Using LIMMA in proteomics

Here, we will explore the use of LIMMA ("linear models for microarray data") for performing linear modelling. As we will see, the advantage of LIMMA is that it can take account of the relationship between feature intensity and variance to update the standard error estimates for the coefficients in the linear model. This is particularly useful when we have a low number of replicates and the standard error estimates for a given feature are likely to be over- or under-esimated. Thus limma can help address issues with both False negatives and False positives.

The original limma publication (2004!!) is here: https://www.ncbi.nlm.nih.gov/pubmed/16646809 For a proper explanation of the statitical model see: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5373812/

Although `limma` was originally designed for microarray data, it is also widely used for RNA-Seq. However, limma is agnostic to the type of input data so long as it's reasonable to assume the values are approximately gaussian distributed. For this reason, the quantification values should first be log transformed.

Note that the expected data format for `limma` is an `ExpressionSet` object. As stated in the documentation for the MSnSet class (https://www.rdocumentation.org/packages/MSnbase/versions/1.20.7/topics/MSnSet-class): " The `MSnSet` class is derived from the `eSet` class and mimics the `ExpressionSet` class classically used for microarray data." It's therefore relatively straightforward to use `limma` with proteomics data in a `MSnSet`.

```
# load packages
library(tidyverse)
library(limma)
library(biobroom)
library(Hmisc)
library(MSnbase)

# set up standardised plotting scheme
theme_set(theme_bw(base_size = 20) +
            theme(panel.grid.major=element_blank(),
                  panel.grid.minor=element_blank(),
                  aspect.ratio=1))

cbPalette <- c("#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2", "#D55E00", "#CC79A7", "#999999")
```

Again, we start by reading in the MSnSets

```
total_protein_quant <- readRDS("../raw/total_res_pro_agg_norm.rds")
rbp_protein_quant <- readRDS("../raw/rbp_res_pro_agg_norm.rds")
```

Let's start by applying limma to the total protein quantification data only. First of all we need to create a design matrix. We can do this from the `pData` since this contains the information about the samples

```
condition <- pData(total_protein_quant)$Condition
design <- model.matrix(~condition)
print(design)
```

```
##   (Intercept) conditionM
## 1           1          1
## 2           1          1
## 3           1          1
## 4           1          0
## 5           1          0
## 6           1          0
## attr(,"assign")
## [1] 0 1
## attr(,"contrasts")
## attr(,"contrasts")$condition
## [1] "contr.treatment"
```

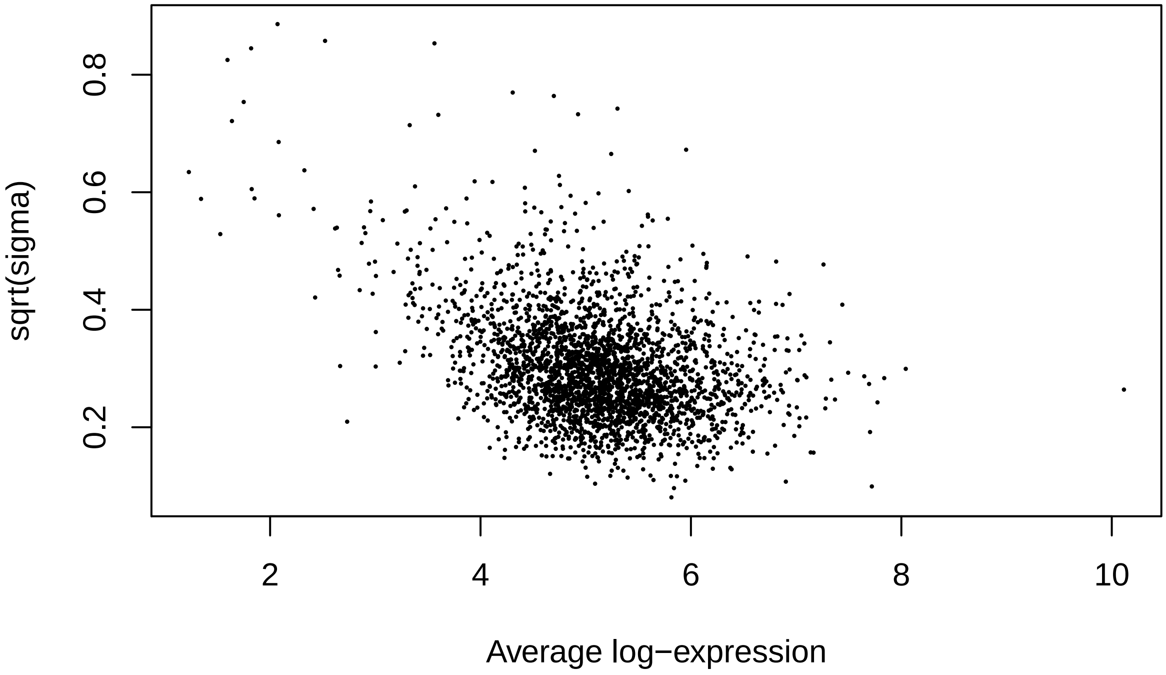
Then we fit the model using this design and update the estimates for the standard errors for each coefficient using the eBayes function.

Below we run limma to idenify the proteins with a significant change in abundance between conditions

```
# fit linear model to each protein
total_fit_lm <- lmFit(exprs(total_protein_quant), design)

# extract results for coefficient of interest
total_fit_lm_c <- contrasts.fit(total_fit_lm, coefficients=c("conditionM"))
```

As expected, there is a relationship between mean intensity and variance, although this is almost all limited to the very low intensity values.

