Using LIMMA in proteomics

Here, we will explore the use of LIMMA ("linear models for microarray data") for performing linear modelling.

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/

To save repeating the work of other in describing the use of `limma`, I refer you to this introduction from Kasper D. Hansen: https://kasperdanielhansen.github.io/genbioconductor/html/limma.html.

Note that the data used in the above is microarray data in an `ExpressionSet` object. However, limma is agnostic to the type of input data and is perfectly suitable for proteomics data so long as it's reasonable to assume the quantification values are approximately gaussian distributed. For this reason, the quantification values should first be log transformed.

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 read in the MSnSets and subset to the samples of interest

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

# identify the samples we want to keep
samples_to_keep <- grep("M_|G1_", pData(total_protein_quant)$Sample_name)
print(samples_to_keep)
```

```
## [1] 2 3 4 5 6 7
```

```
total_protein_quant <- total_protein_quant[,samples_to_keep]
rbp_protein_quant <- rbp_protein_quant[,samples_to_keep]
```

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)
```

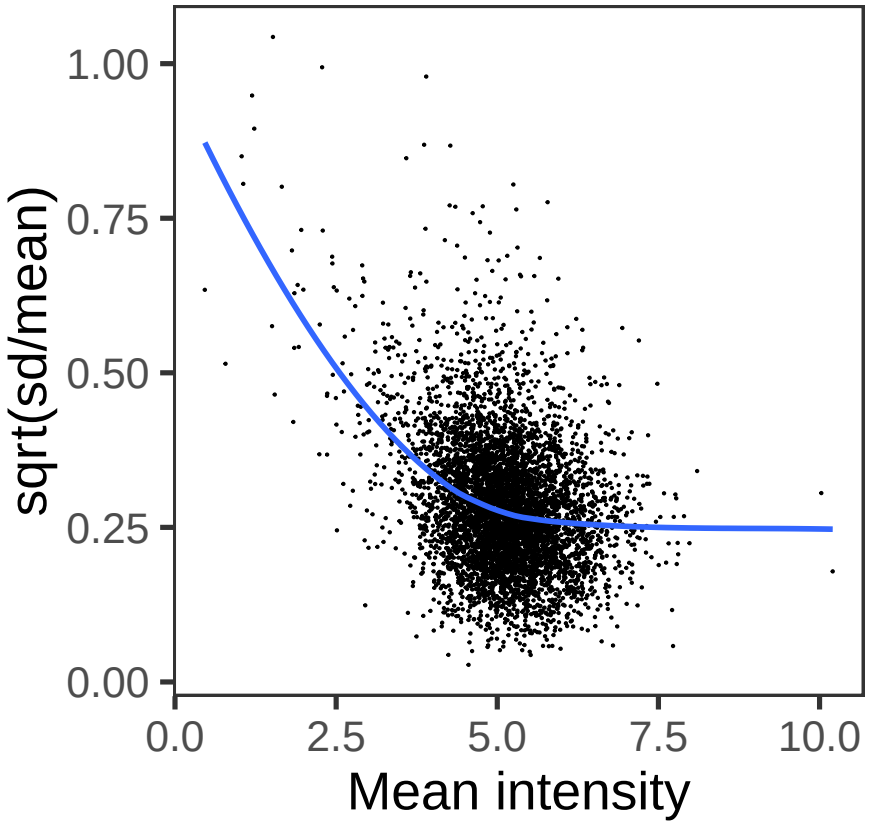
```
##   (Intercept) conditionM
## 1           1          1
## 2           1          1
## 3           1          1
## 4           1          0
## 5           1          0
## 6           1          0
## attr(,"assign")
## [1] 0 1
## attr(,"contrasts")
## attr(,"contrasts")$condition
## [1] "contr.treatment"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

Then we fit the model using this design and update the estimates for the standard errors for each coefficient using the `eBayes` function. As expected, there is a relationship between mean intensity and variance, although this is almost all limited to the very low intensity values.

Questions: - Why do we expect a relationship between mean intensity and variance?

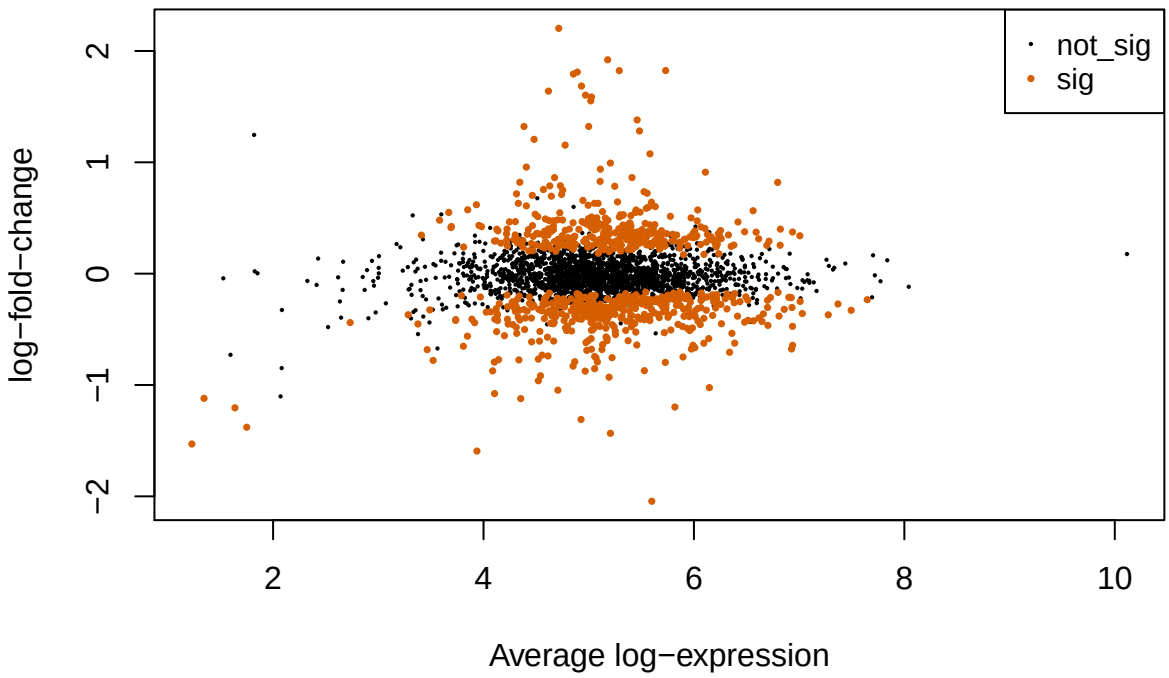
Limma will use this relationship to moderate the standard errors for the coefficients estimated such that the per-protein variance estimates are "squeezed" towards the expectation derived from other proteins with similar mean intensity.

