# Identifying changes in RNA binding using a linear model

In this notebook, we will apply the linear model approach presented in (`1_simple_example`) to a real-life data set.

```r
# load packages
library(tidyverse)
library(biobroom)
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")
```

We start by reading in the data. Our input here is the protein-level quantification for the Nocodazole arrest/release experiment conducted for the OOPS NBT paper. In this experiment, we wanted to assess changes in RNA binding in arrested/released cells. To do this, we quantified "total" protein abundance and RNA-bound (extracted by OOPS) protein abundance. The peptide-level abundances have been aggregated to protein level abundance and center-median normalised. Proteins with missing values have been removed.

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

The input data are in MSnSets. As a reminder, the `MSnSet` class mimics the `ExpressionSet` class and contains 3 matrices: 1. assay data (obtained via: `exprs`) 2. feature data (`fData`) 3. phenotype data (`pData`)

The assay data is the quantification of the features (PSMs/peptides/proteins) and contains one column per sample

The feature data describes each feature, e.g peptide sequence, master protein accession, retention time etc etc

The phenotype data describes the samples

Let's take a look at our total protein quantification data. If we "print" the object, we get a summary including the processing steps performed.

Here we have 2761 features (proteins) quantified across 6 samples.

The `varLabels` describe the "Condition", "Replicate" and "Type" for these samples. We'll take a look at these in more detail shortly.

We can see that there were originally 20171 features which were combined into 18111 features using a user-defined function. This was the step at which peptides with the same sequence but different variable modifications were aggregated. Then, these 18111 features were combined into the 2761 features (peptide->protein aggregation). Finally, the data was center-median normalised and missing values with imputed with knn.

```r
print(total_protein_quant)
```

```
## MSnSet (storageMode: lockedEnvironment)
## assayData: 2761 features, 6 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: Abundance.F1.127N.Sample Abundance.F1.127C.Sample ...
##     Abundance.F1.129C.Sample (6 total)
##   varLabels: Sample_name Condition Replicate Type
##   varMetadata: labelDescription
## featureData
##   featureNames: A0AVT1 A0MZ66 ... Q9Y6Y8 (2761 total)
##   fvarLabels: Checked Confidence ... CV.Abundance.F1.131.Sample (47
##     total)
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
## Annotation:
## - - - Processing information - - -
## Subset [20171,10][20171,10] Thu Aug  9 16:17:13 2018
## Combined 20171 features into 18111 using user-defined function: Thu Aug  9 16:17:32 2018
## Combined 18111 features into 2761 using user-defined function: Thu Aug  9 16:17:46 2018
## Normalised (center.median): Thu Aug  9 16:17:51 2018
## Data imputation using knn Thu Aug  9 16:17:58 2018
##   Using default parameters
## Subset [2761,10][2761,6] Wed Jun 12 15:16:25 2019
##  MSnbase version: 2.4.2
```

Here's the top of the assay data

```
print(dim(exprs(total_protein_quant)))
```

```
## [1] 2761    6
```

```
print(head(exprs(total_protein_quant), 2))
```

```
##        Abundance.F1.127N.Sample Abundance.F1.127C.Sample
## A0AVT1                 5.379489                 5.296457
## A0MZ66                 4.926415                 4.876517
##        Abundance.F1.128N.Sample Abundance.F1.128C.Sample
## A0AVT1                 5.356980                 5.444870
## A0MZ66                 4.955341                 4.735426
##        Abundance.F1.129N.Sample Abundance.F1.129C.Sample
## A0AVT1                 5.532511                 5.411776
## A0MZ66                 4.886217                 4.834322
```