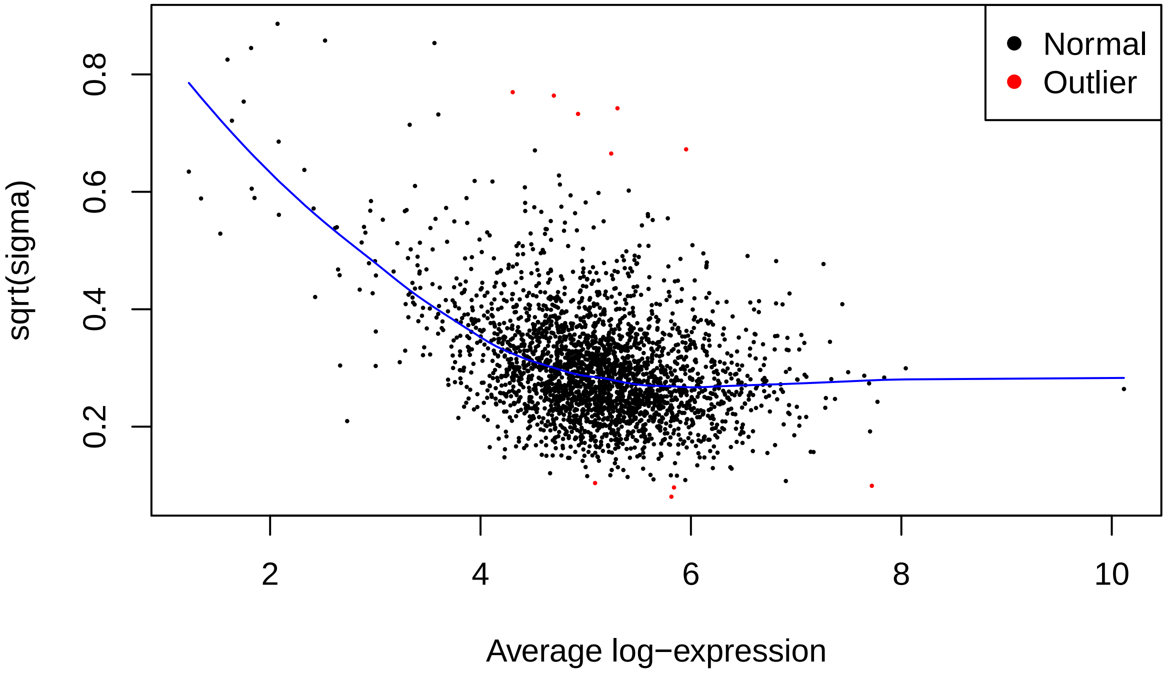
```
plotSA(total_fit_lm_c)
```



3.1 Question: Why do we expect a relationship between mean intensity and variance?

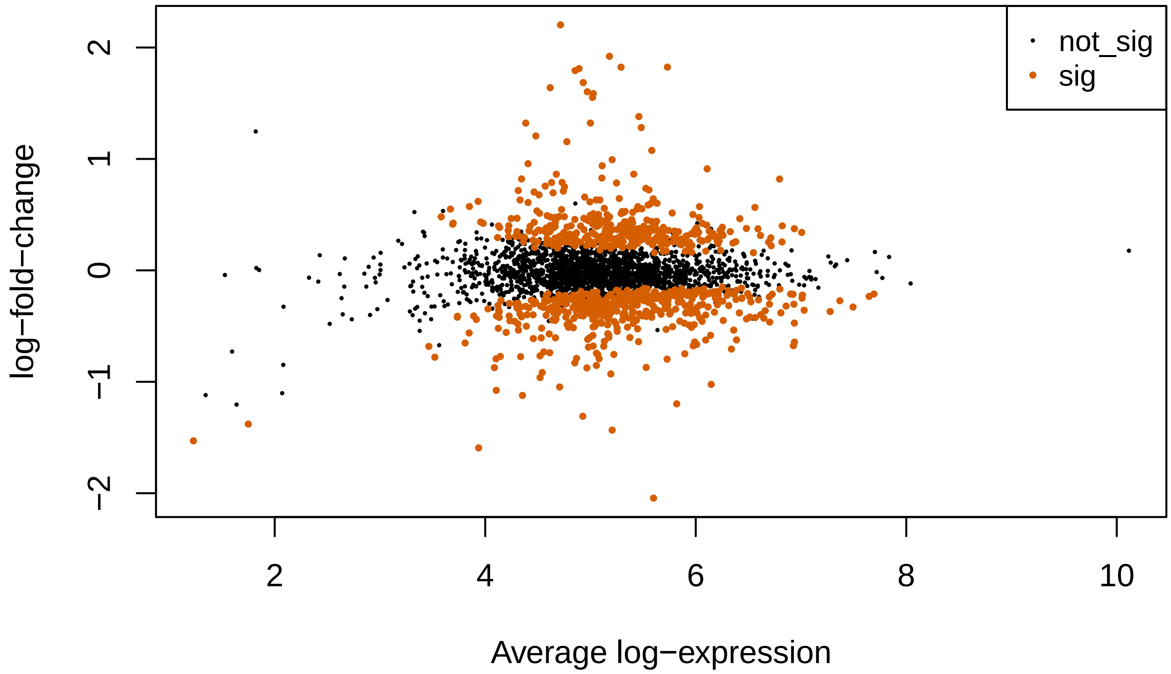
Below, we perform the emperical Bayesian shrinking of the std errors towards the trend (trend=TRUE). We will also use the robust=TRUE option to ensure that the outliers don't affect the trend estimation.

```
# shrink std errors to abundance vs. stdev trend
total_fit_lm_e_c <- eBayes(total_fit_lm_c, trend=TRUE, robust=TRUE)

plotSA(total_fit_lm_e_c)
```



