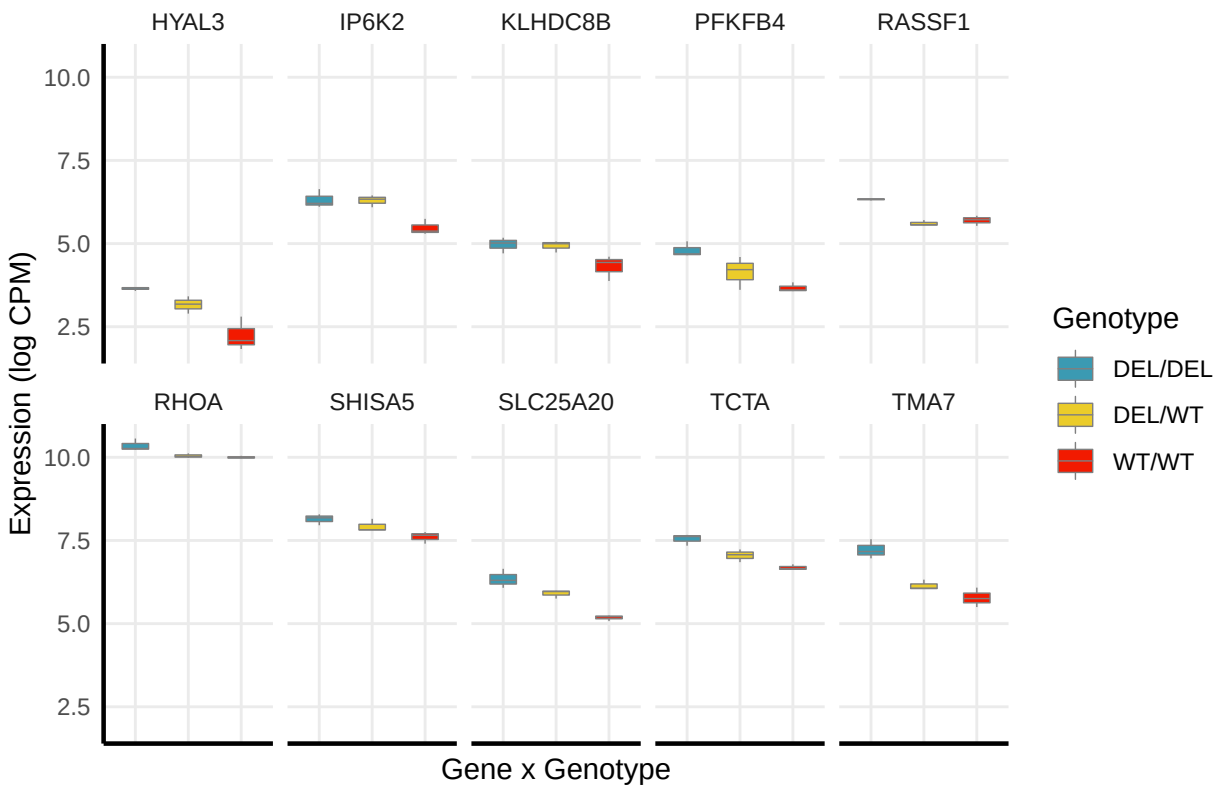
Starting with Deletions of Enhancer 1 at rs4955430:

```
# get dysregulated genes
upreg_1 <- get_altered(de_cisreg_1,"Up")
downreg_1 <- get_altered(de_cisreg_1, "Down")
```

Enhancer 1 (rs4955320 cis-regulatory/ down-regulated genes):