… and the associated feature data. Notice that there are many columns in the feature data. These are all the additional columns output from PD in addition to the quantification. They are all stored here in case they are required.

```
print(head(fData(total_protein_quant), 2))
```

```
##        Checked Confidence       Sequence
## A0AVT1   False       High     ACIGDTLCQK
## A0MZ66   False       High ATQPETTEEVTDLK
##                                                Modifications Qvality.PEP
## A0AVT1 2xTMT6plex [N-Term; K10]; 2xCarbamidomethyl [C2; C8] 0.000118596
## A0MZ66                             2xTMT6plex [N-Term; K14] 0.000709357
##        Qvality.q.value Number.of.Protein.Groups Number.of.Proteins
## A0AVT1     5.56241e-05                        1                  1
## A0MZ66     0.000157133                        1                  1
##        Number.of.PSMs Master.Protein.Accessions Number.of.Missed.Cleavages
## A0AVT1              3                    A0AVT1                          0
## A0MZ66              3                    A0MZ66                          0
##        Theo.MHplus.in.Da Quan.Info Amanda.Score.MS.Amanda Confidence.MS.Amanda
## A0AVT1     1623.85986925    Unique       330.291304522885                 High
## A0MZ66     2020.08503667    Unique       342.226818892066                 High
##        Search.Space.MS.Amanda Percolator.q.Value.MS.Amanda
## A0AVT1                 1206.0                    9.831e-05
## A0MZ66                 1908.0                    0.0001625
##        Percolator.PEP.MS.Amanda XCorr.Sequest.HT Confidence.Sequest.HT
## A0AVT1   0.00015900000000000002 3.72960662841797                  High
## A0MZ66                0.0004789 3.85038113594055                  High
##        Search.Space.Sequest.HT Percolator.q.Value.Sequest.HT
## A0AVT1                                             7.671e-05
## A0MZ66                                0.00014419999999999998
##        Percolator.PEP.Sequest.HT Ions.Score.Mascot Confidence.Mascot
## A0AVT1                 0.0001077             71.39              High
## A0MZ66                 0.0003799             32.36              High
##        Search.Space.Mascot Percolator.q.Value.Mascot  Percolator.PEP.Mascot
## A0AVT1                                     8.725e-05 0.00010700000000000001
## A0MZ66                                     0.0001791              0.0007973
##        master_protein protein_length   protein_description peptide_start
## A0AVT1         A0AVT1           1052  sp|A0AVT1|UBA6_HUMAN           447
## A0MZ66         A0MZ66            631 sp|A0MZ66|SHOT1_HUMAN           392
##        peptide_end crap_protein associated_crap_protein unique
## A0AVT1         457            0                       0      1
## A0MZ66         406            0                       0      1
##                                  filename CV.Abundance.F1.126.Sample
## A0AVT1 Nocodazole_Total_PeptideGroups.txt                 0.08912426
## A0MZ66 Nocodazole_Total_PeptideGroups.txt                 0.14097226
##        CV.Abundance.F1.127N.Sample CV.Abundance.F1.127C.Sample
## A0AVT1                  0.08717847                   0.1025999
## A0MZ66                  0.10111405                   0.1889357
##        CV.Abundance.F1.128N.Sample CV.Abundance.F1.128C.Sample
## A0AVT1                  0.12057913                  0.06179052
## A0MZ66                  0.08956286                  0.13456950
##        CV.Abundance.F1.129N.Sample CV.Abundance.F1.129C.Sample
## A0AVT1                  0.11457832                  0.04857286
## A0MZ66                  0.04768461                  0.03199063
##        CV.Abundance.F1.130N.Sample CV.Abundance.F1.130C.Sample
## A0AVT1                  0.05473802                   0.1359917
## A0MZ66                  0.13902945                   0.2397172
##        CV.Abundance.F1.131.Sample
## A0AVT1                  0.1632846
## A0MZ66                  0.1071883
```

… and here is the phenotype data. As we can see, we have 3 replicates each of "M", "G1" and "S" phase, plus an additional Control sample. For our purposes, we're only going to be interested in the M and G1 phases so we can remove the other data. Both the total and RBP quantification objects have the exact same order

```
print(pData(total_protein_quant))
```

```
##                             Sample_name Condition Replicate  Type
## Abundance.F1.127N.Sample         M_1         M         1 Total
## Abundance.F1.127C.Sample         M_2         M         2 Total
## Abundance.F1.128N.Sample         M_3         M         3 Total
## Abundance.F1.128C.Sample        G1_1        G1         1 Total
## Abundance.F1.129N.Sample        G1_2        G1          2 Total
## Abundance.F1.129C.Sample        G1_3        G1          3 Total
```

To detect changes in RNA binding, we can only consider RBPs where we have also quantified the total protein. Below, we identify these cases by intersecting the rownames of each MSnSet (the protein names)

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

cat(sprintf("Out of a total of %s RBPs quantified,\nwe have total protein quantification for %s protein
            length(rownames(rbp_protein_quant)),
            length(intersecting_proteins),
            round(100*length(intersecting_proteins)/length(rownames(rbp_protein_quant)), 2)))
```

```
## Out of a total of 2149 RBPs quantified,
## we have total protein quantification for 1916 proteins = 89.16 %
```

Below, we convert the MSnSet into a "tidy" format `data.frame` using `biobroom::tidy()`

```
total_exprs <- total_protein_quant[intersecting_proteins,] %>% # subset to intersecting proteins
  tidy(addPheno=TRUE)  %>% # "tidy" the object, e.g make it into a tidy data format --> long
  mutate(intensity=value) %>% dplyr::select(-value) # rename the "value" column -> "intensity"
```

Top of the total protein expression `data.frame`. See how each intensity value now has it's own row with the other columns describing the associated aspects of the intensity value, e.g the protein and experimental condition