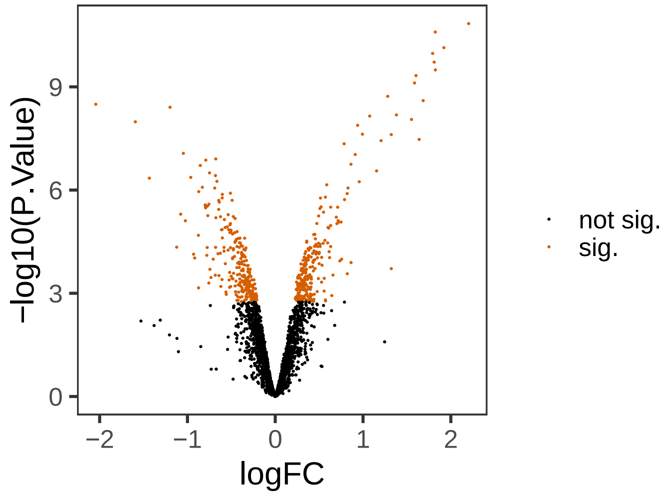
Below, we plot the average intensity vs log change. This is a useful QC plot to show that nothing odd has happened with our linear modeling.

```
# identify significant changes
p_value_status <- ifelse(total_fit_lm_e_c$p.value[,'conditionM']<0.01, "sig", "not_sig")

# plot
limma::plotMA(total_fit_lm_e_c, status=p_value_status,
       col=c(cbPalette[6], "black"), cex=c(0.5,0.1), main="")
```



Note that most of these changes are relatively slight (<2-fold)

```
# Extract all results from limma (n=Inf)
all_results <- topTable(total_fit_lm_e_c, coef = "conditionM", n = Inf)

# we'll make a coupld of volcano plots so easier to wrap this up into a function
my_volcanoplot <- function(topTableResults){
  p <- topTableResults %>%
    mutate(sig=ifelse(adj.P.Val<0.01, "sig.", "not sig.")) %>% # add "sig" column
    ggplot(aes(logFC, -log10(P.Value), colour=sig)) +
    geom_point(size=0.25) +
    scale_colour_manual(values=c("black", cbPalette[6]), name="") # manually adjust colours

  return(p)
}

my_volcanoplot(all_results)
```



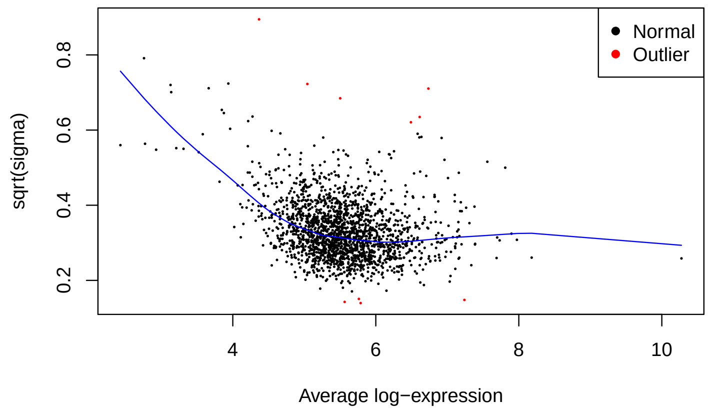
OK, so it's easy to perform the pairwise comparison. What about changes in RNA binding? For this, we need combine the two MSnSets into a single ExpressionSet

```
intersecting_proteins <- intersect(rownames(total_protein_quant), rownames(rbp_protein_quant))

total_for_combination <- total_protein_quant[intersecting_proteins,]
rbp_for_combination <- rbp_protein_quant[intersecting_proteins,]

# make the column names for the two MSnSets unique
colnames(total_for_combination) <- paste0(colnames(total_for_combination), "_Total")
colnames(rbp_for_combination) <- paste0(colnames(rbp_for_combination), "_OOPS")

# make the ExpressionSet
combined_intensities <- ExpressionSet(cbind(exprs(total_for_combination), exprs(rbp_for_combination)))

# Add the feature data
fData(combined_intensities) <- fData(total_for_combination)

# Add the phenotype data
pData(combined_intensities) <- rbind(pData(total_for_combination), pData(rbp_for_combination))

pData(combined_intensities)$Condition <- factor(pData(combined_intensities)$Condition, level=c("M","G1"
pData(combined_intensities)$Type <- factor(pData(combined_intensities)$Type, level=c("Total","OOPS"))
```

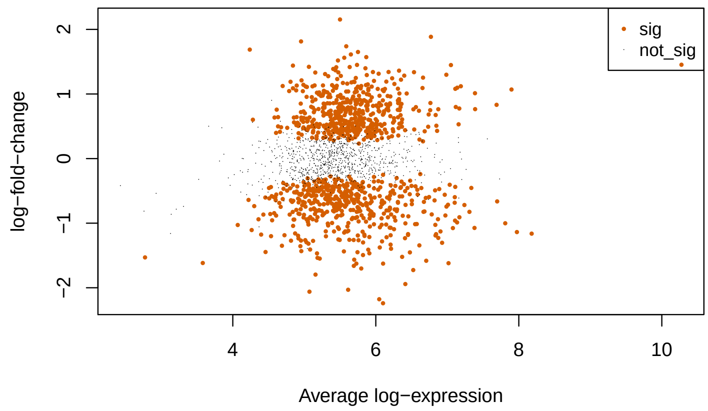
The we run `limma` on the combined intensities and this time test for a signficant interaction coefficient

```
condition <- combined_intensities$Condition
type <- combined_intensities$Type
sample_name <- combined_intensities$Sample_name

design <- model.matrix(~condition*type)

rna_binding_fit <- lmFit(combined_intensities, design)

rna_binding_fit <- contrasts.fit(rna_binding_fit, coefficients="conditionG1:typeOOPS")

rna_binding_fit <- eBayes(rna_binding_fit, trend=TRUE, robust=TRUE)
```

```
plotSA(rna_binding_fit)
```



```
rna_binding_p_value_status <- ifelse(rna_binding_fit$p.value[,'conditionG1:typeOOPS']<0.01, "sig", "not

limma::plotMA(rna_binding_fit, status=rna_binding_p_value_status, values=c("sig", "not_sig"),
              col=c(cbPalette[6], "black"), cex=c(0.5,0.1), main="")
```