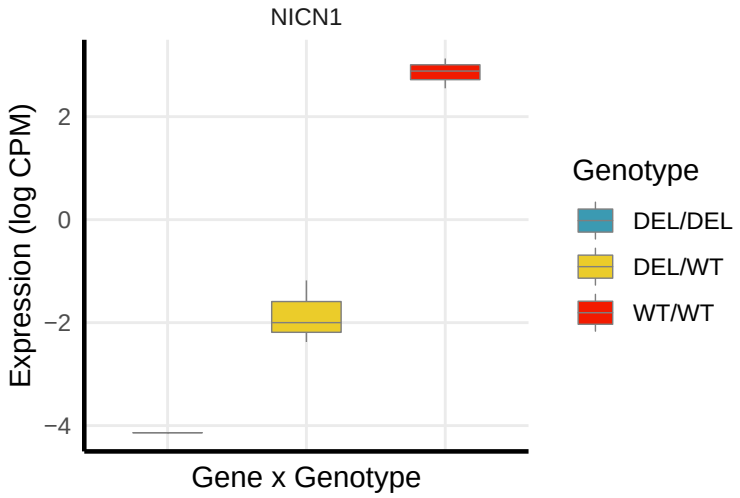
Enhancer 1 (rs4955320 cis-regulatory/ up-regulated genes):



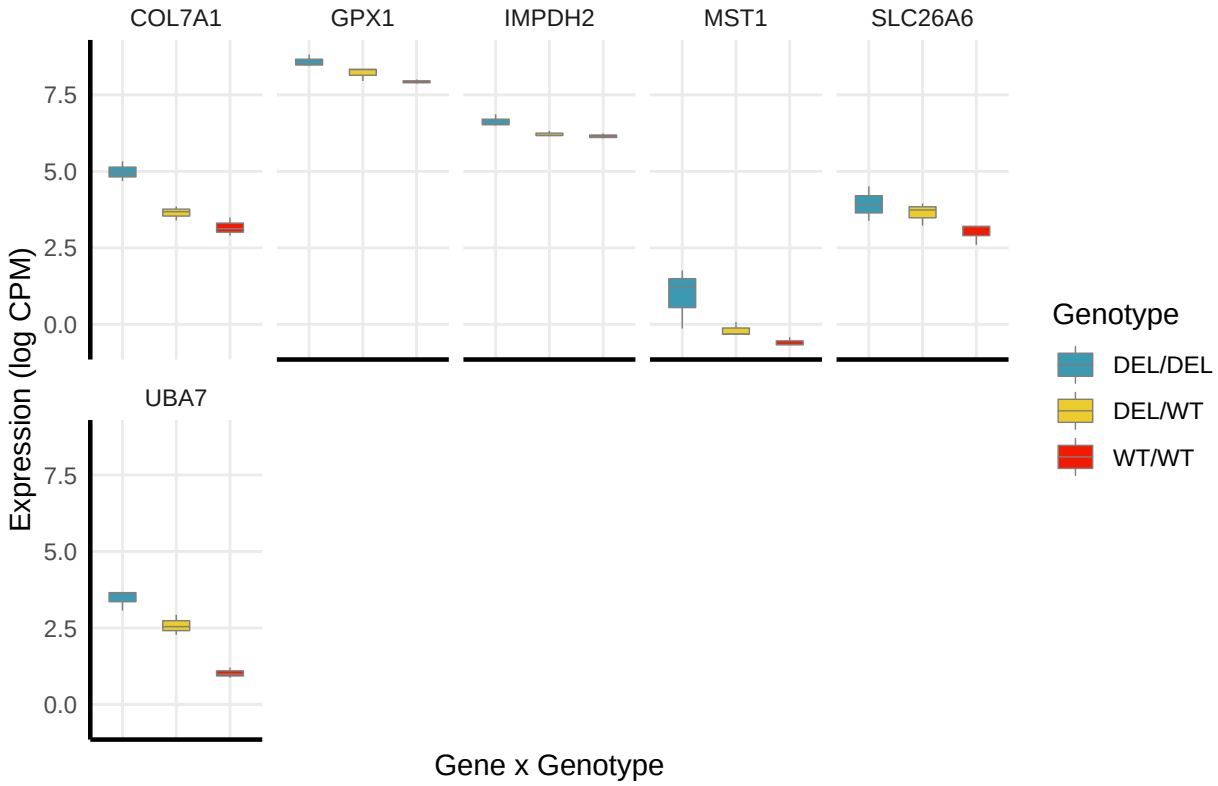
We can also view cis-regulatory changes from Deletions of Enhancer 3 at rs6446275:

```
# get dysregulated genes
upreg_3 <- get_altered(de_cisreg_3,"Up")
downreg_3 <- get_altered(de_cisreg_3, "Down")
```