```
tidy(total_protein_quant, addPheno=TRUE) %>%# "tidy" the object, e.g make it into a tidy data format --
  group_by(protein, Condition) %>% # group by protein and condition
  dplyr::summarise(mean=mean(value), sqrt_sd=sqrt(sd(value))) %>% # mean and var for each group
  ggplot(aes(mean, sqrt_sd)) + # plot mean(intensity) vs sqrt(sd(intensity))
  geom_point(size=0.1) +
  geom_smooth(se=FALSE, method="loess") + # local regression
  xlab("Mean intensity") +
  ylab("sqrt(sd/mean)")
```



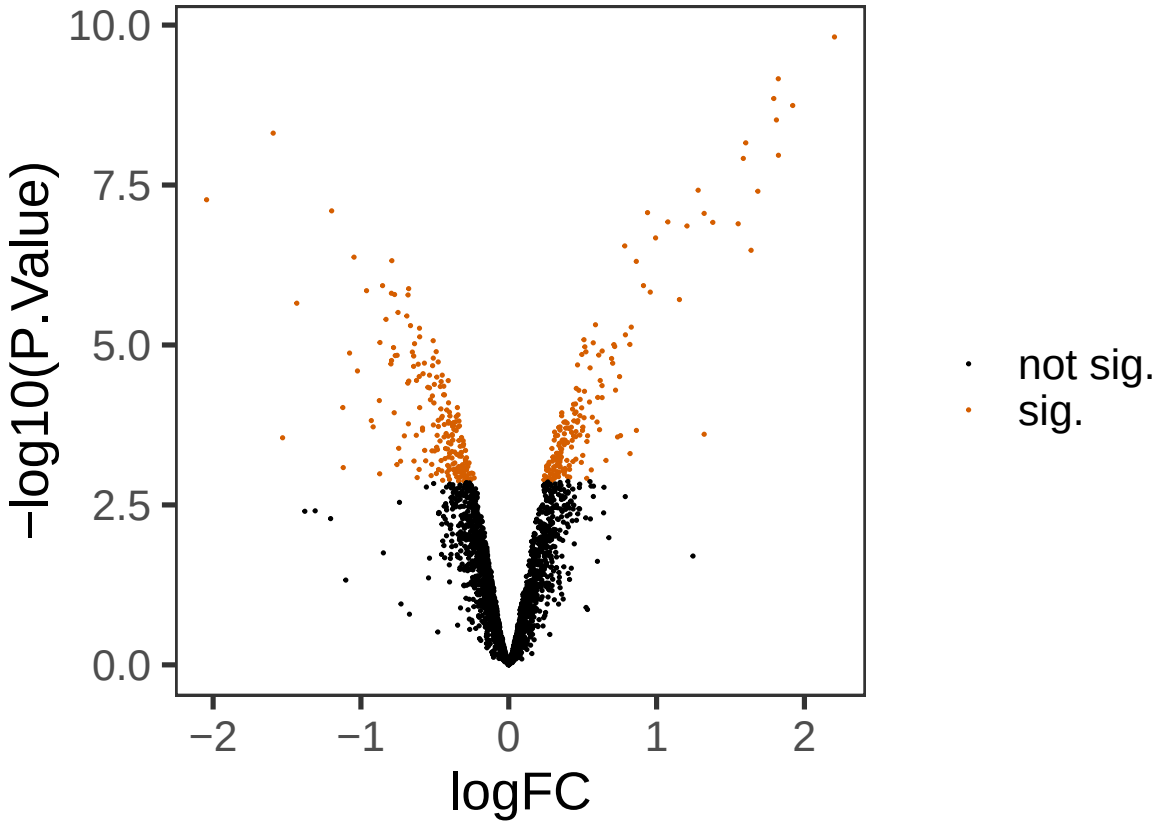
Below we run limma to idenify the proteins with a significant change in abundance between conditions

```
total_fit_lm <- lmFit(exprs(total_protein_quant), design) # fit model to each protein
total_fit_lm_e <- eBayes(total_fit_lm) # shrink std errors to abundance vs. stdev trend

total_fit_lm_e_c <- contrasts.fit(total_fit_lm_e, coefficients=c("conditionM")) # extract results for c

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

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)
```



Questions:

- Where are most of the data points in a volcano plot?
- Can you estimate what propotion of proteins had a significant change in abundance?
- Why does the plot look so symetrical?

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)
```

```r
# 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"))

dim(exprs(combined_intensities))
```

```
## [1] 1916   12
```

```r
print(head(data.frame(exprs(combined_intensities)), 2))
```

```
##        Abundance.F1.127N.Sample_Total Abundance.F1.127C.Sample_Total
## A0AVT1                       5.379489                       5.296457
## A1L0T0                       3.787581                       4.416164
##        Abundance.F1.128N.Sample_Total Abundance.F1.128C.Sample_Total
## A0AVT1                       5.356980                       5.444870
## A1L0T0                       3.992692                       4.010076
##        Abundance.F1.129N.Sample_Total Abundance.F1.129C.Sample_Total
## A0AVT1                       5.532511                       5.411776
## A1L0T0                       4.064894                       3.799227
##        Abundance.F1.127N.Sample_OOPS Abundance.F1.127C.Sample_OOPS
## A0AVT1                      5.707774                      5.830157
## A1L0T0                      4.610035                      4.481285
##        Abundance.F1.128N.Sample_OOPS Abundance.F1.128C.Sample_OOPS
## A0AVT1                      5.771748                      6.602972
## A1L0T0                      4.421321                      4.911068
##        Abundance.F1.129N.Sample_OOPS Abundance.F1.129C.Sample_OOPS
## A0AVT1                      6.558550                      6.449456
## A1L0T0                      5.083586                      4.654846
```

```r
print(head(fData(combined_intensities), 2))
```

```
##        Checked Confidence   Sequence
## A0AVT1   False       High ACIGDTLCQK
## A1L0T0   False       High   AAMGLGAR
##                                          Modifications Qvality.PEP
## A0AVT1 2xTMT6plex [N-Term; K10]; 2xCarbamidomethyl [C2; C8] 0.000118596
## A1L0T0                                1xTMT6plex [N-Term]   0.0248863
##        Qvality.q.value Number.of.Protein.Groups Number.of.Proteins
## A0AVT1     5.56241e-05                       1                  1
## A1L0T0      0.00319759                       1                  2
##        Number.of.PSMs Master.Protein.Accessions Number.of.Missed.Cleavages
## A0AVT1              3                    A0AVT1                          0
## A1L0T0              2                    A1L0T0                          0
##        Theo.MHplus.in.Da Quan.Info Amanda.Score.MS.Amanda
## A0AVT1    1623.85986925    Unique       330.291304522885
## A1L0T0     975.56070138    Unique
##        Confidence.MS.Amanda Search.Space.MS.Amanda
## A0AVT1                 High                 1206.0
## A1L0T0
##        Percolator.q.Value.MS.Amanda Percolator.PEP.MS.Amanda
```