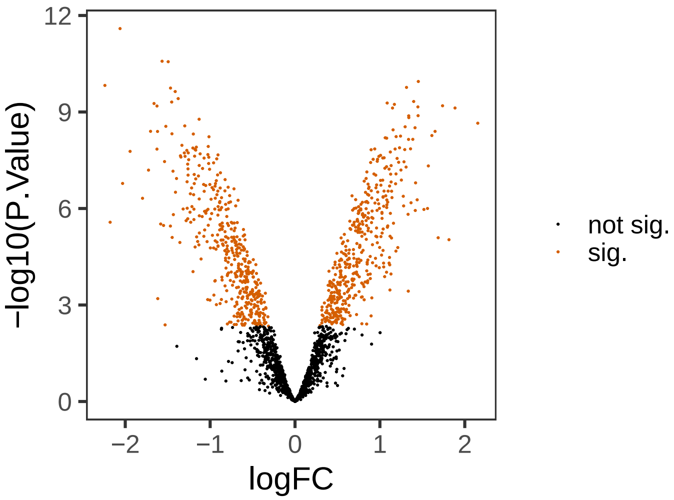
Below, we summarise the number of signficant p-values (post BH FDR correction) using a 1% FDR threshold.

```
summary(decideTests(rna_binding_fit, p.value=0.01, adjust.method="BH"))
```

```
##        conditionG1:typeOOPS
## Down                    439
## NotSig                 1036
## Up                      441
```

And then make the volcano plot for the RNA binding changes

```
all_rna_binding_results <- topTable(rna_binding_fit, coef = "conditionG1:typeOOPS", n = Inf, confint=TRU

my_volcanoplot(all_rna_binding_results)
```



So again, lots of RNA binding changes. Nearly 50% of the proteins show a change in RNA binding!

Now, let's compare the results from the two methods. To do this, we will merge together the results from the two methods.

```
M_G1_simple_lm <- readRDS("../results/M_G1_changes_in_RNA_binding_linear_model.rds")

compare_methods <- all_rna_binding_results %>%
  dplyr::select("logFC", "AveExpr", "adj.P.Val", "P.Value") %>%
  merge(M_G1_simple_lm, by.x="row.names", by.y="protein")
```

First, let's tabulate the proteins significant in each method

```
lm_sig <- ifelse(compare_methods$lm_BH<0.01, "lm sig", "lm insig")
limma_sig <- ifelse(compare_methods$adj.P.Val<0.01, "limma sig", "limma insig")

print(table(lm_sig, limma_sig))
```

```
##            limma_sig
## lm_sig      limma insig limma sig
##   lm insig         1016       140
##   lm sig             20       740
```

```
compare_methods$sig_status <- interaction(lm_sig, limma_sig)
```

OK, so most proteins with significant change in RNA binding using `lm` or `limma` are significant in both, although `limma` does indicate more proteins have a significant change. Note that only 20/1916 proteins are significant by `lm` only.

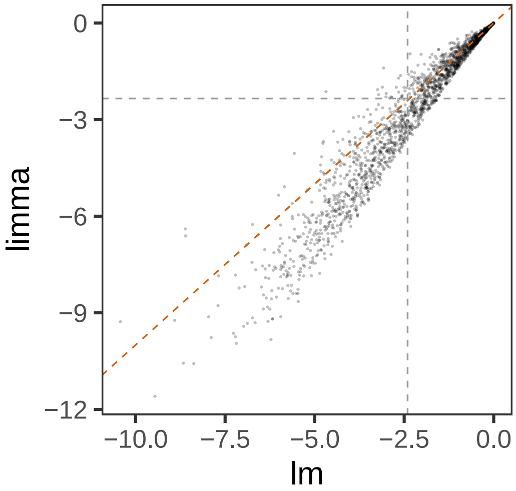
First, let's compare the p-values. Note that the p-values are usually lower in `limma`

```
ifelse(compare_methods$P.Value<compare_methods$lm_p_value, "limma_lower", "lm_lower") %>% table()
```

```
## .
## limma_lower    lm_lower
##        1459         457
```

```
# identify the maxmimum p.value which gives an adjusted p.value < 0.01 for each method
max_p_sig_lm <- compare_methods %>% filter(lm_BH<0.01) %>% pull(lm_p_value) %>% max()
max_p_sig_limma <- compare_methods %>% filter(adj.P.Val<0.01) %>% pull(P.Value) %>% max()

# plot p-values
```

```
compare_methods %>%
  ggplot(aes(log10(lm_p_value), log10(P.Value))) +
  geom_point(alpha=0.25, size=0.25) +
  geom_abline(slope=1, linetype=2, colour=cbPalette[6]) +
  xlab("lm") +
  ylab("limma")  +
  geom_vline(xintercept=log10(max_p_sig_lm), linetype=2, colour="grey60") +
  geom_hline(yintercept=log10(max_p_sig_limma), linetype=2, colour="grey60")
```



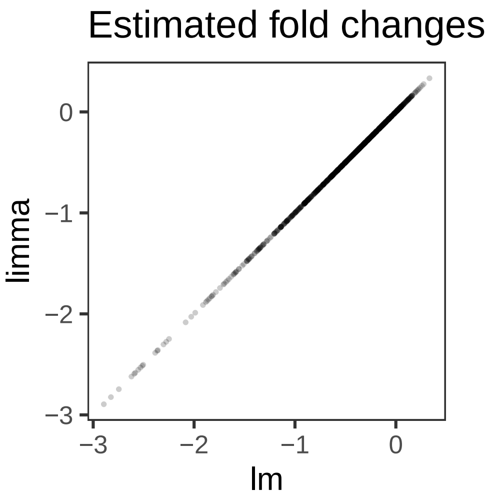
Note that the estimates for the fold change are the same with `lm` and `limma`

```
print(table(compare_methods$lm_fold_change==compare_methods$logFC))
```

```
##
## TRUE
## 1916
```

```
compare_methods %>% ggplot(aes(log10(lm_fold_change), log10(logFC))) +
  geom_point(size=1, alpha=0.2) +
  xlab("lm") +
  ylab("limma") +
  ggtitle("Estimated fold changes")
```

```
## Warning in FUN(X[[i]], ...): NaNs produced

## Warning in FUN(X[[i]], ...): NaNs produced

## Warning in FUN(X[[i]], ...): NaNs produced

## Warning in FUN(X[[i]], ...): NaNs produced

## Warning: Removed 1005 rows containing missing values (geom_point).
```



3.2 Question: Why is the fold change the same with lm and limma?

Finally, let's explore some of the proteins which were detected as having a significant change in RNA binding with only one method. Remember from above that the p-values for lm and limma are very well correlated so we're looking here at slight differences close to the 1% FDR thresholds.

First, we need a function to plot the intensities for a protein(s)

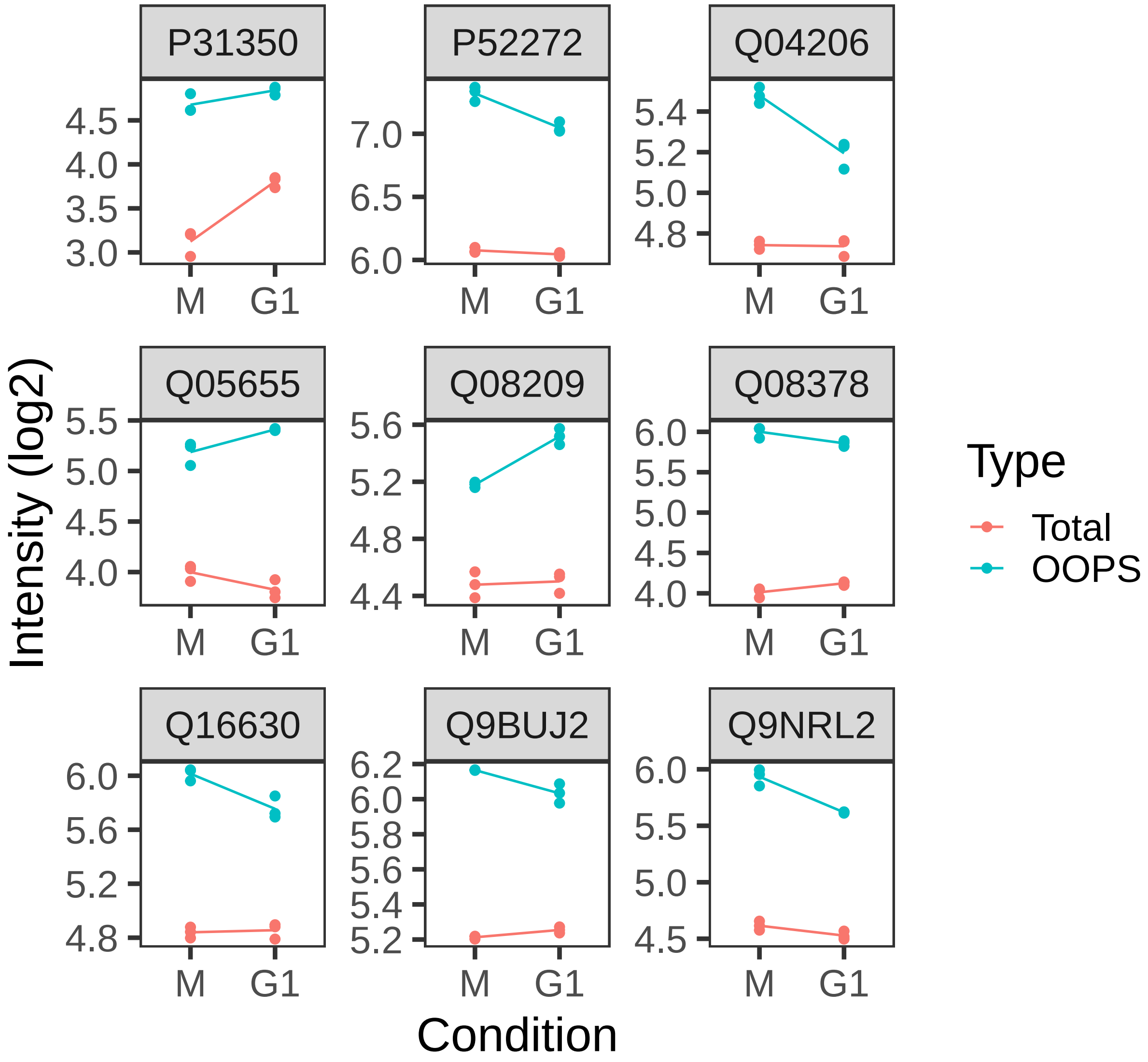
```
# Function to plot the intensities values
plotIntensities <- function(obj){

  p <- tidy(obj, addPheno=TRUE) %>%
    ggplot(aes(Condition, value, colour=Type, group=Type)) +
    geom_point() +
    stat_summary(geom="line", fun.y=mean) +
    facet_wrap(~gene, scales='free') +
    ylab("Intensity (log2)")

  invisible(p)
}
```

Let's look at some of the proteins which are "lm only". Notice that in all cases, the replicates are very tightly distributed.

```
lm_only <- compare_methods %>%
  filter(sig_status=='lm sig.limma insig') %>% # sig in lm only
  dplyr::select(Row.names, lm_fold_change, P.Value, adj.P.Val, lm_p_value, lm_BH, lm_std_error) %>% # s
  arrange(desc(P.Value)) # arrange by limma p-value (descending order)

print(head(lm_only, 2))
```

```
##   Row.names lm_fold_change    P.Value  adj.P.Val   lm_p_value       lm_BH
## 1    Q9BUJ2     -0.1742925 0.03992611 0.06594691 0.0008460464 0.002879263
## 2    Q9NRL2     -0.2279263 0.02325841 0.04130038 0.0025156687 0.006955298
##   lm_std_error
## 1    0.0336679
## 2    0.0526521
```

```
print(plotIntensities(combined_intensities[lm_only$Row.names[1:9],]))
```



In some cases, the intensity values are near identical (see below).

```
# Heterogeneous nuclear ribonucleoprotein U-like protein 1
# HNRNPUL1
# Acts as a basic transcriptional regulator. Represses basic transcription driven by several virus and
# When associated with BRD7, activates transcription of glucocorticoid-responsive promoter in the absen
# ligand-stimulation. Plays also a role in mRNA processing and transport. Binds avidly to poly(G) and p
# homopolymers in vitro.

tidy(combined_intensities, addPheno=TRUE) %>%
  filter(gene=="Q9BUJ2", Type=="OOPS", Condition=="M") %>%
  dplyr::select(Condition, Replicate, value)
```

```
## # A tibble: 3 x 3
##   Condition Replicate value
##   <fct>     <chr>     <dbl>
## 1 M         1          6.17
## 2 M         2          6.16
## 3 M         3          6.17
```

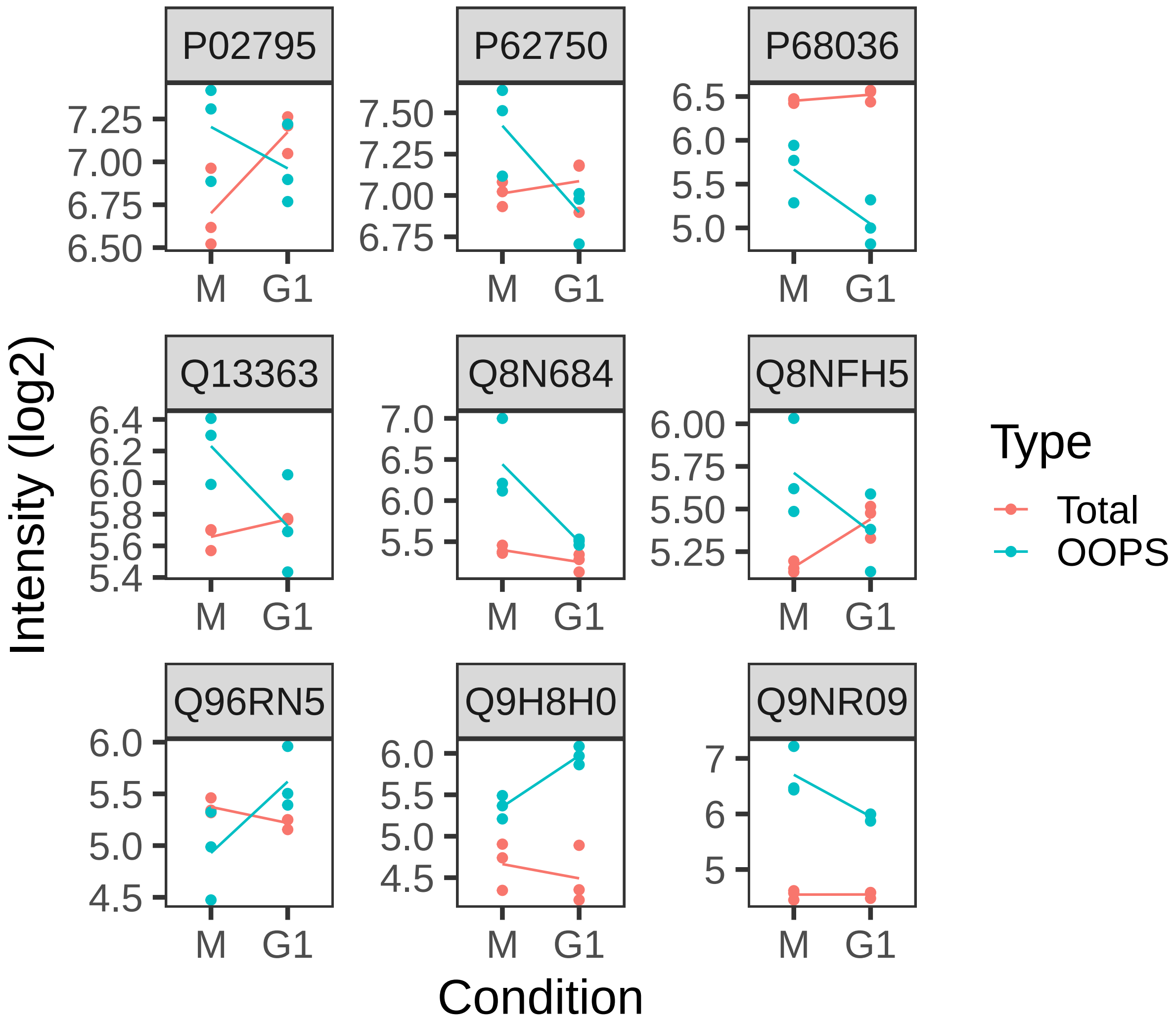
3.3 Questions: Why would the intensity values for this protein be so similar?

Below, we explore the observed intensity values for 9 of the proteins which are significant according only to `limma`. Note that these have relatively large variability in comparison.

```
limma_only <- compare_methods %>%
  filter(sig_status=='lm insig.limma sig') %>%
  dplyr::select(Row.names, lm_fold_change, P.Value, adj.P.Val, lm_p_value, lm_BH, lm_std_error) %>%
  arrange(desc(lm_p_value))

print(head(limma_only, 2))
```

```
##   Row.names lm_fold_change     P.Value   adj.P.Val lm_p_value      lm_BH
## 1    Q96RN5      0.8446206 0.003908340 0.008627166 0.02501420 0.04586335
## 2    Q8N684     -0.7967253 0.003884313 0.008592135 0.02494248 0.04577566
##   lm_std_error
## 1    0.3070057
## 2    0.2894011
```

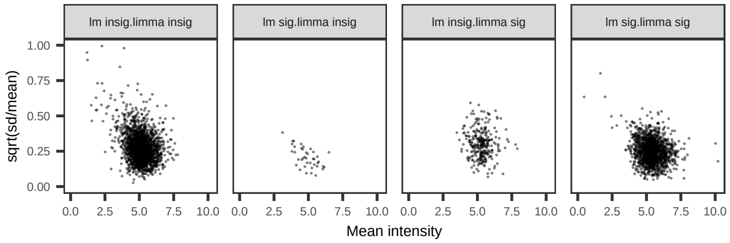
```
print(plotIntensities(combined_intensities[limma_only$Row.names[1:9],]))
```



Let's go back to that plot of mean vs sqrt and see how the proteins compare depending on whether they were detected as signficant in each method.

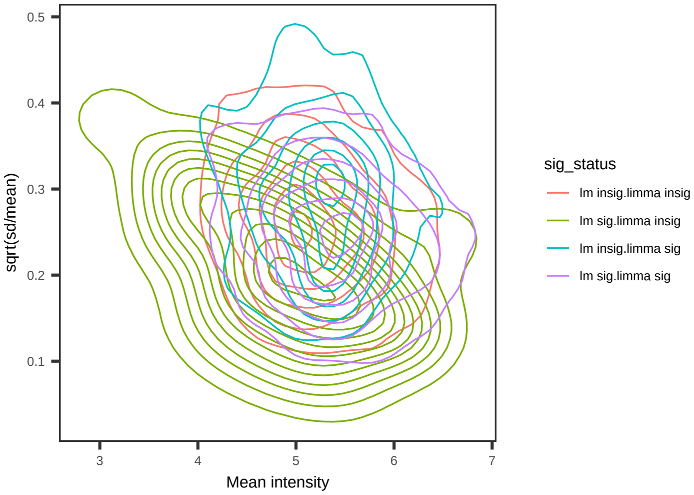
```
mean_sd_data <- tidy(total_protein_quant, addPheno=TRUE) %>%# "tidy" the object, e.g make it into a tid
  group_by(protein, Condition) %>% # group by protein and condition
  dplyr::summarise(mean=mean(value), sqrt_sd=sqrt(sd(value))) %>% # mean and stdev for each group
  ungroup() %>%
  merge(compare_methods, by.x="protein", by.y="Row.names")  # merge in the results from the two methods

mean_sigma <- mean_sd_data %>%
  ggplot(aes(mean, sqrt_sd)) +
  xlab("Mean intensity") +
  ylab("sqrt(sd/mean)") +
  theme(text=element_text(size=10))
```

```
mean_sigma_point <- mean_sigma + geom_point(size=0.2, alpha=0.5) +
  facet_grid(.~sig_status) +
  xlim(0, NA) + ylim(0, NA) # include 0,0
print(mean_sigma_point)
```



We can show this with a density plot too

```
mean_sigma_density <- mean_sigma +  geom_density_2d(aes(colour=sig_status))
print(mean_sigma_density)
```



3.4 Question: Why do the proteins which are detected as having a significant change with lm only have a lower standard deviation?

Finally, we can get `limma` to return the signficant changes using both p-value and log-fold change thresholds using the TREAT method (https://www.ncbi.nlm.nih.gov/pubmed/19176553). Note that this is not the same as thresholding on the p-value and the point estimate for the fold change as limma is actually testing the null hypothesis that the fold change is less than our specified threshold. To be explicit, let's check the difference

```
# Subset to proteins passing a 1% FDR threshold with limma
sig_changes_p <- topTable(rna_binding_fit, coef = "conditionG1:typeOOPS", n = Inf, p.value=0.01, adjust

# Then subset by absolute fold change > 2
sig_changes_p_logfc_point_estimate <- sig_changes_p[abs(sig_changes_p$logFC)>1,]

# how many proteins in the sequential subsets
cat(sprintf("%s proteins pass adjusted p-value threshold, of which %s pass the threshold on 2 fold chang
            nrow(sig_changes_p), nrow(sig_changes_p_logfc_point_estimate)))
```

```
## 880 proteins pass adjusted p-value threshold, of which 202 pass the threshold on 2 fold change point
```

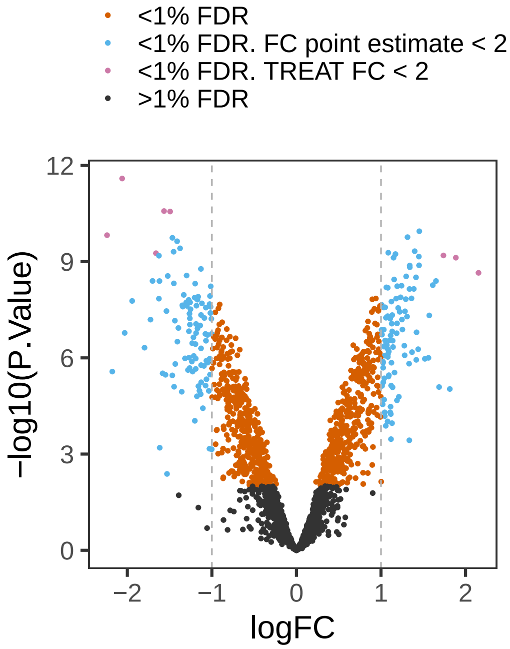
```
# Test null hypothesis than change is <2-fold
rna_binding_fit_treat <- treat(rna_binding_fit,lfc=1)

# subset to proteins passing the 1% FDR threshold for absolute fold change > 2
sig_changes_p_logfc <- topTreat(rna_binding_fit_treat, coef = "conditionG1:typeOOPS", n = Inf,
                                p.value=0.01, lfc=1, adjust.method="fdr", confint=0.95)
cat(sprintf("%s proteins pass the combined adjusted p-value threshold + fold change > 2\n",  nrow(sig_cl
```

```
## 8 proteins pass the combined adjusted p-value threshold + fold change > 2
```

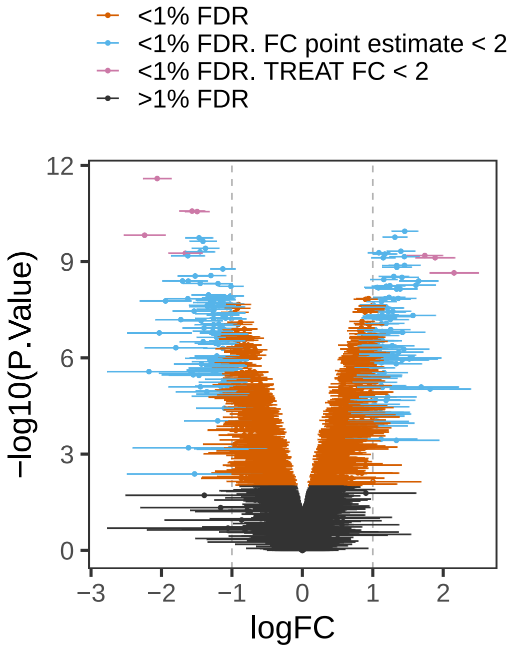
And below we reproduce our volcano plot including the 95% confidence interval and highlight those proteins which have < 1% FDR and an absolute fold change significantly greater than 2. Below, we can see that many of the proteins with a fold change (FC) point estimate > 2 have a 95% confidence interval that overlaps the dashed lines for >2-fold change. TREAT also takes the multiple testing into account so it's even more conservative than just using the 95% CI shown below.

Of course, the threshold for the fold changes you are interested in depends entirely on your prior expectations.

```
.tmp_df <- all_rna_binding_results
.tmp_df$sig <- ifelse(.tmp_df$P.Value<=0.01, "<1% FDR", ">1% FDR") # add "sig" column
.tmp_df$sig[rownames(.tmp_df) %in% rownames(sig_changes_p_logfc_point_estimate)] <- "<1% FDR. FC point e
.tmp_df$sig[rownames(.tmp_df) %in% rownames(sig_changes_p_logfc)] <- "<1% FDR. TREAT FC < 2"
.tmp_df$SE <- sqrt(rna_binding_fit$s2.post) * rna_binding_fit$stdev.unscaled[,1]

p <- .tmp_df %>%
  ggplot(aes(logFC, -log10(P.Value), colour=sig)) +
  geom_point(size=1) +
  scale_colour_manual(values=c(cbPalette[c(6,2,7)], "grey20"), name="") + # manually adjust colours
  geom_vline(xintercept=1, linetype=2, colour="grey70") +
  geom_vline(xintercept=-1, linetype=2, colour="grey70") +
  theme(legend.position="top", legend.direction=2)

print(p)
```



```
print(p + geom_errorbarh(aes(xmin=CI.L, xmax=CI.R)))
```



Finally, let’s save out the results objects for later notebooks.

```
saveRDS(rna_binding_fit, "../results/limma_rna_binding_fit.rds")

saveRDS(all_rna_binding_results, "../results/limma_rna_binding_results.rds")

saveRDS(rna_binding_fit_treat, "../results/limma_rna_binding_results_treat.rds")
```

```r
saveRDS(compare_methods, "../results/compare_methods_rna_binding_results.rds")

saveRDS(combined_intensities, "../results/combined_intensities.rds")
```