Enhancer 3 (rs6446275 cis-regulatory/ down-regulated genes):



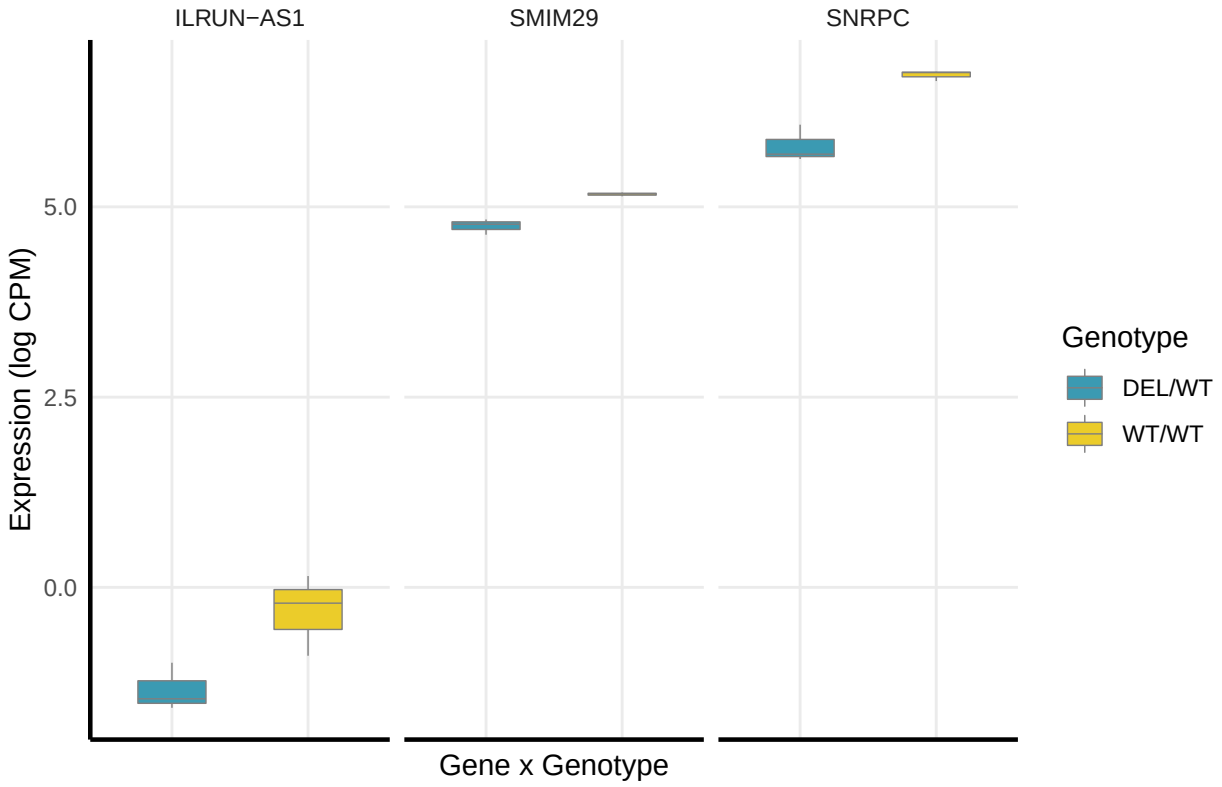
Enhancer 3 (rs46446275 cis-regulatory/ up-regulated genes):