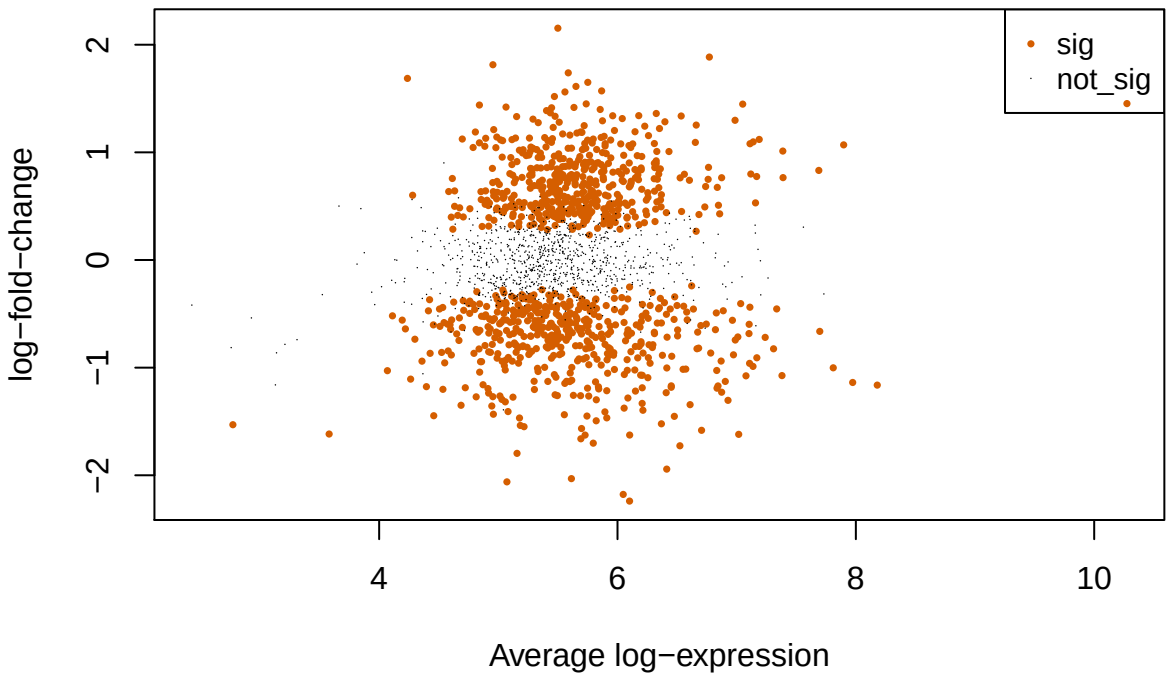
```
## A0AVT1                    9.831e-05   0.00015900000000000002
## A1L0T0
##          XCorr.Sequest.HT Confidence.Sequest.HT Search.Space.Sequest.HT
## A0AVT1   3.72960662841797                  High
## A1L0T0 2.3773212432861297                  High
##        Percolator.q.Value.Sequest.HT Percolator.PEP.Sequest.HT
## A0AVT1                     7.671e-05                 0.0001077
## A1L0T0         0.0012779999999999998                   0.01145
##        Ions.Score.Mascot Confidence.Mascot Search.Space.Mascot
## A0AVT1             71.39              High
## A1L0T0             19.82              High
##        Percolator.q.Value.Mascot  Percolator.PEP.Mascot master_protein
## A0AVT1                 8.725e-05 0.00010700000000000001         A0AVT1
## A1L0T0                    0.0026                   0.02         A1L0T0
##        protein_length   protein_description peptide_start peptide_end
## A0AVT1           1052  sp|A0AVT1|UBA6_HUMAN           447         457
## A1L0T0            632 sp|A1L0T0|ILVBL_HUMAN           577         585
##        crap_protein associated_crap_protein unique
## A0AVT1            0                       0      1
## A1L0T0            0                       0      1
##                               filename CV.Abundance.F1.126.Sample
## A0AVT1 Nocodazole_Total_PeptideGroups.txt                0.08912426
## A1L0T0 Nocodazole_Total_PeptideGroups.txt                0.04624279
##        CV.Abundance.F1.127N.Sample CV.Abundance.F1.127C.Sample
## A0AVT1                  0.08717847                   0.1025999
## A1L0T0                  0.07954851                   0.1520306
##        CV.Abundance.F1.128N.Sample CV.Abundance.F1.128C.Sample
## A0AVT1                  0.12057913                  0.06179052
## A1L0T0                  0.06404729                  0.02991015
##        CV.Abundance.F1.129N.Sample CV.Abundance.F1.129C.Sample
## A0AVT1                  0.11457832                  0.04857286
## A1L0T0                  0.03921029                  0.06404895
##        CV.Abundance.F1.130N.Sample CV.Abundance.F1.130C.Sample
## A0AVT1                  0.05473802                   0.1359917
## A1L0T0                  0.10704177                   0.2304991
##        CV.Abundance.F1.131.Sample
## A0AVT1                 0.16328462
## A1L0T0                 0.09747969
```

```
print(head(pData(combined_intensities), 2))
```

```
##                                Sample_name Condition Replicate  Type
## Abundance.F1.127N.Sample_Total         M_1         M         1 Total
## Abundance.F1.127C.Sample_Total         M_2         M         2 Total
```

```
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)

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="")
```



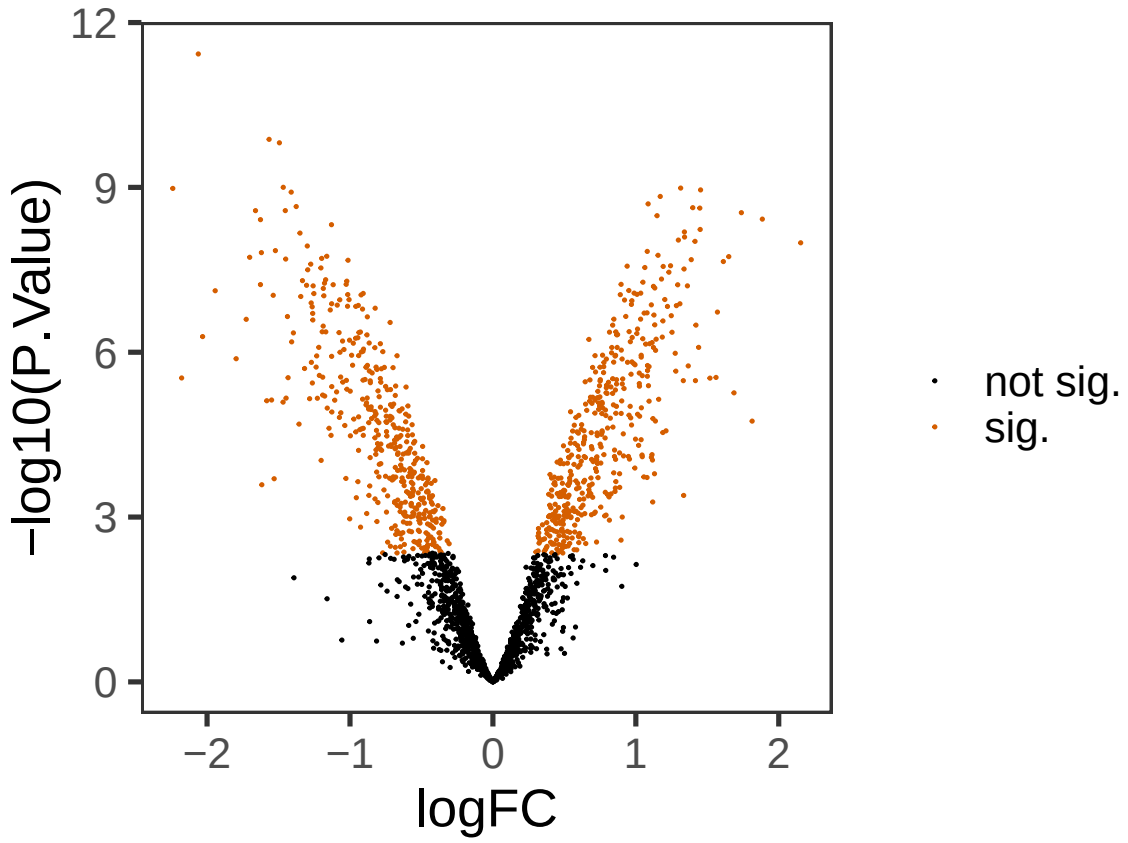
Below, we summarise the number of signficant p-values (post BH FDR correction) using a 1% FDR threshold.

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

```
##        conditionG1:typeOOPS
## Down                    432
## NotSig                 1047
## Up                      437
```

And then make another volcano plot

```
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!

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$lm_BH<0.01, "lm sig", "lm not sig")
limma_sig <- ifelse(compare_methods$adj.P.Val<0.01, "limma sig", "limma not sig")

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

```
##             limma_sig
## lm_sig       limma not sig limma sig
##   lm not sig          1022       201
##   lm sig                25       668
```

```
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 25/1916 proteins are significant by `lm` only.

What about if we separate by the `lm` model used, e.g +/- tag

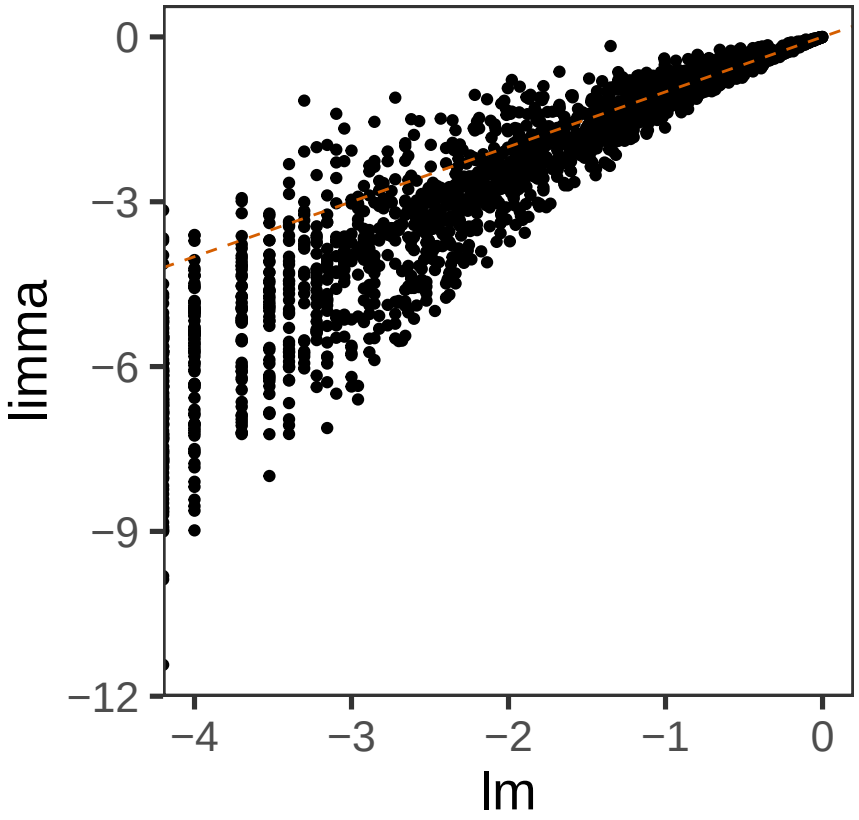
```
fit <- compare_methods$fit
print(table(lm_sig, limma_sig, fit))
```

```
## , , fit = With_tag
##
##             limma_sig
## lm_sig       limma not sig limma sig
##   lm not sig           541       139
##   lm sig                18       349
##
## , , fit = Without_tag
##
##             limma_sig
## lm_sig       limma not sig limma sig
##   lm not sig           481        62
##   lm sig                 7       319
```

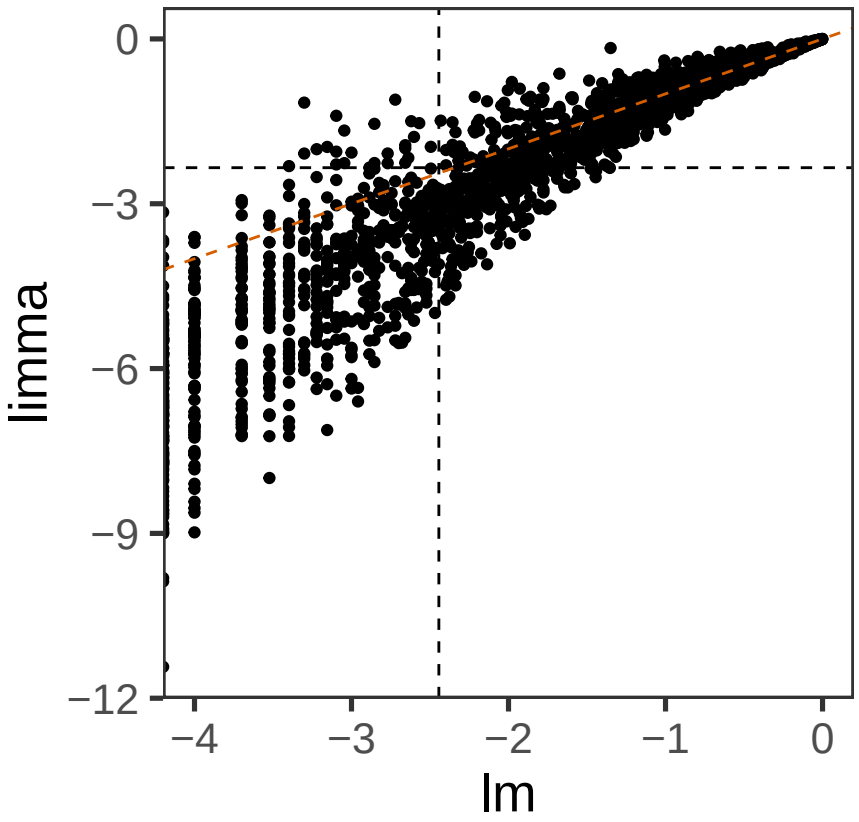
OK, so `lm` and `limma` results are more similar if the `lm` model did not include the tag. This is no suprise since the `limma` fomula we used did not include the tag covariate so this is the closest comparison

First, let's compare the p-values. Note that the p-values are usually lower in `limma`. The second plot shows the threshold for the maximum p-value which results in an estimated FDR $< 1\%$ for both methods.

```
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()

p <- compare_methods %>%
  ggplot(aes(log10(lm_p_value), log10(P.Value))) +
  geom_point() +
  geom_abline(slope=1, linetype=2, colour=cbPalette[6]) +
  xlab("lm") +
  ylab("limma")

print(p)
```



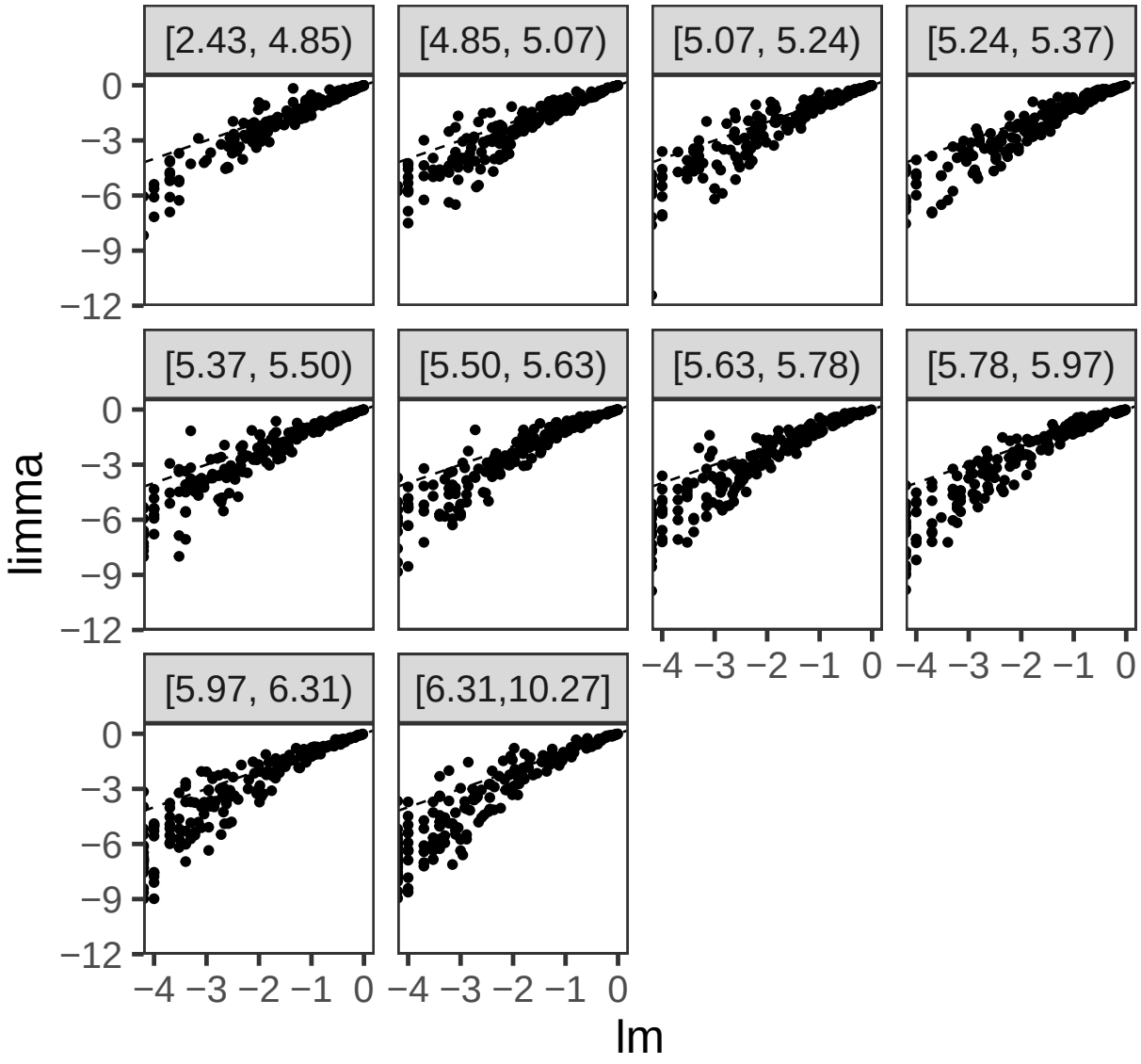

```
print(p +
  geom_vline(xintercept=log10(max_p_sig_lm), linetype=2) +
  geom_hline(yintercept=log10(max_p_sig_limma), linetype=2))
```



Task: Copy the cell above into a new cell and modify the plotting code to indicate which lm model was used

What about if we separate the proteins by their intensity?

```
compare_methods %>%
  mutate(binned_ave_exprs=cut2(AveExpr, g=10)) %>% # bin the AveExpr using Hmisc::cut()
  ggplot(aes(log10(lm_p_value), log10(P.Value))) +
  geom_point() +
  geom_abline(slope=1, linetype=2) +
  xlab("lm") +
  ylab("limma") +
  facet_wrap(~binned_ave_exprs)
```



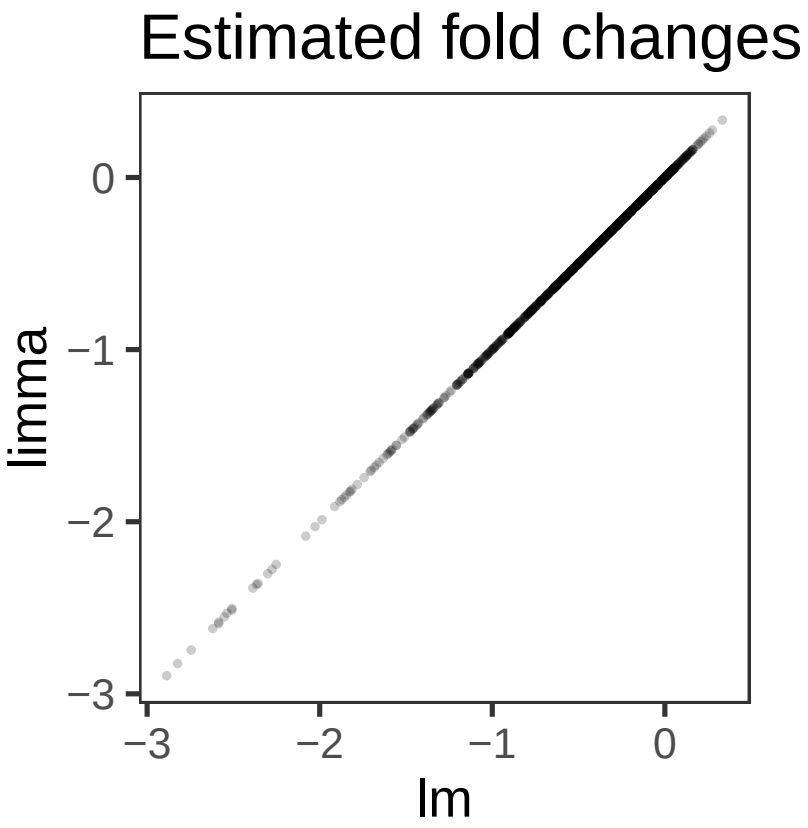
Question: Why are the p-values more similar for low intensity proteins?

Note that the estimates for the fold change are not changed by the bayesian shrinkage of the coefficient standard errors.

```
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).
```



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)
}
```

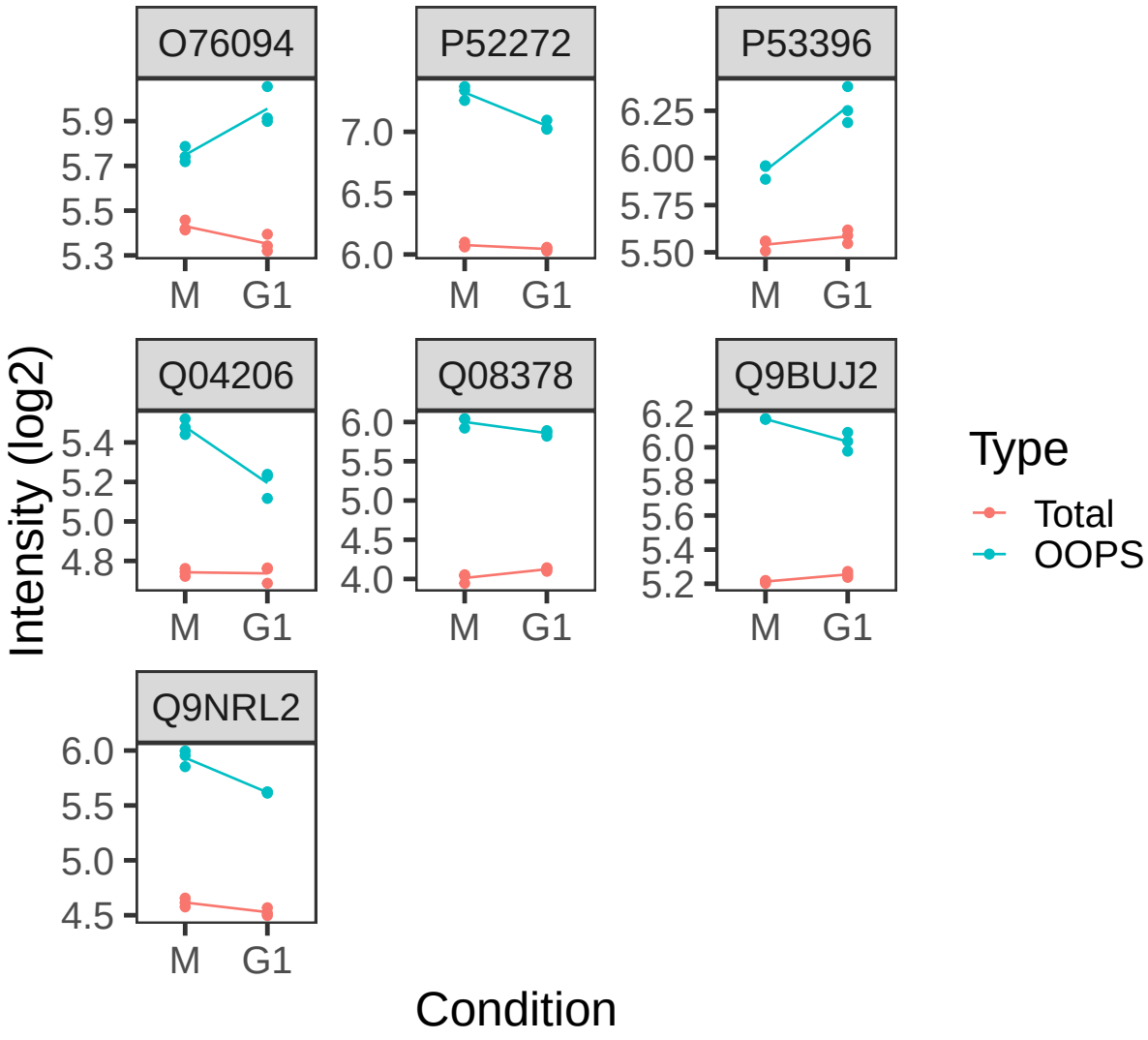
Let's look at the proteins which are "lm only". We'll ignore those where we used the TMT tag in our lm model since this was not included in the limma model so that may be another reason for differences in the p-values.

```
lm_only <- compare_methods %>%
  filter(fit=="Without_tag") %>% # Restrict to protein where lm model did not include the tag
  filter(sig_status=='lm sig.limma not sig') %>% # 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(lm_only)
```

```
##   Row.names lm_fold_change     P.Value  adj.P.Val lm_p_value       lm_BH
## 1    Q9BUJ2        -0.1743 0.039877481 0.06575323     0.0008 0.003452252
## 2    Q9NRL2        -0.2279 0.016460997 0.03062065     0.0025 0.007676282
## 3    Q08378        -0.2546 0.010208665 0.02014398     0.0022 0.007037062
## 4    P52272        -0.2401 0.009836109 0.01944890     0.0006 0.002866833
## 5    P53396         0.2946 0.005566362 0.01178469     0.0022 0.007037062
## 6    O76094         0.2849 0.005505155 0.01168782     0.0014 0.005013832
## 7    Q04206        -0.2784 0.005191059 0.01115030     0.0008 0.003452252
##   lm_std_error
## 1       0.0337
## 2       0.0527
## 3       0.0575
## 4       0.0437
## 5       0.0664
## 6       0.0596
## 7       0.0530
```

Now let's plot these proteins. Notice that in all cases, the replicates are very tightly distributed.

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



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
```

Questions:

- Why would the intensity values for this protein be so similar?

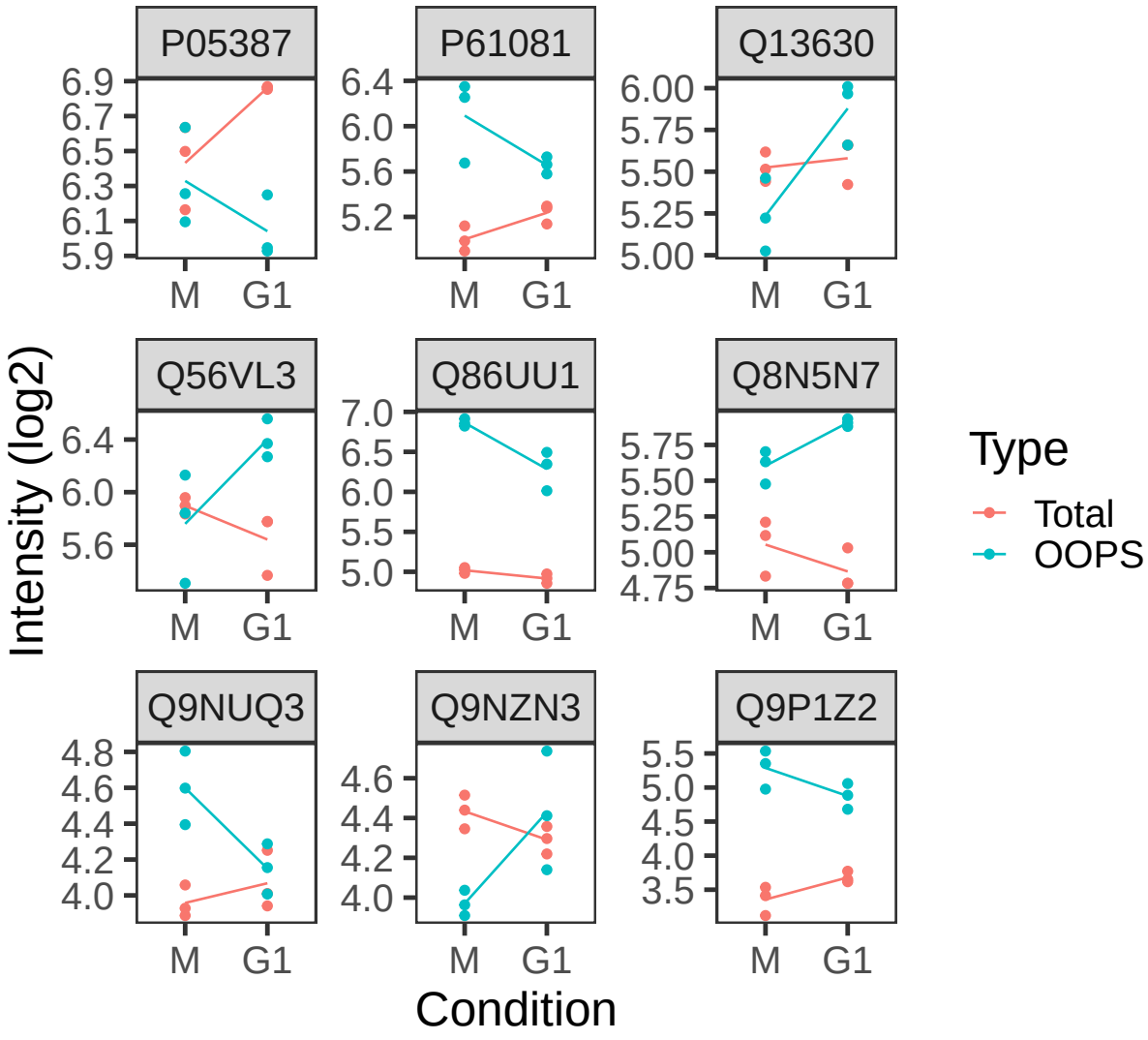
My answer: While it's possible the biological variability for this protein is very low, it seems unlikely that the exact same amount of RNA-bound protein was recovered given the expected technical variability from the OOPS protocol and sample preparation. A much more likely explanation is that these intensity values are so similar simply by chance. This is the (reasonable) assumption by which 'limma' alters the standard deviations for the coefficients using features with similar abundance.

Below, we explore the observed intensity values for some 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 not sig.limma sig', fit=="Without_tag") %>%
  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))
```

```
##   Row.names lm_fold_change     P.Value   adj.P.Val lm_p_value      lm_BH
## 1    P61081        -0.6705 0.004497911 0.009940019     0.0194 0.03683885
## 2    P05387        -0.7171 0.003400573 0.007775058     0.0164 0.03190091
## 3    Q13630         0.5868 0.003956296 0.008865806     0.0156 0.03065600
## 4    Q56VL3         0.8967 0.002630243 0.006237061     0.0153 0.03025263
## 5    Q9P1Z2        -0.7361 0.002861809 0.006719640     0.0146 0.02929173
## 6    Q9NUQ3        -0.5587 0.003760201 0.008475935     0.0142 0.02860904
##   lm_std_error
## 1       0.2300
## 2       0.2368
## 3       0.1919
## 4       0.2917
## 5       0.2371
## 6       0.1789
```

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



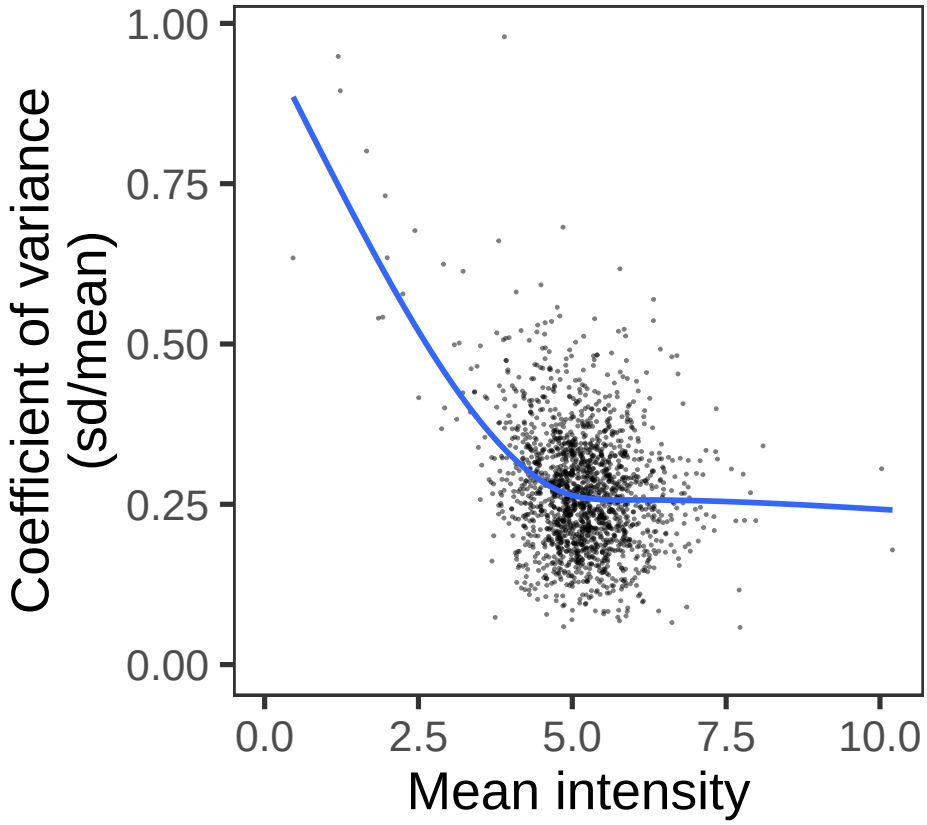
Let's go back to that plot of mean vs sqrt and see how the proteins compare depending on whether they were detected as signficant in each method.

As expected, those significant in just `lm` have relatively low observed std. dev. and those significant in just `limma` have relatively high std. dev.

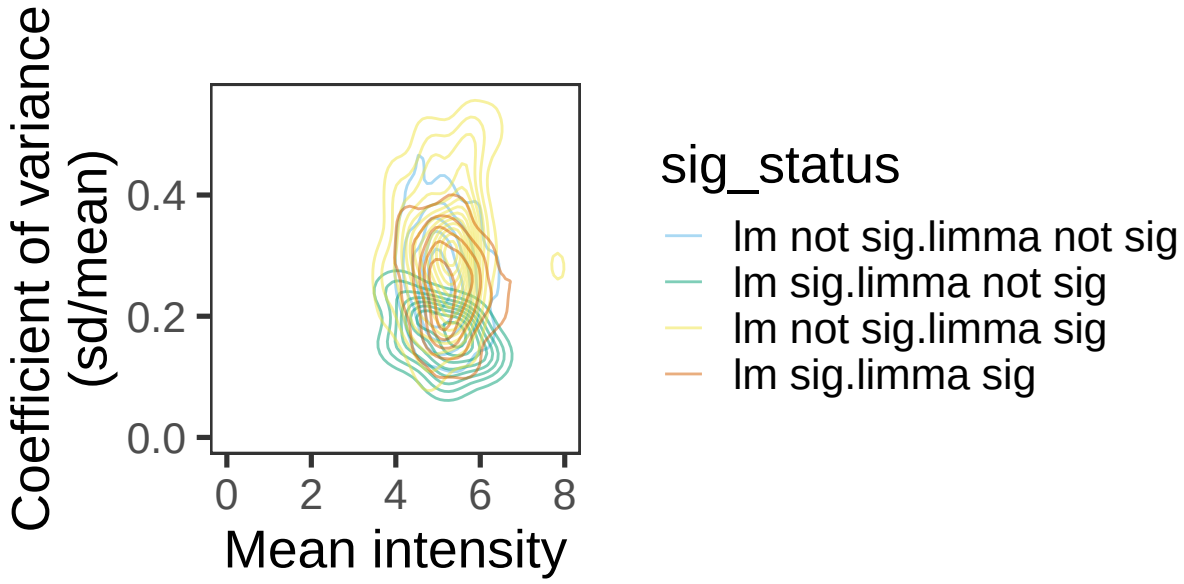
```
mean_sd_data <- tidy(total_protein_quant, addPheno=TRUE) %>%# "tidy" the object, e.g make it into a tid
  group_by(protein, Condition) %>% # group by protein and condition
  dplyr::summarise(mean=mean(value), sqrt_sd=sqrt(sd(value))) %>% # mean and stdev for each group
  ungroup() %>%
  merge(compare_methods, by.x="protein", by.y="Row.names") %>% # merge in the results from the two meth
  filter(fit=="Without_tag") # Only keep those proteins fitted without the tag covariate in lm

# remake the basic plot showing the relationship
p_basic <- mean_sd_data %>%
```

```
  ggplot(aes(mean, sqrt_sd)) +
  xlab("Mean intensity") +
  ylab("Coefficient of variance\n(sd/mean)") +
  xlim(0, NA) + ylim(0, NA) # include 0,0

print(p_basic + geom_point(size=0.2, alpha=0.5) + geom_smooth(se=FALSE))
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```
p_density <- p_basic +
  geom_density_2d(aes(colour=sig_status), alpha=0.5, size=0.5) + # density of points
  scale_colour_manual(values=cbPalette[c(2:4,6)])# set colours

print(p_density)
```



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

```
sig_changes_p <- topTable(rna_binding_fit, coef = "conditionG1:typeOOPS", n = Inf, p.value=0.01, adjust
sig_changes_p_logfc_point_estimate <- sig_changes_p[abs(sig_changes_p$logFC)>1,]
cat(sprintf("%s proteins pass adjusted p-value threshold, of which %s pass the threshold on 2 fold chan
            nrow(sig_changes_p), nrow(sig_changes_p_logfc_point_estimate)))
```

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

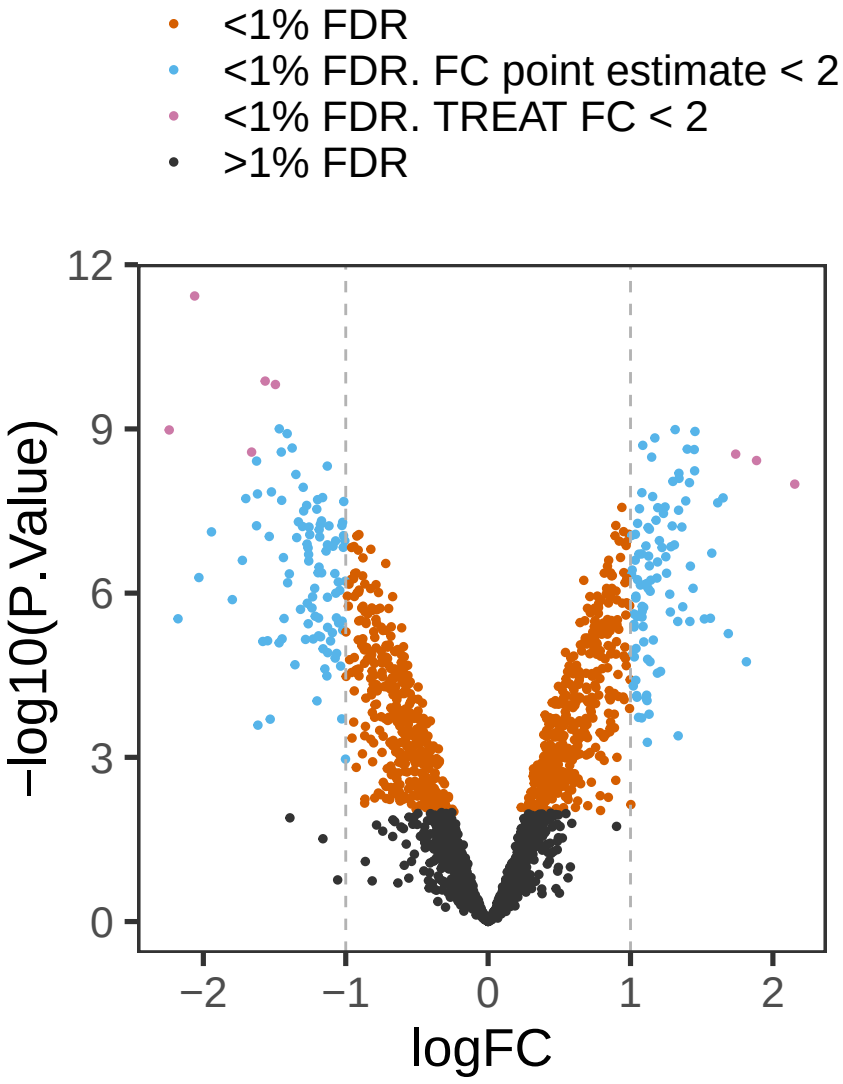
```
rna_binding_fit_treat <- treat(rna_binding_fit,lfc=1) # Test null hypothesis than change is <2-fold
sig_changes_p_logfc <- topTreat(rna_binding_fit_treat, coef = "conditionG1:typeOOPS", n = Inf,
                                p.value=0.01, lfc=1, adjust.method="fdr", confint=0.95)
cat(sprintf("%s 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
```

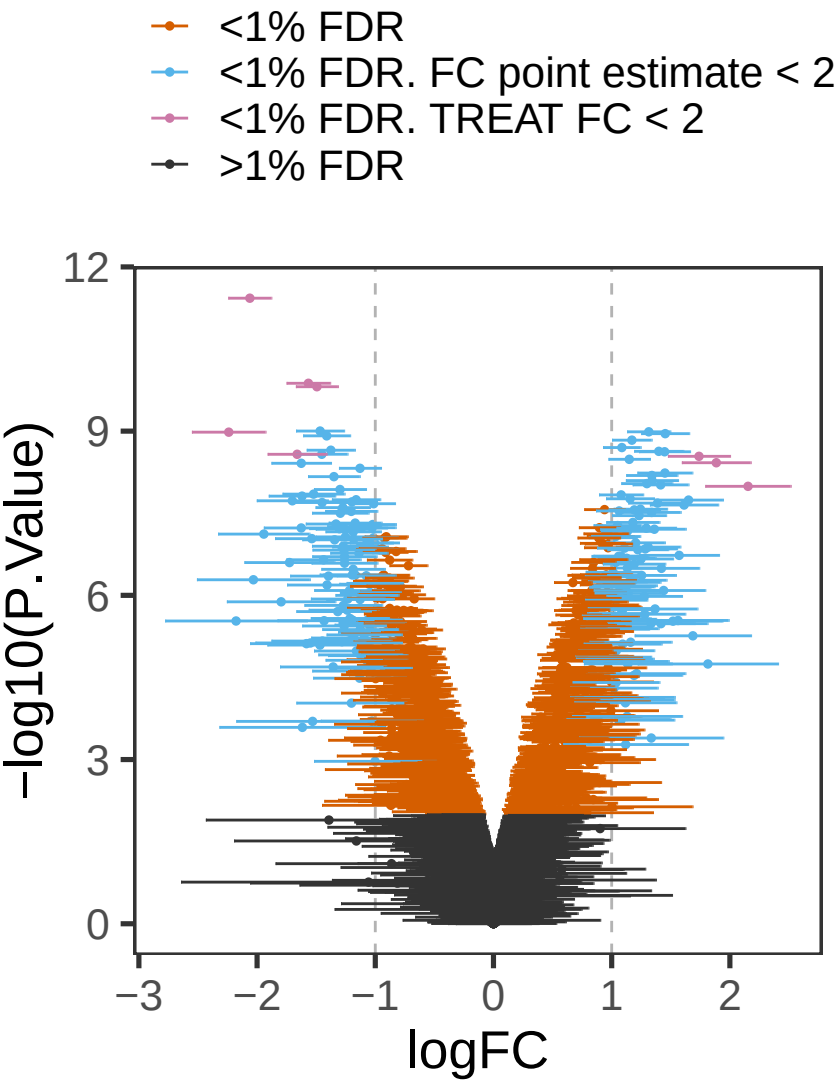
And below we reproduce our volcano plot including the 95% confidence interval and highlight those proteins which have $< 1\%$ FDR and an absolute fold change significantly greater than 2. Below, we can see that many of the proteins with a fold change (FC) point estimate > 2 have a 95% confidence interval 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.

```r
.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$sig[rownames(.tmp_df) %in% rownames(sig_changes_p_logfc_point_estimate)] <- "<1% FDR. FC point
.tmp_df$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")

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

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