```
print(head(total_exprs))
```

```
## # A tibble: 6 x 7
##   protein sample                   Sample_name Condition Replicate Type  intensity
##   <fct>   <fct>                    <chr>       <chr>     <chr>     <chr>     <dbl>
## 1 A0AVT1  Abundance.F1.127N.Sam~ M_1         M         1         Total      5.38
## 2 A1L0T0  Abundance.F1.127N.Sam~ M_1         M         1         Total      3.79
## 3 A1L390  Abundance.F1.127N.Sam~ M_1         M         1         Total      4.90
## 4 A1X283  Abundance.F1.127N.Sam~ M_1         M         1         Total      5.00
## 5 A5YKK6  Abundance.F1.127N.Sam~ M_1         M         1         Total      5.22
## 6 A6NFI3  Abundance.F1.127N.Sam~ M_1         M         1         Total      5.06
```

2.1 Question: Why doesn't the MSnSet object store all the data in this long format?

Now we do the same for the RBP quantification and then concatenate the two data frames together.

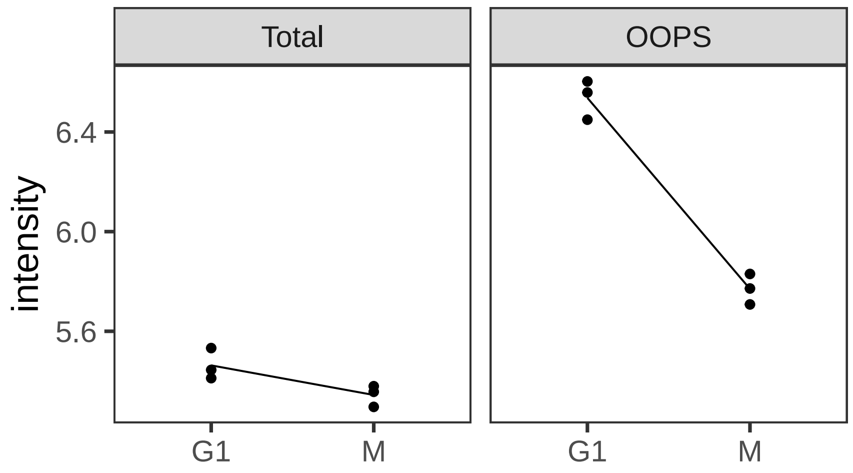
```
oops_exprs <- rbp_protein_quant[intersecting_proteins,] %>%
  tidy(addPheno=TRUE)  %>%
  mutate(intensity=value) %>% dplyr::select(-value)

combined_exprs <- rbind(total_exprs, oops_exprs)
```

We want to tell R which is the order of the values in the condition and type columns so that the fold changes are in the expected direction, e.g positive = higher in G1 vs M.

```
combined_exprs$condition <- factor(combined_exprs$Condition, levels=c("M", "G1"))
combined_exprs$type <- factor(combined_exprs$Type, levels=c("Total", "OOPS"))
```

Now we model the protein intensity according to the models described in `1_simple_example_vd.Rmd`. As an example, let's see the results from just applying the models to a single UniprotID.

```
combined_exprs %>% filter(protein == 'A0AVT1') %>%
  ggplot(aes(Condition, intensity, group=1)) +
  geom_point(size=2) +
  stat_summary(geom="line", fun.y=mean) +
  xlab("") +
  facet_wrap(~type)
```



```
fit <- combined_exprs %>% filter(protein == 'A0AVT1') %>%
  lm(formula=intensity~Condition*Type)

print(summary(fit))
```

```
##
## Call:
## lm(formula = intensity ~ Condition * Type, data = .)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.087537 -0.048708  0.007263  0.041451  0.069459
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)             6.53699    0.03620 180.590 9.89e-16 ***
## ConditionM             -0.76710    0.05119 -14.985 3.88e-07 ***
## TypeTotal              -1.07394    0.05119 -20.979 2.80e-08 ***
## ConditionM:TypeTotal    0.64836    0.07240   8.956 1.92e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0627 on 8 degrees of freedom
## Multiple R-squared:  0.988,  Adjusted R-squared:  0.9835
## F-statistic: 219.6 on 3 and 8 DF,  p-value: 5.07e-08
```

We can see that the model fits the data well (“Multiple R-squared: 0.9673, Adjusted R-squared: 0.955”). We can see that the interaction term that we’re interested in (for changes in RNA binding) significantly deviates from zero in both models.

Below, we make a function to run the linear models on a protein, select the best model and then return the required values from the model. When we run on the same protein as above, we can see that the best model is the one including the TMT tag as a co-variate.

```
Change_in_RNA_binding_LM <- function(obj, coeff_of_interest="conditionG1:typeOOPS"){

  fit <- obj %>% lm(formula=intensity ~ condition + type + condition*type)

  fit_values <- c(coef(summary(fit))[coeff_of_interest,],
                  summary(fit)$adj.r.squared)

  names(fit_values) <- c("lm_fold_change", "lm_std_error", "lm_t_value", "lm_p_value", "lm_adj_R_square
  fit_values <- as.data.frame(t(fit_values), stringsAsFactors=FALSE)

  return(fit_values)
}
```

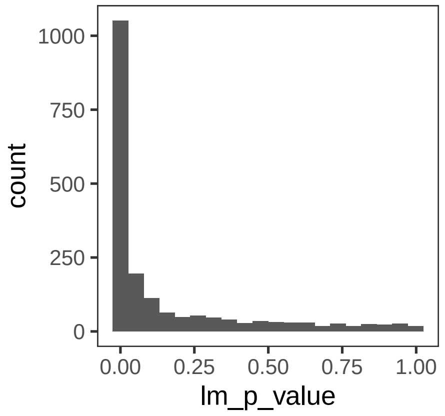
```
combined_exprs %>% filter(protein == 'A0AVT1') %>% Change_in_RNA_binding_LM()
```

```
##   lm_fold_change lm_std_error lm_t_value   lm_p_value lm_adj_R_squared
## 1      0.6483564   0.07239593   8.955702 1.921547e-05         0.983507
```

Below, we make a function to run the `testModels()` function on all proteins in turn using dplyr. We will use the standard Benjamini-Hochberg method to adjust p-values for the multiple tests we have conducted.

Below, we plot the p-values. Under the null hypothesis they should show an approximately uniform distribution. If there were a large number of proteins with a significant change in RNA binding, we would expect an additional “spike” with low p-values (<0.05). We see an approximately uniform distribution but with a slight skew towards low p-value. This may indicate the presence of changes in RNA binding but which we are insufficiently powered to detect, e.g low p-value but not significant low p-value.

```
M_G1 %>% ggplot(aes(lm_p_value)) + geom_histogram(bins=20)
```

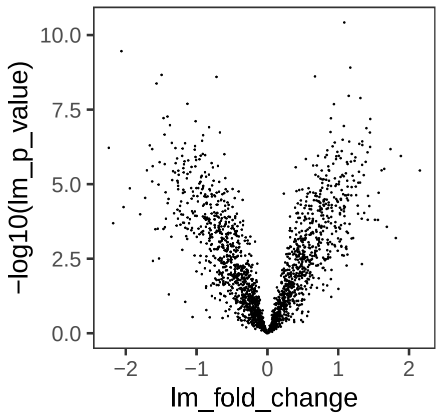


2.2 Task: How many significant changes in RNA binding were detected (You’ll need to settle on a suitable FDR threshold)?

So, we have detected a lot of proteins with a signficant change in RNA binding!!

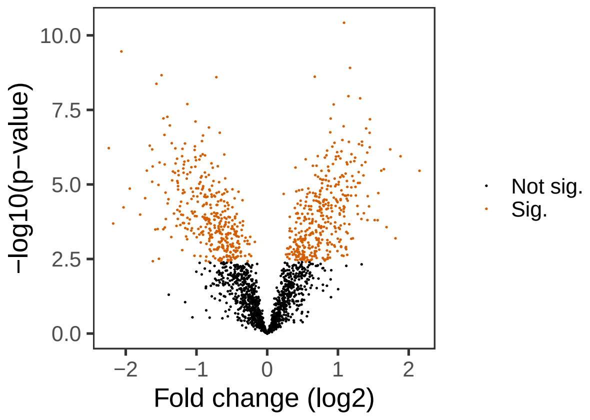
We can use a volcano plot to take a look at the estimated fold changes and associated p-values

```
M_G1 %>%
  ggplot(aes(x=lm_fold_change, y=-log10(lm_p_value))) +
  geom_point(size=0.25)
```



We can make this volcano plot a bit more informative (and prettier) with a few extra lines:

```
M_G1 %>%
  mutate(sig=ifelse(lm_BH<0.01, "Sig.", "Not sig.")) %>% # add "sig" column
  ggplot(aes(x=lm_fold_change, y=-log10(lm_p_value), colour=sig)) +
  geom_point(size=0.25) +
  scale_colour_manual(values=c("black", cbPalette[6]), name="") + # manually adjust colours
  xlab("Fold change (log2)") + ylab("-log10(p-value)") # manual axes labels
```



Finally, we save out the results for use in later notebooks

```
saveRDS(M_G1, "../results/M_G1_changes_in_RNA_binding_linear_model.rds")
```