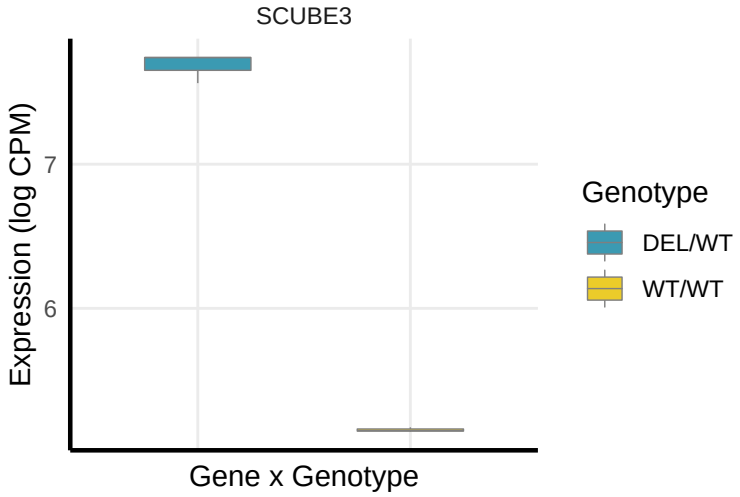
Finally, we can view cis-regulatory changes from Deletions of (non)Enhancer 15 at rs10947532:

```
# get dysregulated genes
upreg_15 <- get_altered(de_cisreg_15,"Up")
downreg_15 <- get_altered(de_cisreg_15, "Down")
```

(non)Enhancer 15 (rs10947532 cis-regulatory/ down-regulated genes):



(non)Enhancer 15 (rs10947532 cis-regulatory/ up-regulated genes):



5.2 Cis-Chromosomal Differential Expression

With a larger gene set comprising all genes on the same chromosome as each enhancer deletion, we can look at trends in differential expression in a more general sense. Below we repeat the analysis with these larger gene sets.

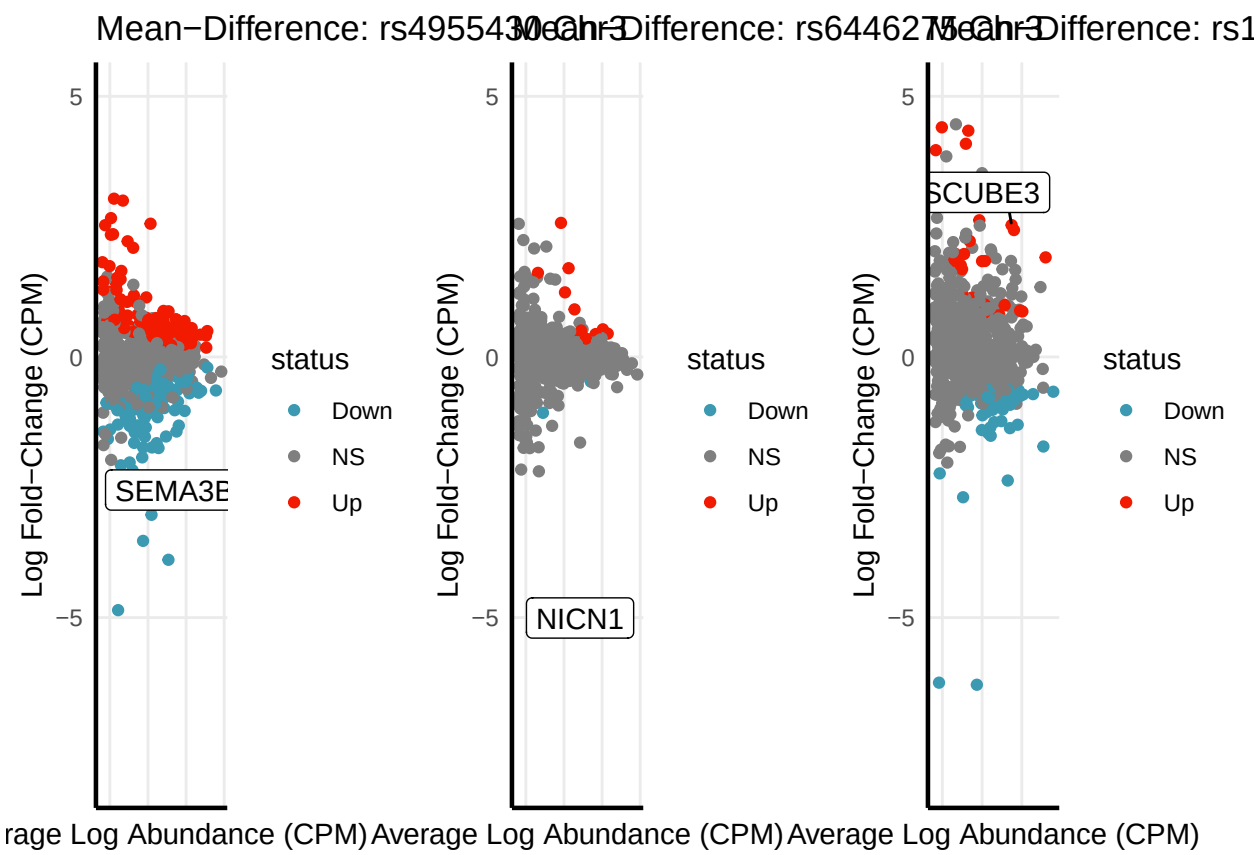
```
de_chrom_1 <- test_diff_exp(fit=fit, coefficient="rs4955430_KO",
                genes=get_genes(gene_sets, "rs4955430", "chrom"))
de_chrom_3 <- test_diff_exp(fit=fit, coefficient="rs6446275_KO",
                genes=get_genes(gene_sets, "rs6446275", "chrom"))
de_chrom_15 <- test_diff_exp(fit=fit, coefficient="rs10947532_KO",
                get_genes(gene_sets, "rs10947532", "chrom"))

de_chrom <- list(de_chrom_1, de_chrom_3, de_chrom_15)
summarize_outputs(de_chrom, exp_names)
```

```
##      1-rs4955430 3-rs6446275 15-rs10947532
## Down         141           4            56
## NS           564         828           605
## Up           138          11            31
```

```
# get dysregulated genes at the chromosome level
upreg_chrom_1 <- get_altered(de_chrom_1,"Up")
downreg_chrom_1 <- get_altered(de_chrom_1, "Down")
upreg_chrom_3 <- get_altered(de_chrom_3,"Up")
downreg_chrom_3 <- get_altered(de_chrom_3, "Down")
upreg_chrom_15 <- get_altered(de_chrom_15,"Up")
downreg_chrom_15 <- get_altered(de_chrom_15, "Down")
```

Because of the number of genes involved, its easier to view these changes on a Mean-Difference plot:



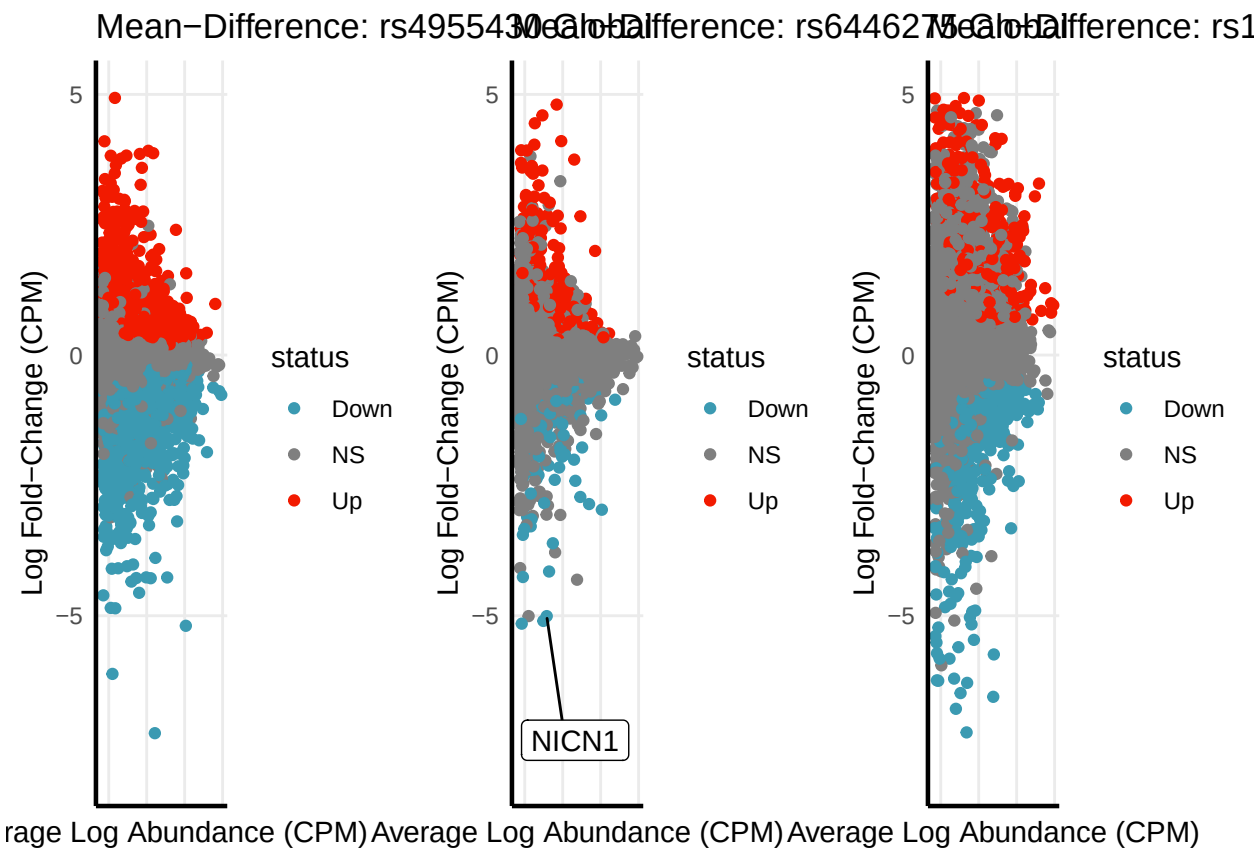
5.3 Global Differential Expression

Finally, we can look at trends in differential expression at the global level.

```r
genes <- rownames(fit$fitted.values)
de_glob_1 <- test_diff_exp(fit=fit, coefficient="rs4955430_KO", genes=genes)
de_glob_3 <- test_diff_exp(fit=fit, coefficient="rs6446275_KO", genes=genes)
de_glob_15 <- test_diff_exp(fit=fit, coefficient="rs10947532_KO", genes=genes)

de_glob <- list(de_glob_1, de_glob_3, de_glob_15)
summarize_outputs(de_glob, exp_names)
```

```
##      1-rs4955430 3-rs6446275 15-rs10947532
## Down        2624         176          1299
## NS          9151       13530         11974
## Up          2260         329           762
```

```r
# get dysregulated genes at the global level
upreg_glob_1 <- get_altered(de_glob_1,"Up")
downreg_glob_1 <- get_altered(de_glob_1, "Down")
upreg_glob_3 <- get_altered(de_glob_3,"Up")
downreg_glob_3 <- get_altered(de_glob_3, "Down")
upreg_glob_15 <- get_altered(de_glob_15,"Up")
downreg_glob_15 <- get_altered(de_glob_15, "Down")
```

Because of the number of genes involved, its again easier to view these changes on a Mean-Difference plot: