

Introduction to proteomics data analysis: robust summarization

Lieven Clement

statOmics, Ghent University (<https://statomics.github.io>)

Contents

1	Background	1
2	Data	2
2.1	Data exploration	3
3	Preprocessing	3
3.1	Log transform the data	3
3.2	Filtering	3
3.3	Normalize the data using median centering	4
3.4	Explore normalized data	4
3.5	Summarization to protein level	6
4	Data Analysis	7
4.1	Estimation	7
4.2	Inference	31
4.3	Plots	31
5	Session Info	53
	References	55

This is part of the online course [Proteomics Data Analysis 2021 \(PDA21\)](#)

1 Background

A study on the facultative pathogen *Francisella tularensis* was conceived by Ramond et al. (2015) [12]. *F. tularensis* enters the cells of its host by phagocytosis. The authors showed that *F. tularensis* is arginine deficient and imports arginine from the host cell via an arginine transporter, ArgP, in order to efficiently escape from the phagosome and reach the cytosolic compartment, where it can actively multiply. In their study, they compared the proteome of wild type *F. tularensis* (WT) to ArgP-gene deleted *F. tularensis* (knock-out, D8). For this exercise, we use a subset of the *F. tularensis* dataset where bacterial cultures were grown in biological quadruplicate and each biorep was run in technical triplicate on a nanoRSLC-Q Exactive PLUS instrument. The data were searched with MaxQuant version 1.4.1.2. and are available on the PRIDE repository: [PXD001584](#).

2 Data

```
library(tidyverse)
library(limma)
library(QFeatures)
library(msqrob2)
library(plotly)

peptidesFile <- "https://raw.githubusercontent.com/statOmics/MSqRobSumPaper/master/Francisella/data/maximal"

ecols <- grep(
  "Intensity\\.\\.",
  names(read.delim(peptidesFile))
)

pe <- readQFeatures(
  table = peptidesFile,
  fnames = 1,
  ecol = ecols,
  name = "peptideRaw", sep="\t")

colnames(pe)
```

```
## CharacterList of length 1
## [["peptideRaw"]] Intensity.1WT_20_2h_n3_1 ... Intensity.3D8_20_2h_n5_3
```

The annotation can be derived from the file name.

```
colData(pe)$genotype <- substr(colnames(pe[[1]]),12,13) %>%
  as.factor
colData(pe)$biorep <- paste(
  substr(colnames(pe[[1]]),12,13),
  substr(colnames(pe[[1]]),21,22),
  sep=" ") %>% as.factor
```

We calculate how many non zero intensities we have per peptide and this is often useful for filtering.

```
rowData(pe[["peptideRaw"]])$nNonZero <- rowSums(assay(pe[["peptideRaw"]]) > 0)
```

Because every biorep is assessed in technical triplicate, we will also calculate the number of biorepeats in which each peptide is observed.

```
rowData(pe[["peptideRaw"]])$nNonZeroBiorep <- apply(
  assay(pe[["peptideRaw"]]),
  1,
  function(intensity)
    colData(pe)$biorep[intensity>0] %>%
    unique %>%
    length)
```

Peptides with zero intensities are missing peptides and should be represent with a NA value rather than 0.

```
pe <- zeroIsNA(pe, "peptideRaw") # convert 0 to NA
```

2.1 Data exploration

49% of all peptide intensities are missing.

3 Preprocessing

This section performs preprocessing for the peptide data. This includes

- log transformation,
- filtering and
- summarisation of the data.

3.1 Log transform the data

```
pe <- logTransform(pe, base = 2, i = "peptideRaw", name = "peptideLog")
```

3.2 Filtering

1. Handling overlapping protein groups

In our approach a peptide can map to multiple proteins, as long as there is none of these proteins present in a smaller subgroup.

```
pe[["peptideLog"]] <-  
  pe[["peptideLog"]][rowData(pe[["peptideLog"]])$Proteins  
  %in% smallestUniqueGroups(rowData(pe[["peptideLog"]])$Proteins),]
```

2. Remove reverse sequences (decoys) and contaminants

We now remove the contaminants and peptides that map to decoy sequences.

```
pe[["peptideLog"]] <- pe[["peptideLog"]][rowData(pe[["peptideLog"]])$Reverse != "+", ]  
pe[["peptideLog"]] <- pe[["peptideLog"]][rowData(pe[["peptideLog"]])$  
  Contaminant != "+", ]
```

3. Drop peptides that were only identified in a single biorepeat

Note, that in experiments without technical repeats we filter on the number of samples in which a peptide is picked up. Here, we will require that a peptide is picked up in at least two biorepeats.

```
pe[["peptideLog"]] <- pe[["peptideLog"]][rowData(pe[["peptideLog"]])$nNonZeroBiorep >= 2, ]  
nrow(pe[["peptideLog"]])
```

```
## [1] 7542
```

We keep 7542 peptides upon filtering.

3.3 Normalize the data using median centering

We normalize the data by subtracting the sample median from every intensity for peptide p in a sample i :

$$y_{ip}^{\text{norm}} = y_{ip} - \hat{\mu}_i$$

with $\hat{\mu}_i$ the median intensity over all observed peptides in sample i .

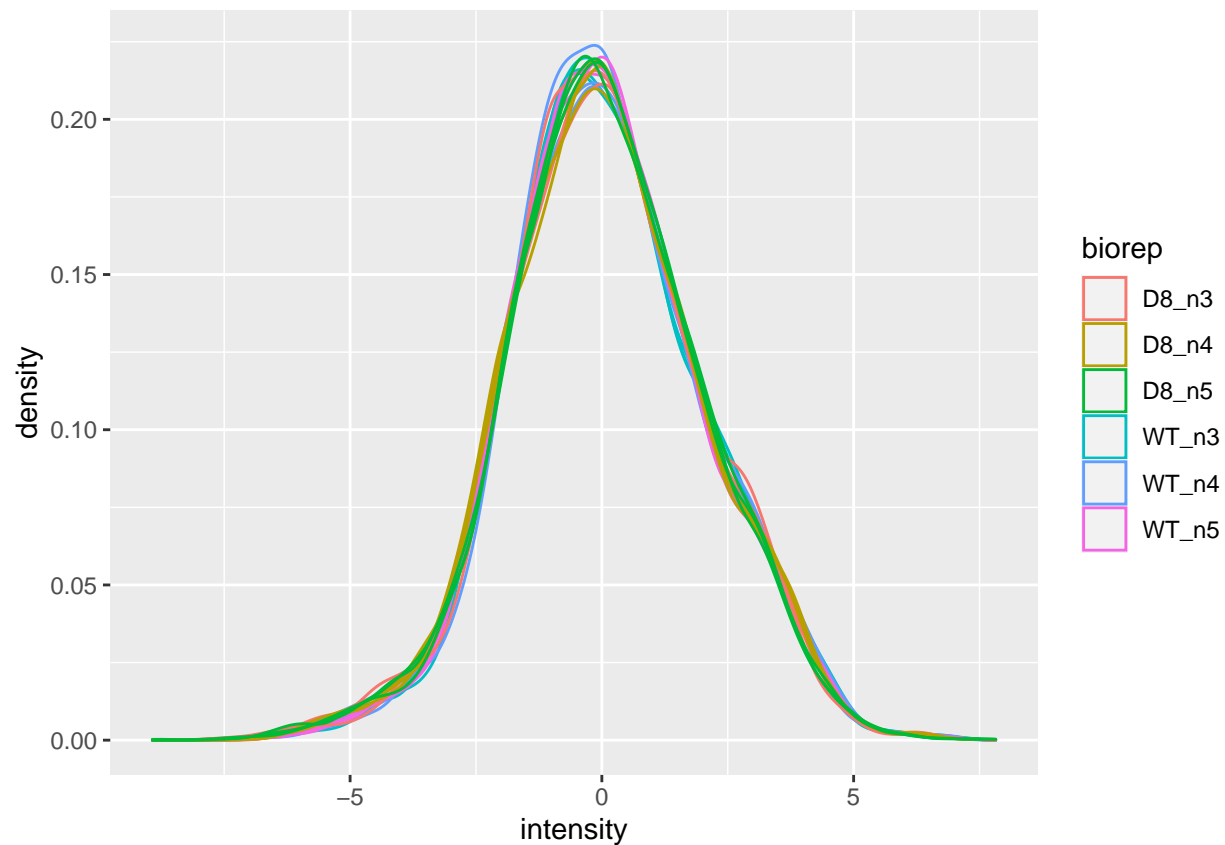
```
pe <- normalize(pe,  
  i = "peptideLog",  
  name = "peptideNorm",  
  method = "center.median")
```

3.4 Explore normalized data

Upon the normalisation the density curves are nicely registered

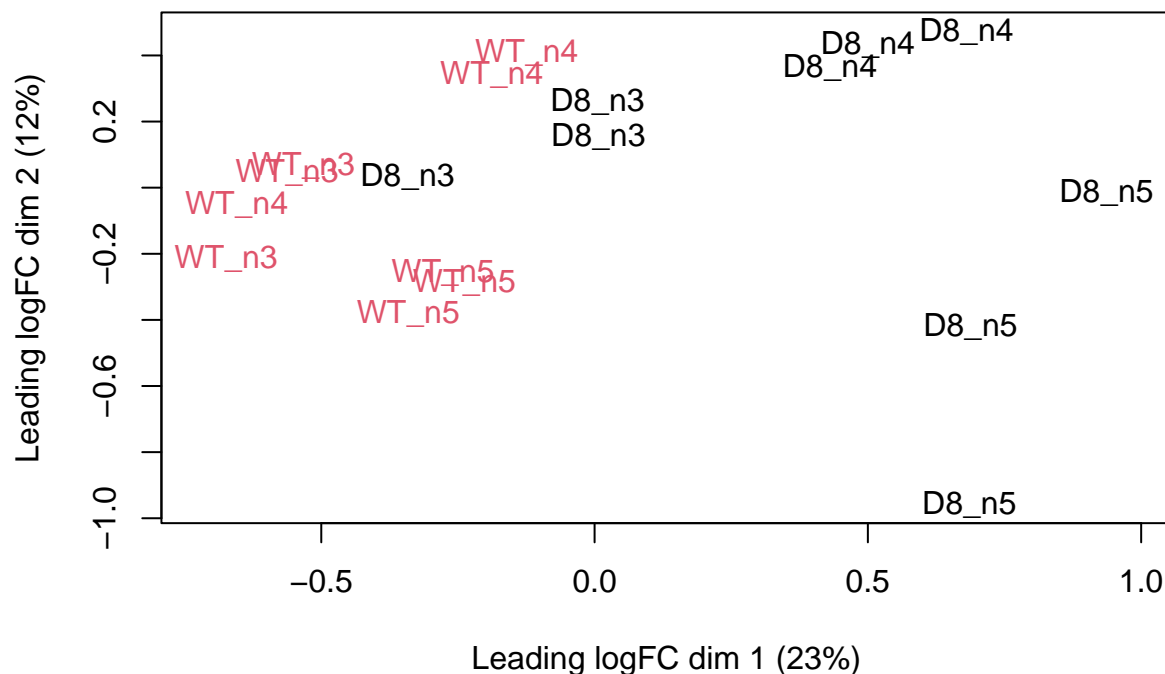
```
pe[["peptideNorm"]] %>%  
  assay %>%  
  as.data.frame() %>%  
  gather(sample, intensity) %>%  
  mutate(biorep = colData(pe)[sample, "biorep"]) %>%  
  ggplot(aes(x = intensity, group = sample, color = biorep)) +  
    geom_density()
```

```
## Warning: Removed 40413 rows containing non-finite values (stat_density).
```



We can visualize our data using a Multi Dimensional Scaling plot, eg. as provided by the `limma` package.

```
pe[["peptideNorm"]] %>%
  assay %>%
  limma::plotMDS(col = as.numeric(colData(pe)$genotype), label=colData(pe)$biorep)
```



The first axis in the plot is showing the leading log fold changes (differences on the log scale) between the samples.

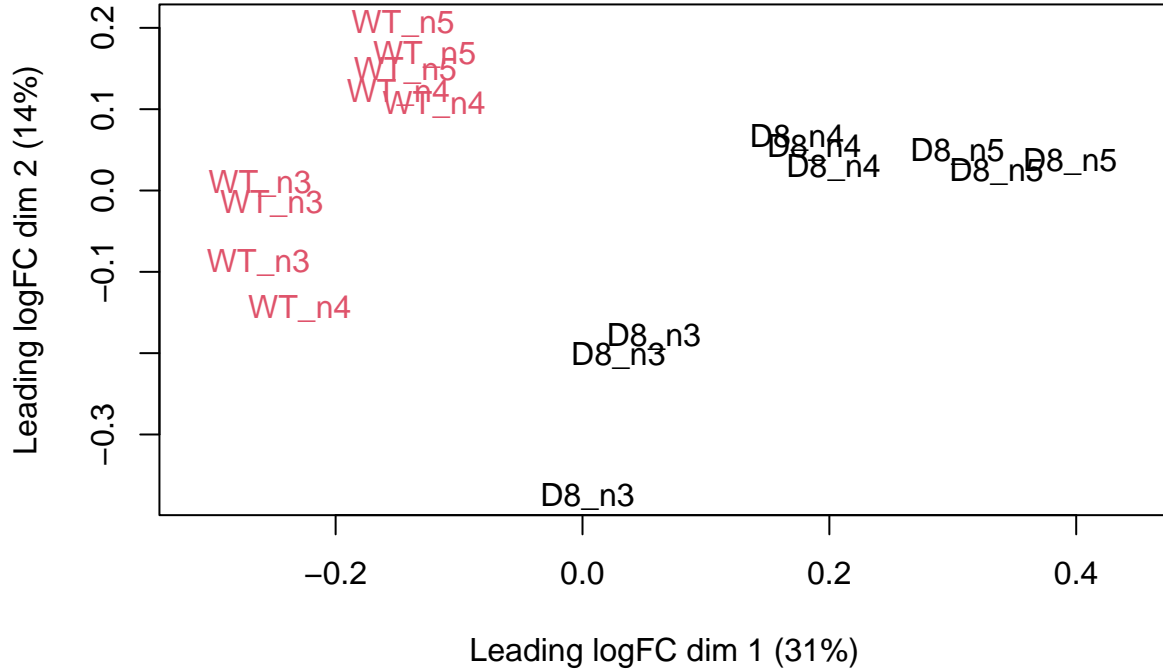
3.5 Summarization to protein level

- By default robust summarization is used: `fun = MsCoreUtils::robustSummary()`

```
pe <- aggregateFeatures(pe,
  i = "peptideNorm",
  fcol = "Proteins",
  na.rm = TRUE,
  name = "protein")
```

```
## Your quantitative and row data contain missing values. Please read the
## relevant section(s) in the aggregateFeatures manual page regarding the
## effects of missing values on data aggregation.
```

```
pe[["protein"]] %>%
  assay %>%
  limma::plotMDS(col = as.numeric(colData(pe)$genotype), label=colData(pe)$biorep)
```



Note that the samples upon robust summarisation show a clear separation according to the genotype in the first dimension of the MDS plot.

4 Data Analysis

4.1 Estimation

We model the protein level expression values using `msqrob`. By default `msqrob2` estimates the model parameters using robust regression.

We will model the data with a different group mean. The group is encoded in the variable `genotype` of the `colData`. We will also have to include a random effect for bio-repeat to address the pseudo-replication in the experiment. Indeed, the data from the same bio-repeat will be correlated!

We can specify this model by using a formula with the factor `genotype` as a fixed effect and as the factor `biorep` a random effect: `formula = ~genotype + (1|biorep)`.

In the current implementation of `msqrob2`, you can only work with mixed models if you set the ridge argument `ridge=TRUE`. The fixed effects are then estimated using ridge regression and random effects can be introduced in the model. Our implementation exploits the link between penalized ridge regression and mixed models. Indeed, by reformulating the fixed effects as random effects ridge regression can be implemented and the ridge penalty is estimated from the data.

Note, that ridge regression only works if two or more slope parameters have to be estimated for the fixed effects. Here, we only have a factor with two levels resulting in an encoding with only one slope parameter so the `msqrob` function will throw an error.

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

4.2 Inference

First, we extract the parameter names of the model by looking at the first model. The models are stored in the row data of the assay under the default name `msqrobModels`.

```
getCoef(rowData(pe[["protein"]])$msqrobModels[[1]])
```

```
##           (Intercept) (Intercept)biorepD8_n3 (Intercept)biorepD8_n4
##      3.675785e-02      -2.694535e-01      5.394845e-02
## (Intercept)biorepD8_n5 (Intercept)biorepWT_n3 (Intercept)biorepWT_n4
##      8.643411e-02      -1.003137e-01      -3.294403e-03
## (Intercept)biorepWT_n5 (Intercept)genotypeD8 (Intercept)genotypeWT
##      2.326791e-01      -8.654787e-15      8.654787e-15
```

With our encoding we get an estimate for each genotype: `(Intercept)genotypeD8` and `(Intercept)genotypeWT`.

Thus, we assess the contrast `'(Intercept)genotypeD8 - (Intercept)genotypeWT = 0'` with our statistical test. Note, that specifying this contrast will not work with the default `makeContrast` function due to the parameter names. So we therefore first

```
L <- makeContrast("genotypeD8 - genotypeWT=0", parameterNames = c("genotypeWT", "genotypeD8"))
```

Next, we change the rownames of the matrix `L`

```
rownames(L)<-paste0("(Intercept)",rownames(L))
L
```

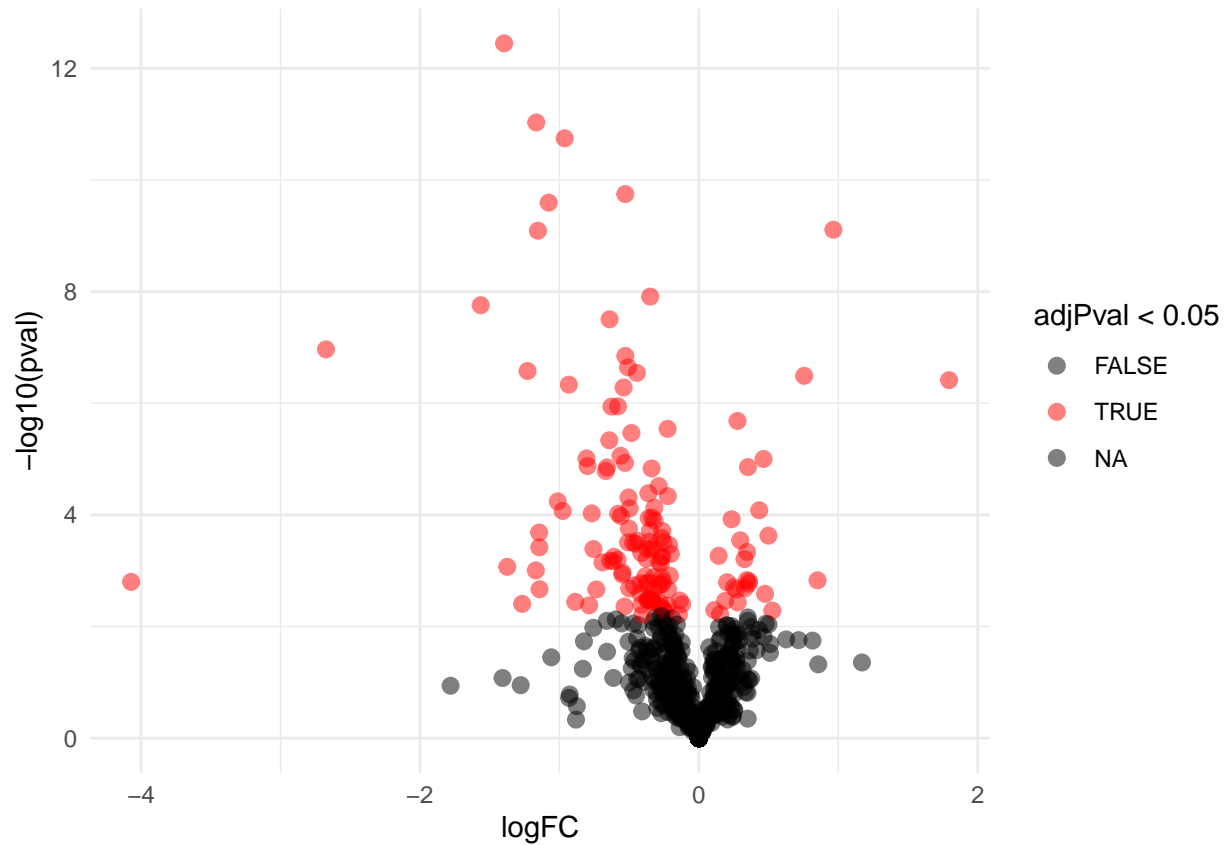
```
##           genotypeD8 - genotypeWT
## (Intercept)genotypeWT           -1
## (Intercept)genotypeD8            1
```

```
pe <- hypothesisTest(object = pe, i = "protein", contrast = L)
```

4.3 Plots

4.3.1 Volcano-plot

```
volcano <- ggplot(rowData(pe[["protein"]])$'genotypeD8 - genotypeWT',
                  aes(x = logFC, y = -log10(pval), color = adjPval < 0.05)) +
  geom_point(cex = 2.5) +
  scale_color_manual(values = alpha(c("black", "red"), 0.5)) + theme_minimal()
volcano
```

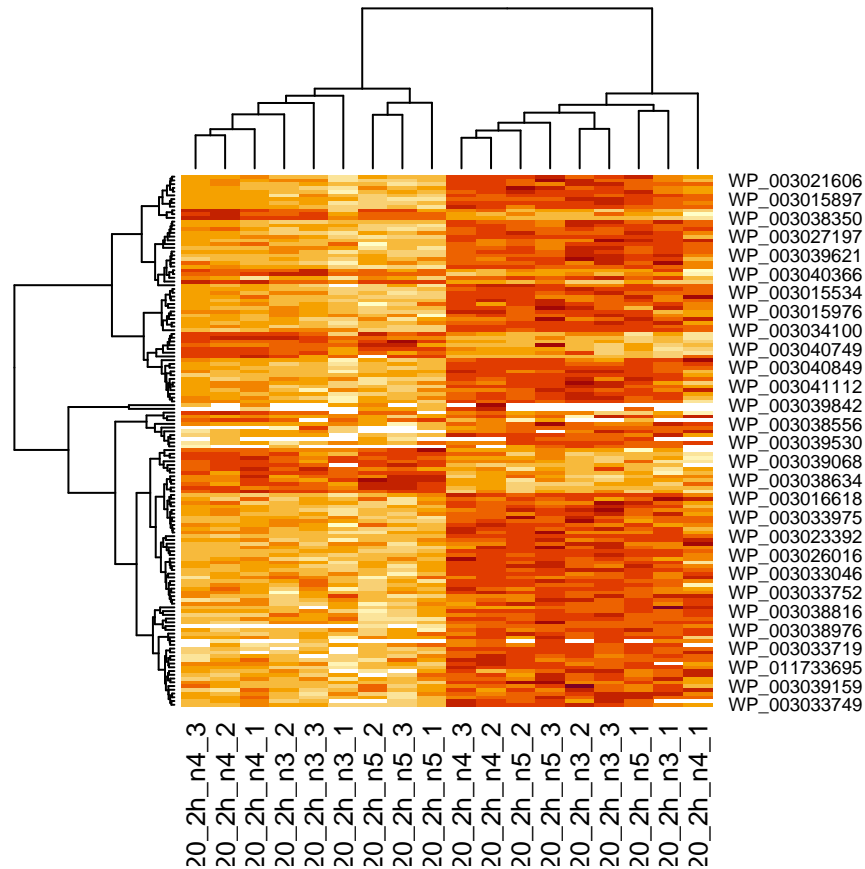


Note, that 142 proteins are found to be differentially abundant.

4.3.2 Heatmap

We first select the names of the proteins that were declared significant.

```
sigNames <- rowData(pe[["protein"]])$'genotypeD8 - genotypeWT' %>%
  rownames_to_column("protein") %>%
  filter(adjPval<0.05) %>%
  pull(protein)
heatmap(assay(pe[["protein"]])[sigNames, ])
```

4.3.3 Detail plots

We will make detail plots for the first 10 DE proteins.

```
maxPlot <- 10
for (protName in sigNames[1:maxPlot])
{
  pePlot <- pe[protName, , c("peptideNorm", "protein")]
  pePlotDf <- data.frame(longFormat(pePlot))
  pePlotDf$assay <- factor(pePlotDf$assay,
                          levels = c("peptideNorm", "protein"))
  pePlotDf$condition <- as.factor(colData(pePlot)[pePlotDf$colname, "genotype"])

  # plotting
  p1 <- ggplot(data = pePlotDf,
               aes(x = colname, y = value, group = rowname)) +
    geom_line() +
    geom_point() +
    theme(axis.text.x = element_text(angle = 70, hjust = 1, vjust = 0.5)) +
    facet_grid(~assay) +
    ggtitle(protName)
  print(p1)

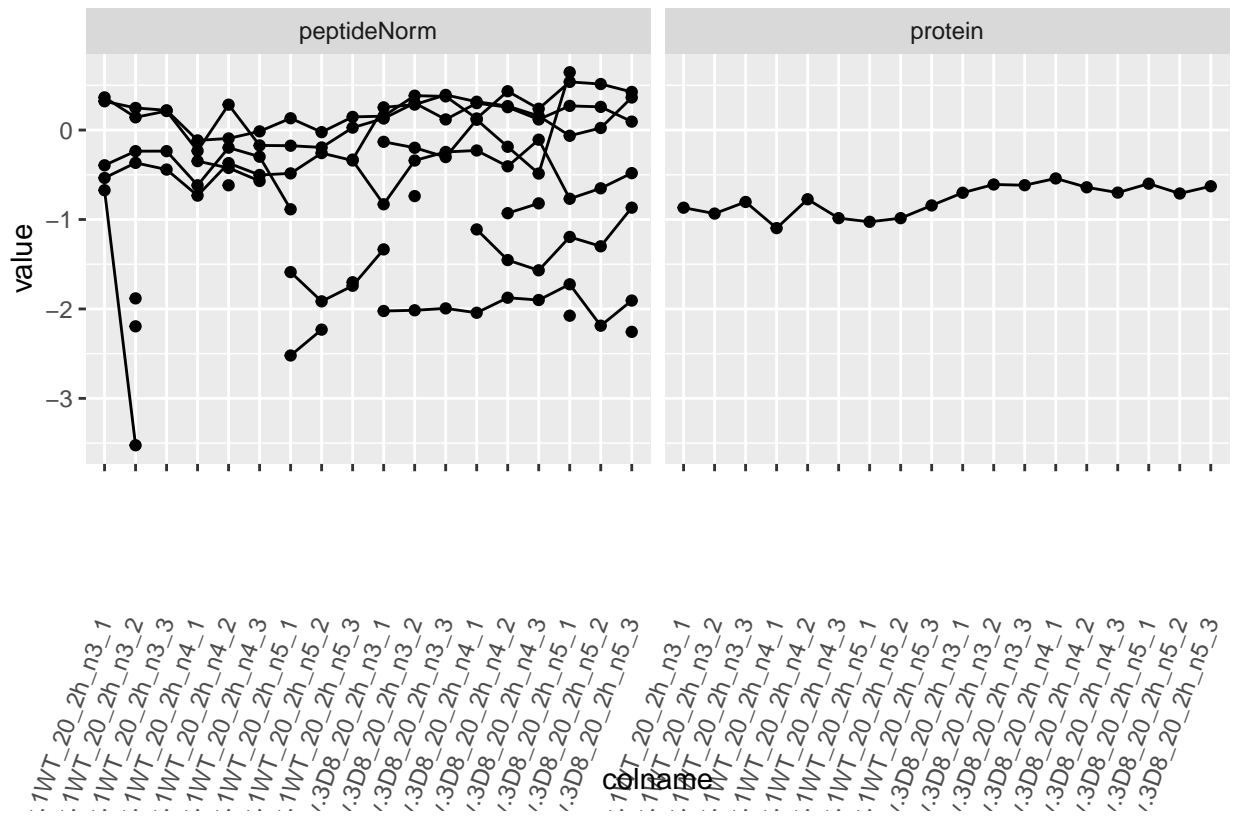
  # plotting 2
  p2 <- ggplot(pePlotDf, aes(x = colname, y = value, fill = condition)) +
```

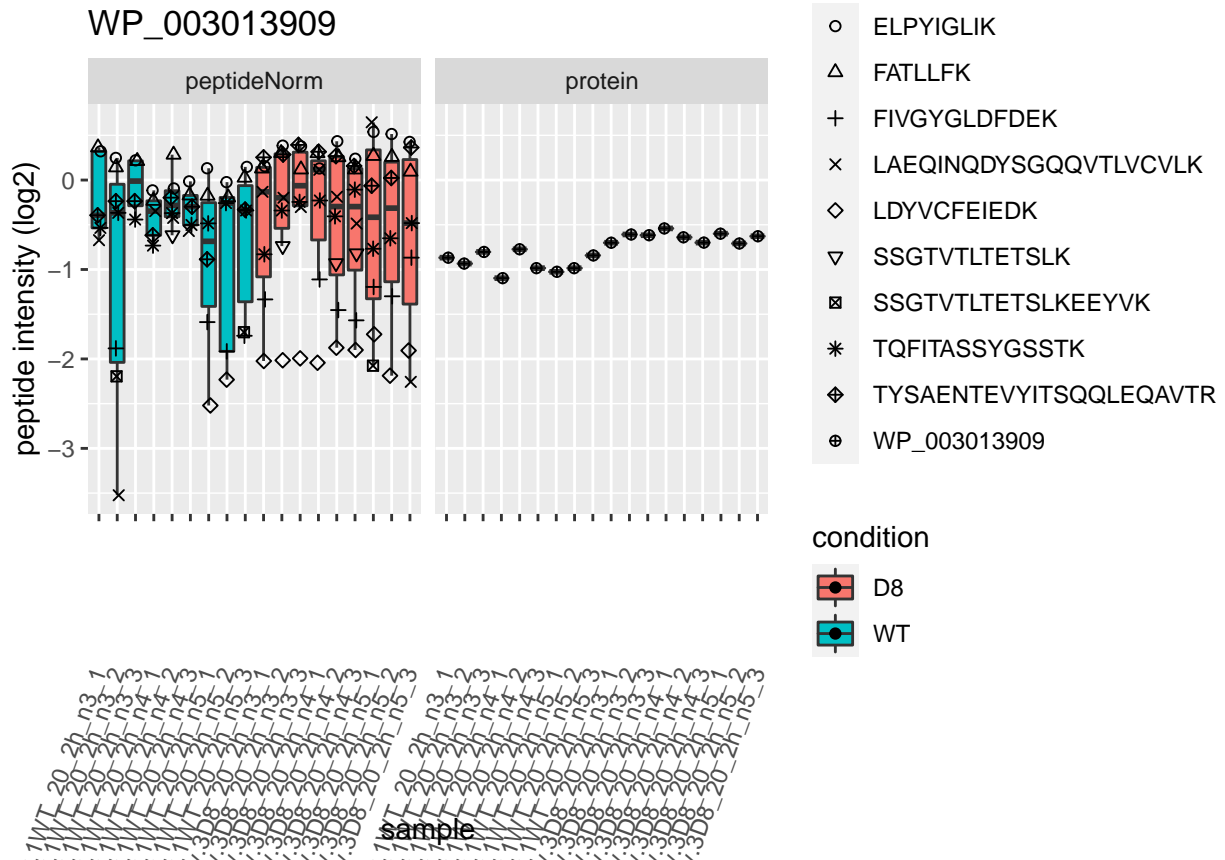
```

geom_boxplot(outlier.shape = NA) +
geom_point(
  position = position_jitter(width = .1),
  aes(shape = rowname)) +
scale_shape_manual(values = 1:nrow(pePlotDf)) +
labs(title = protName, x = "sample", y = "peptide intensity (log2)") +
theme(axis.text.x = element_text(angle = 70, hjust = 1, vjust = 0.5)) +
facet_grid(~assay)
print(p2)
}

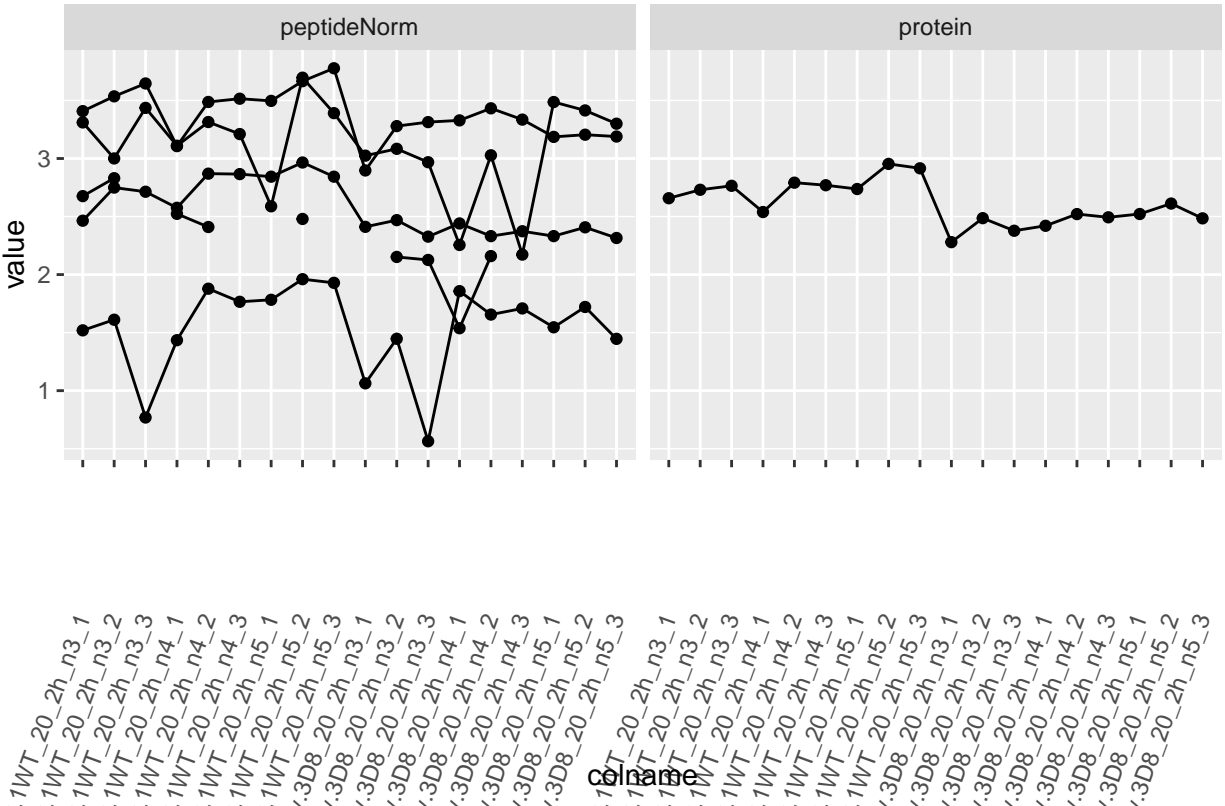
```

WP_003013909

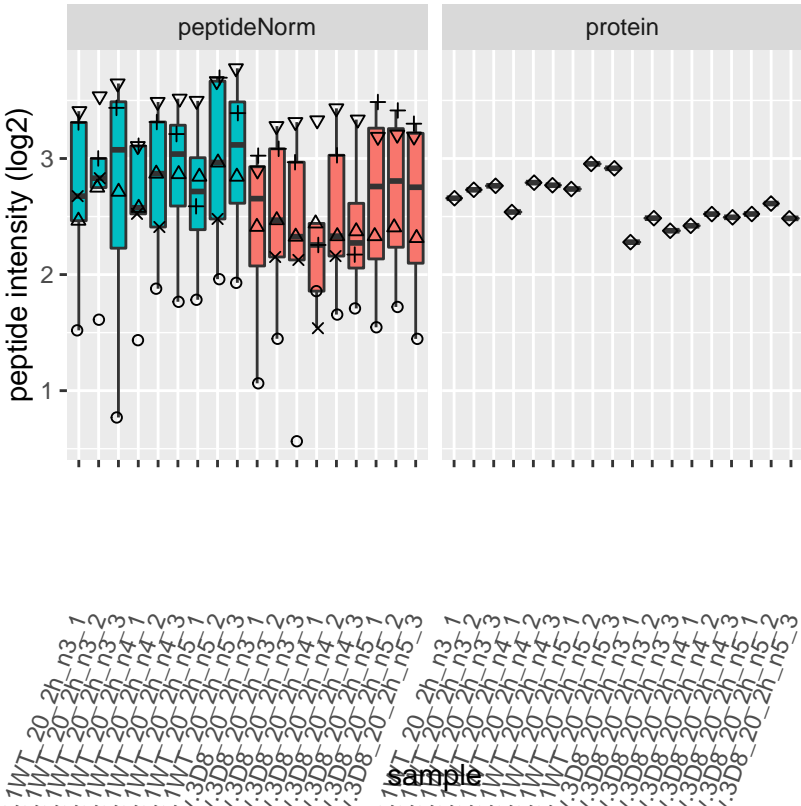




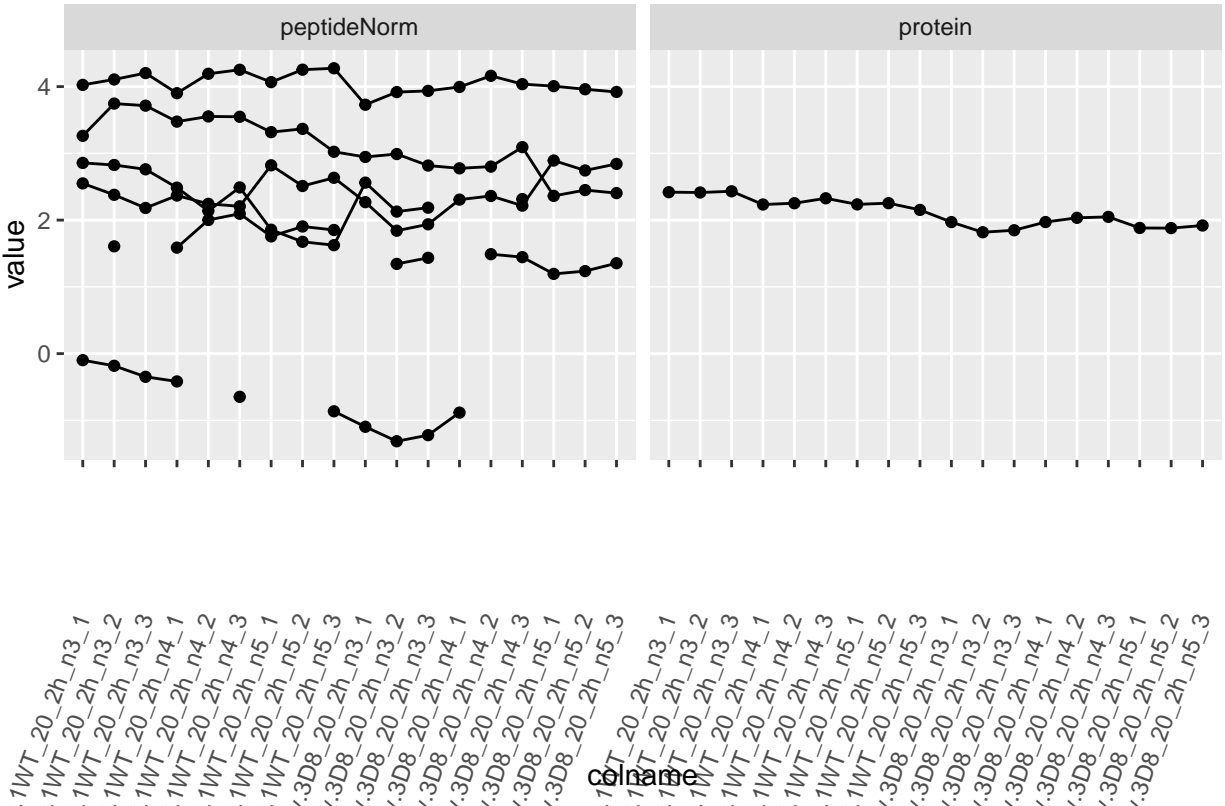
WP_003014346



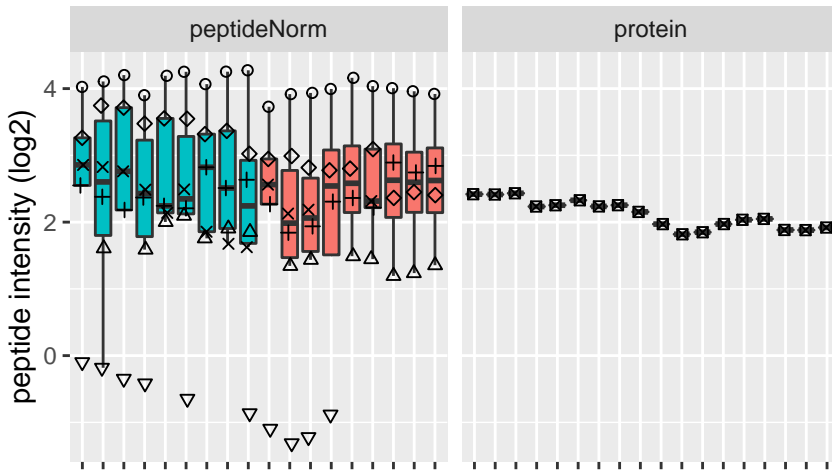
WP_003014346



WP_003014349



WP_003014349



rowname

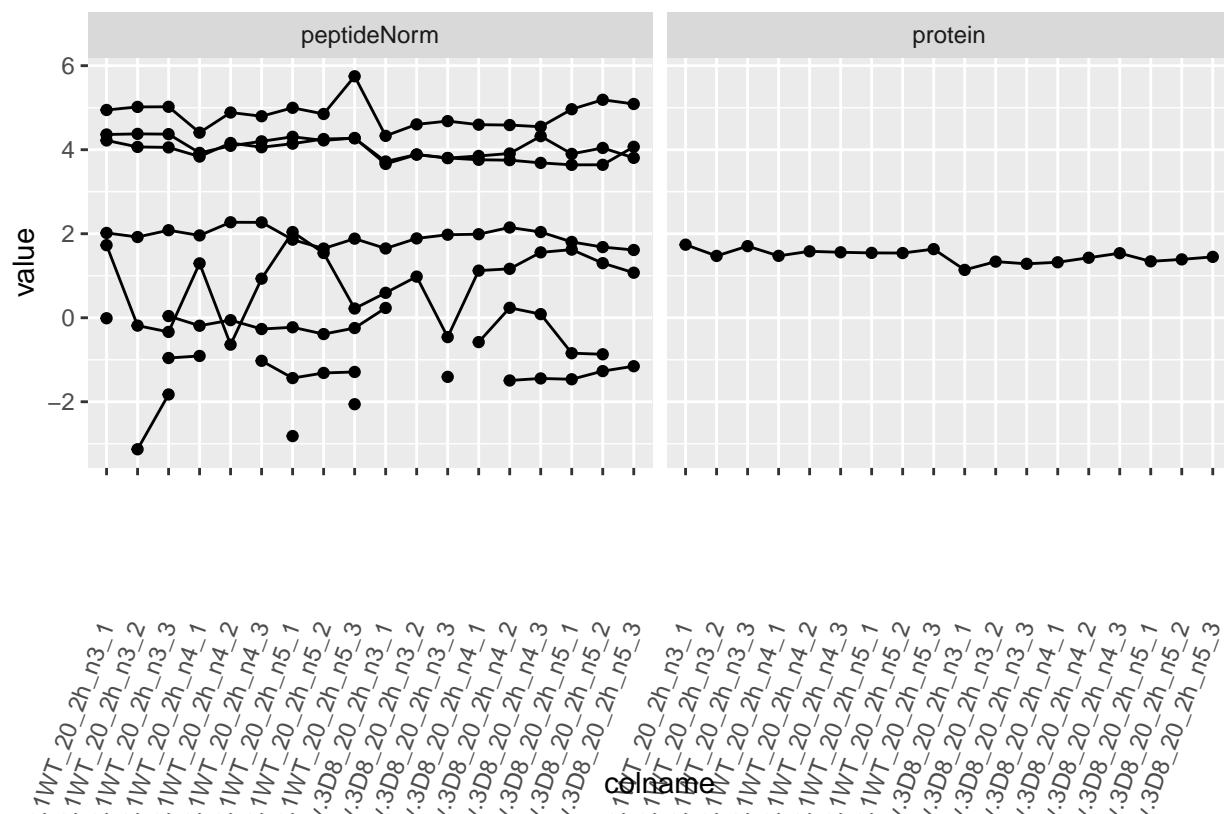
- ELPVDASNVAIFNPATEK
- △ GDDVIVIAGK
- + GGSLVLVEGINIVK
- × HIKPNPNR
- ◇ KGDDVIVIAGK
- ▽ KHIKPNPNR
- ☒ WP_003014349

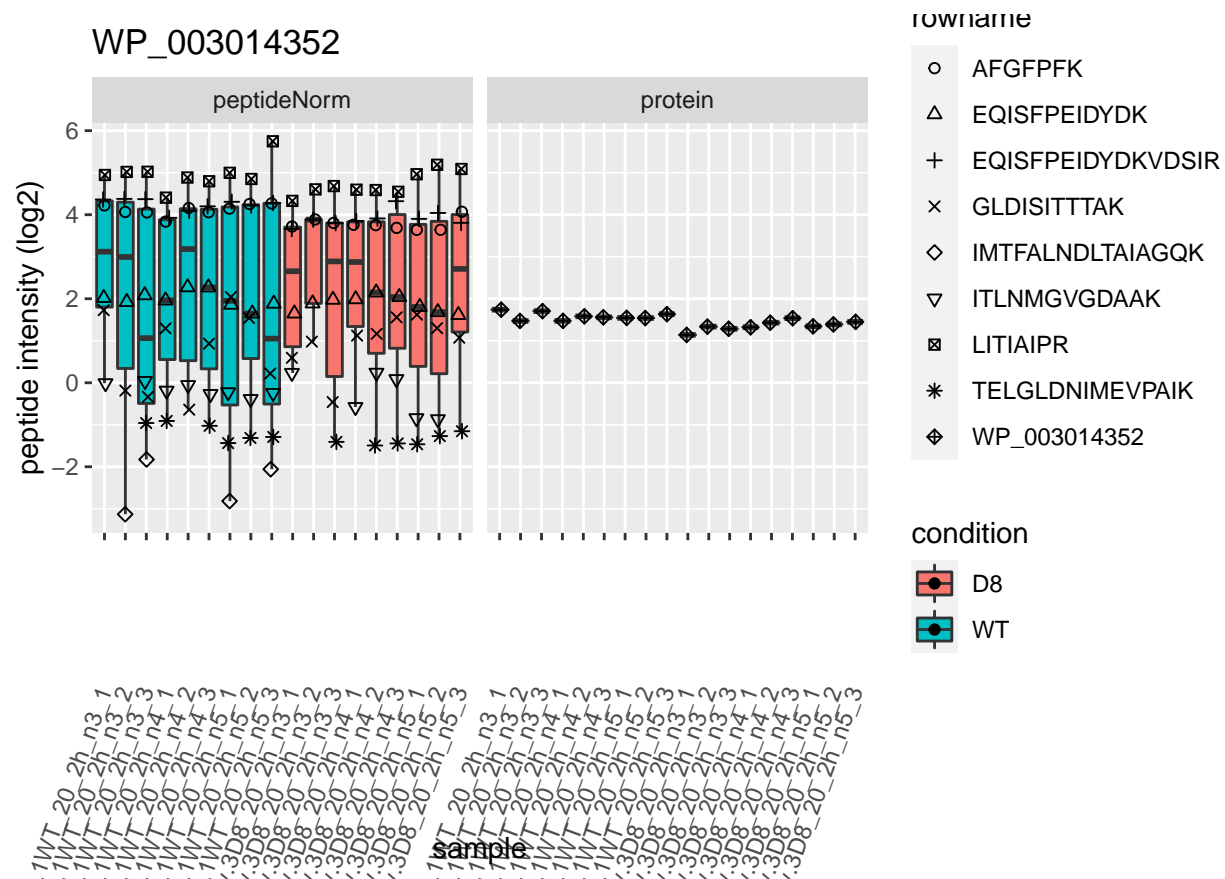
condition

- 
- Box plots showing the distribution of a variable for two genotypes: D8 (red box) and WT (blue box). The y-axis represents a numerical value, and the x-axis labels the genotypes. The D8 group shows a higher median and greater variability than the WT group.

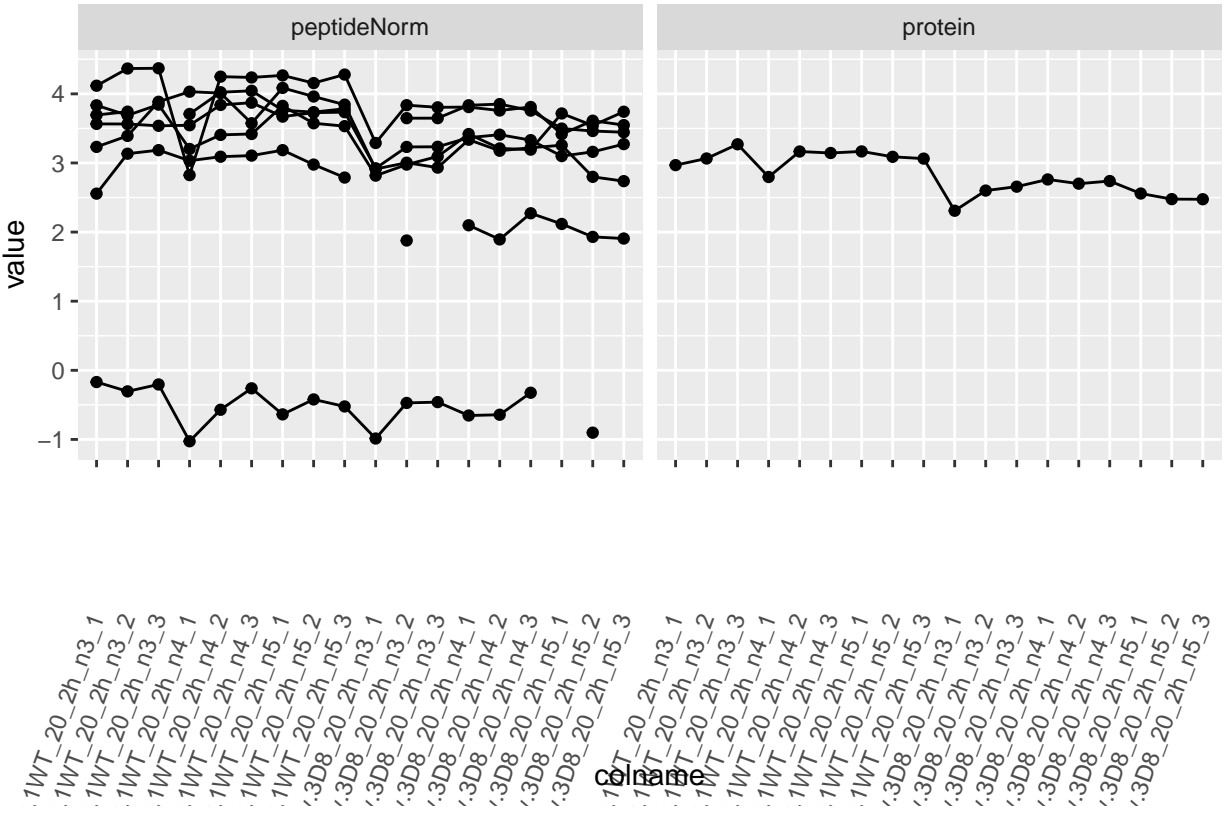
sample

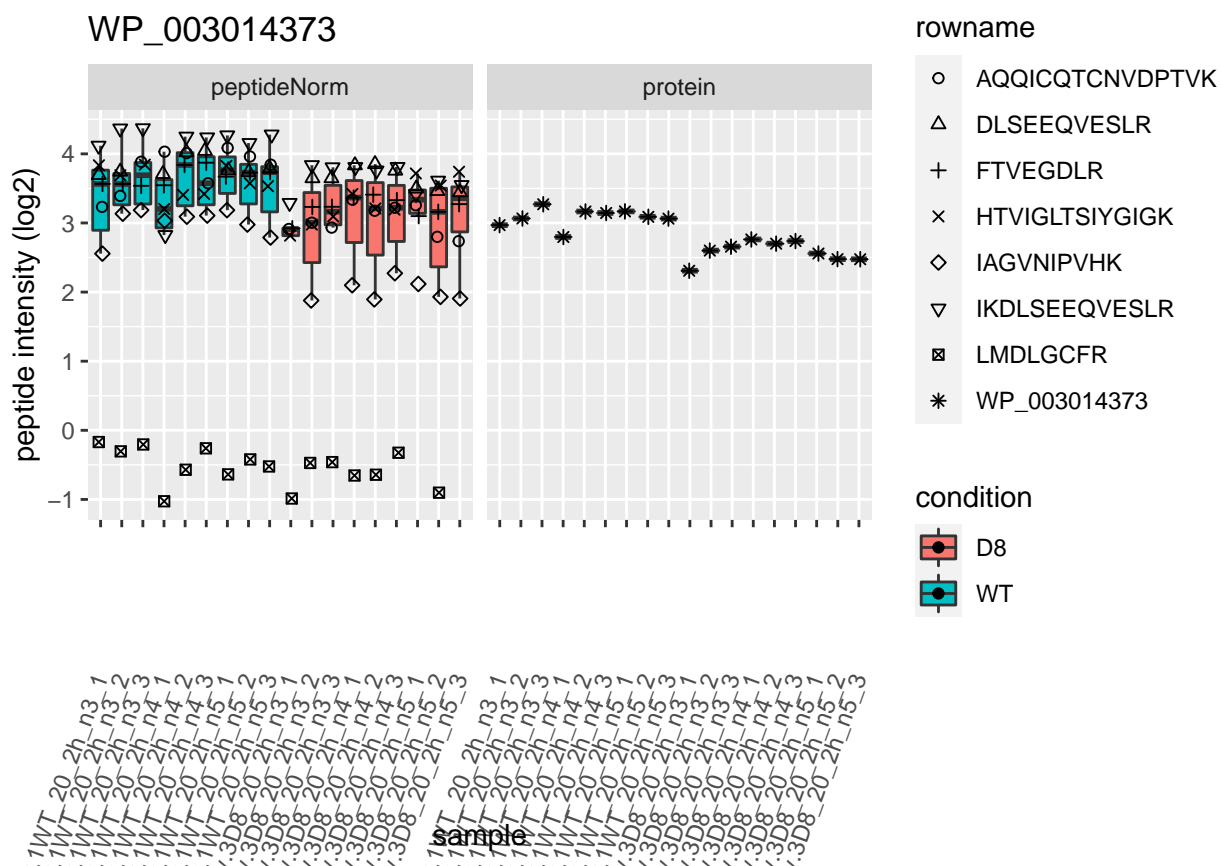
WP_003014352



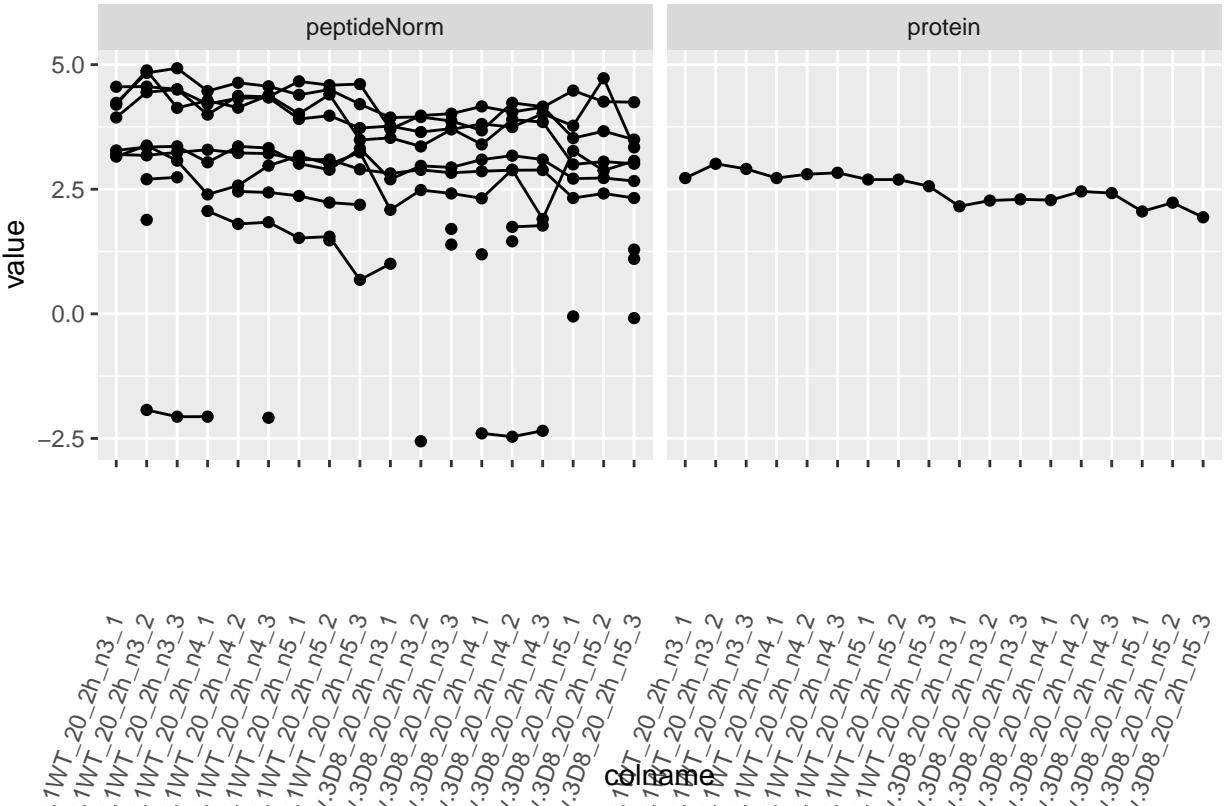


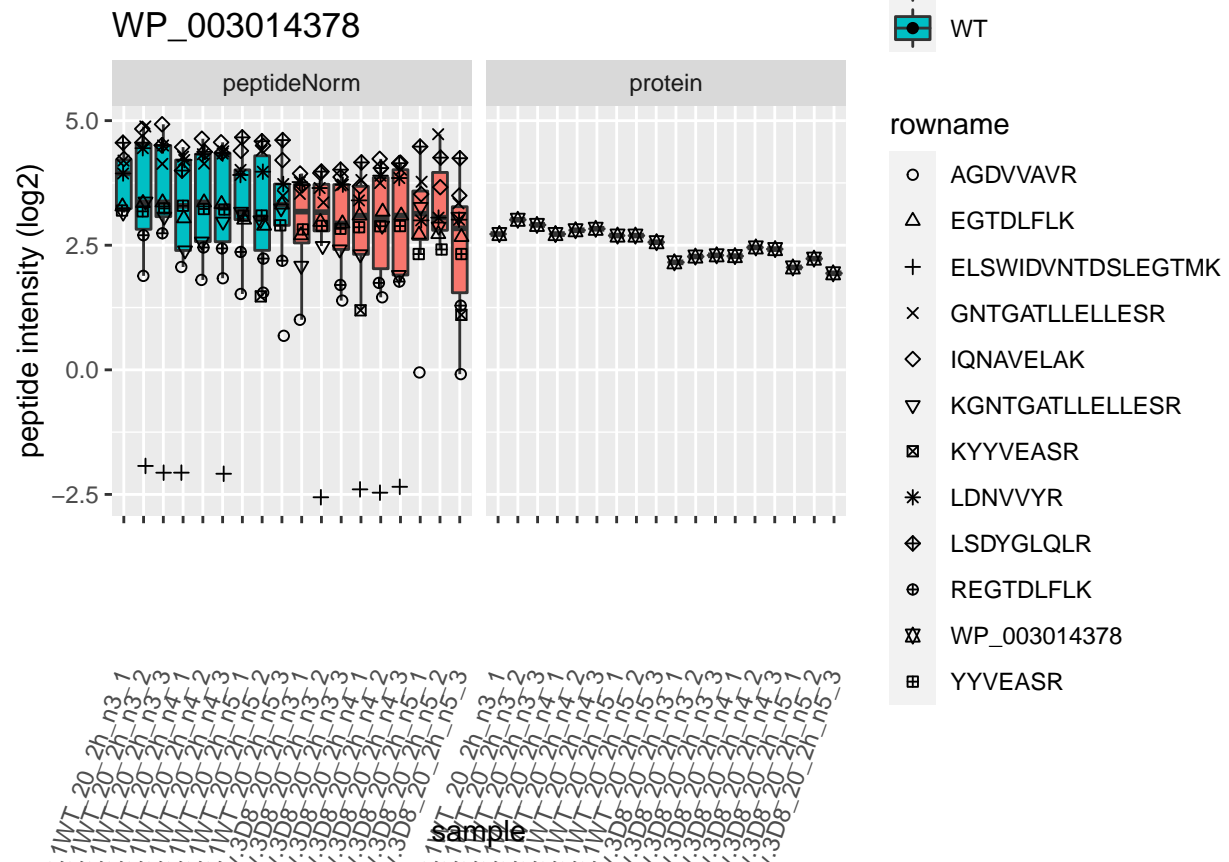
WP_003014373



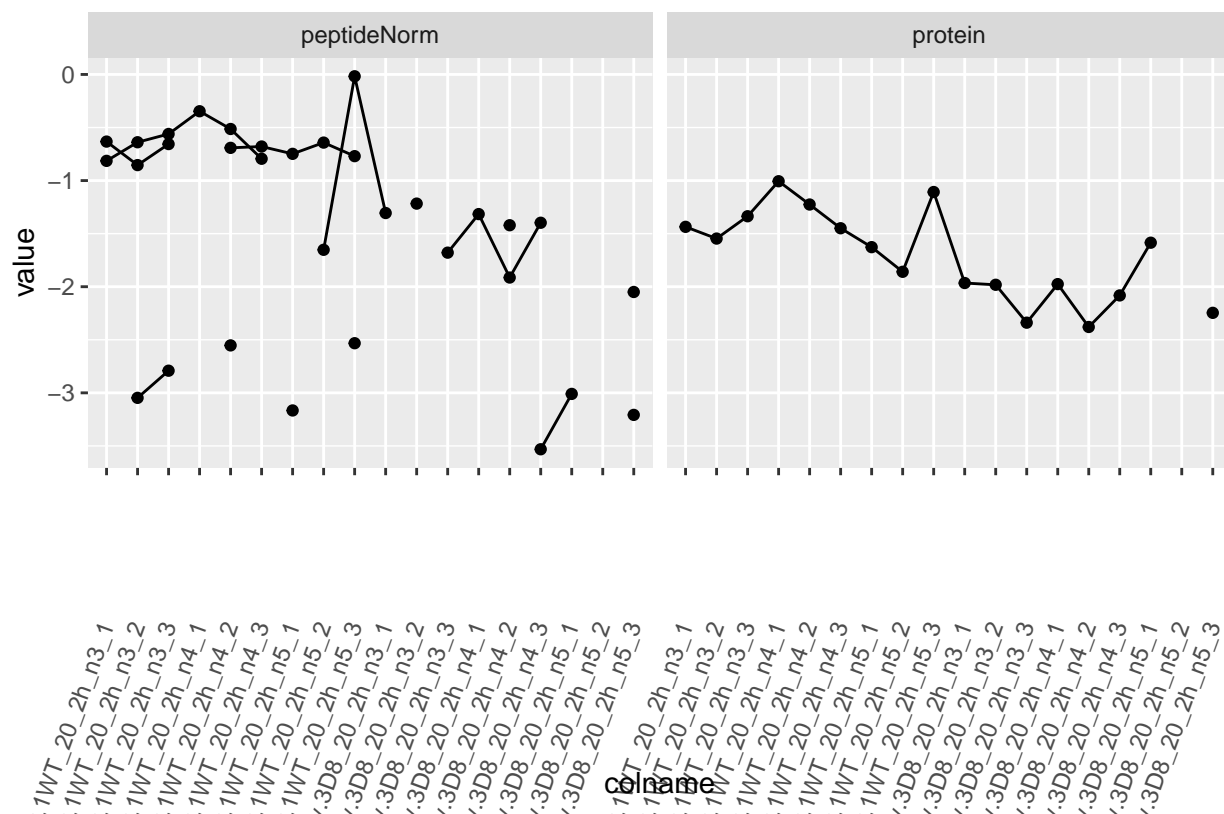


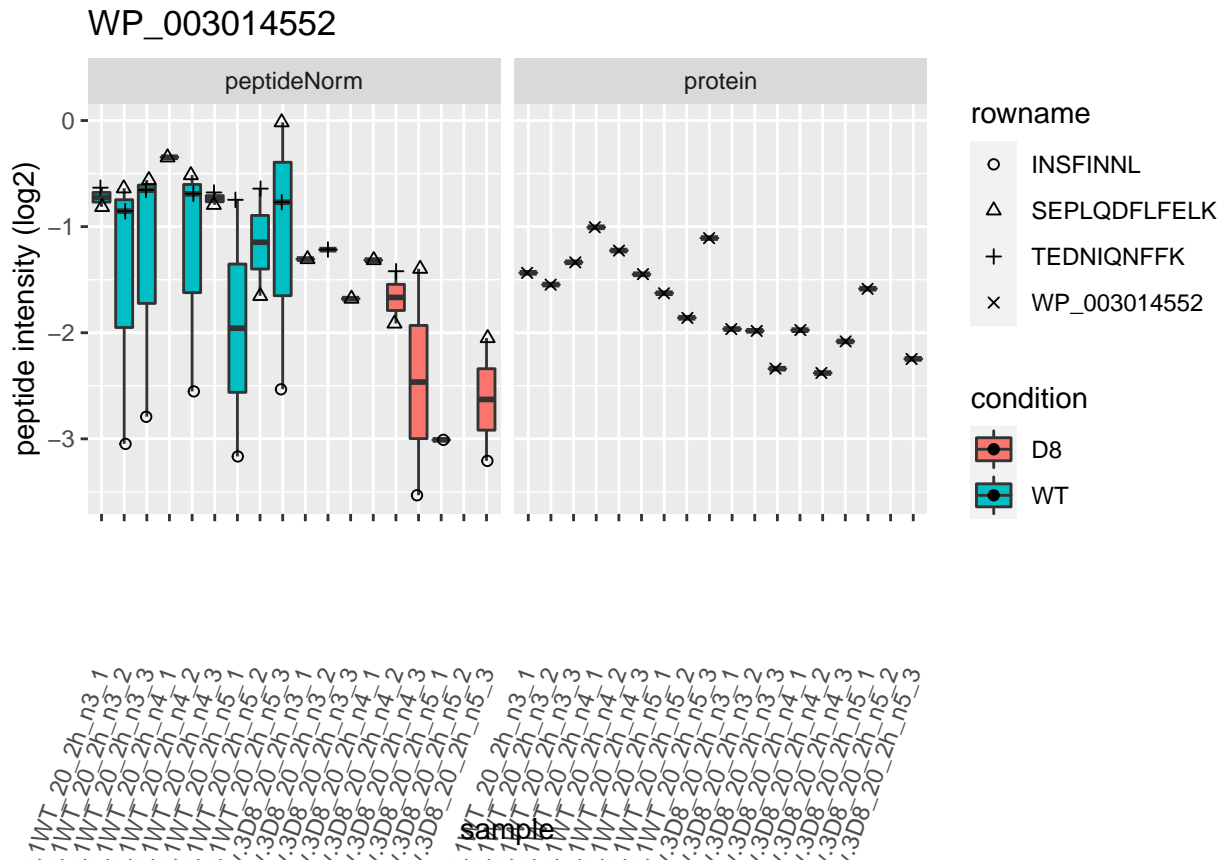
WP_003014378



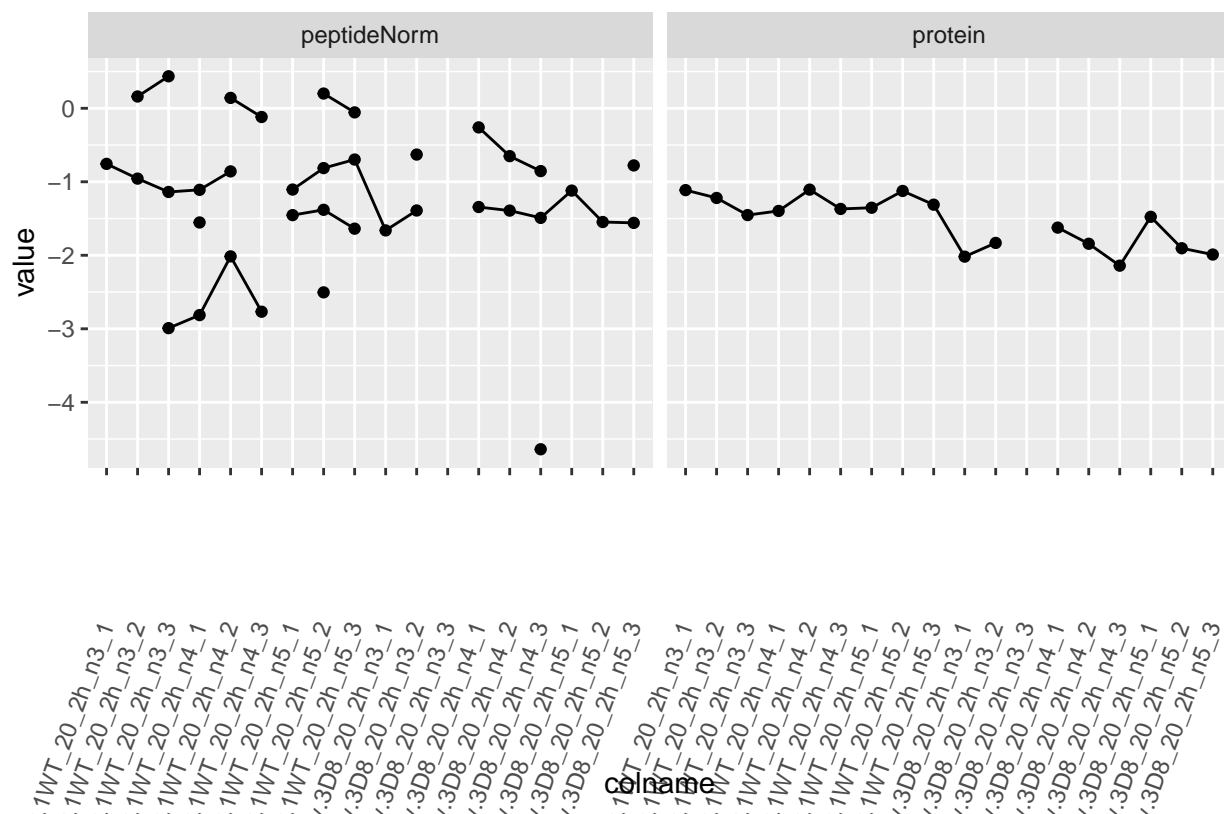


WP_003014552

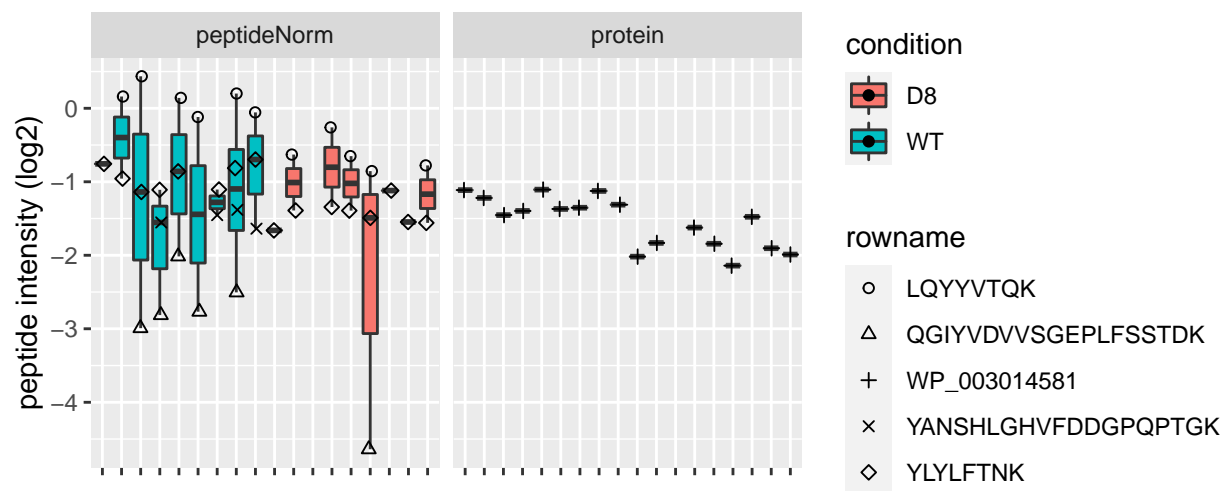




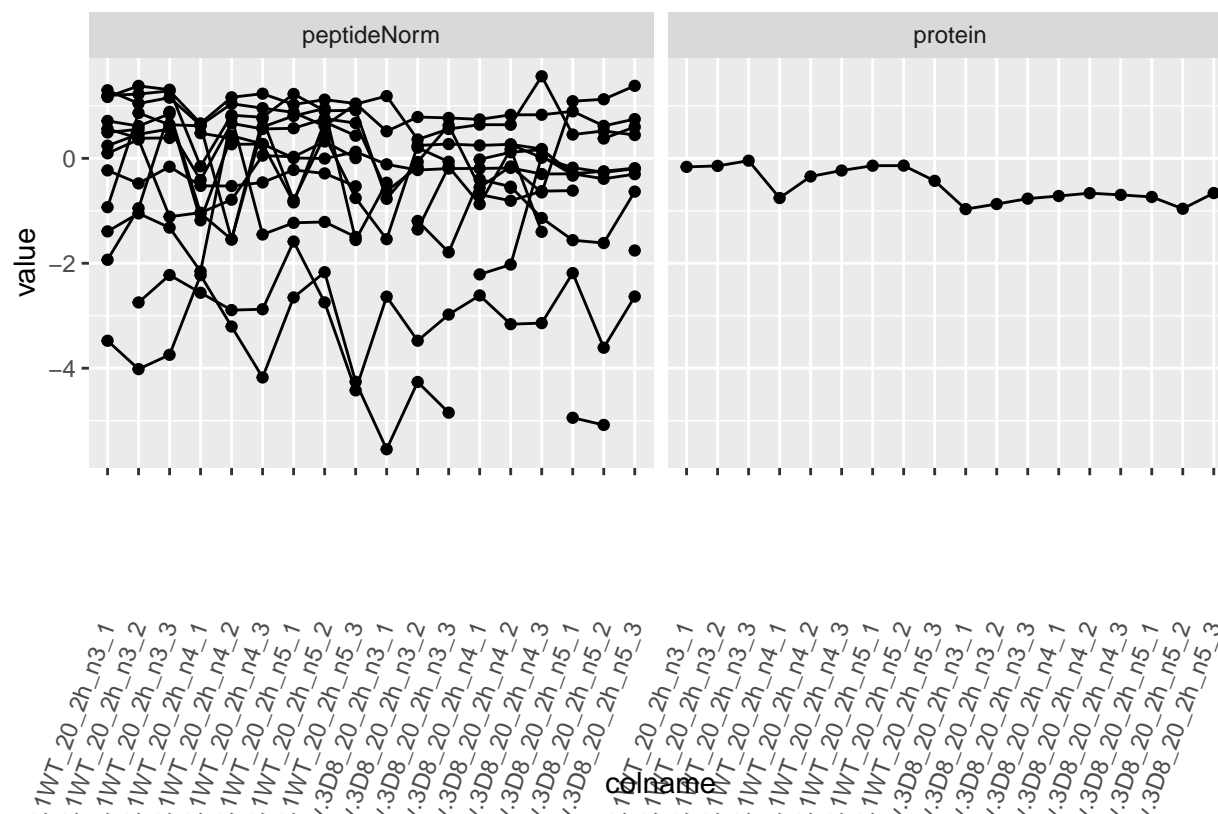
WP_003014581

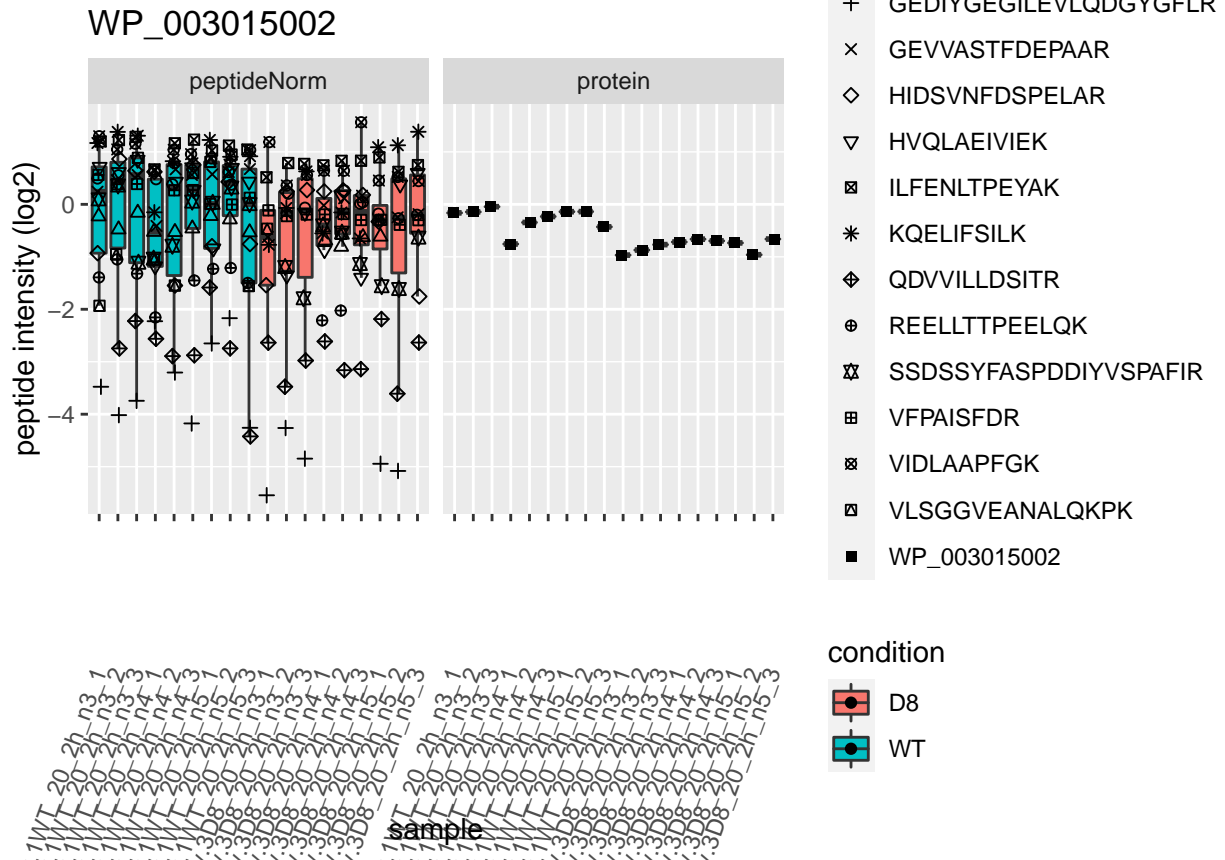


WP_003014581

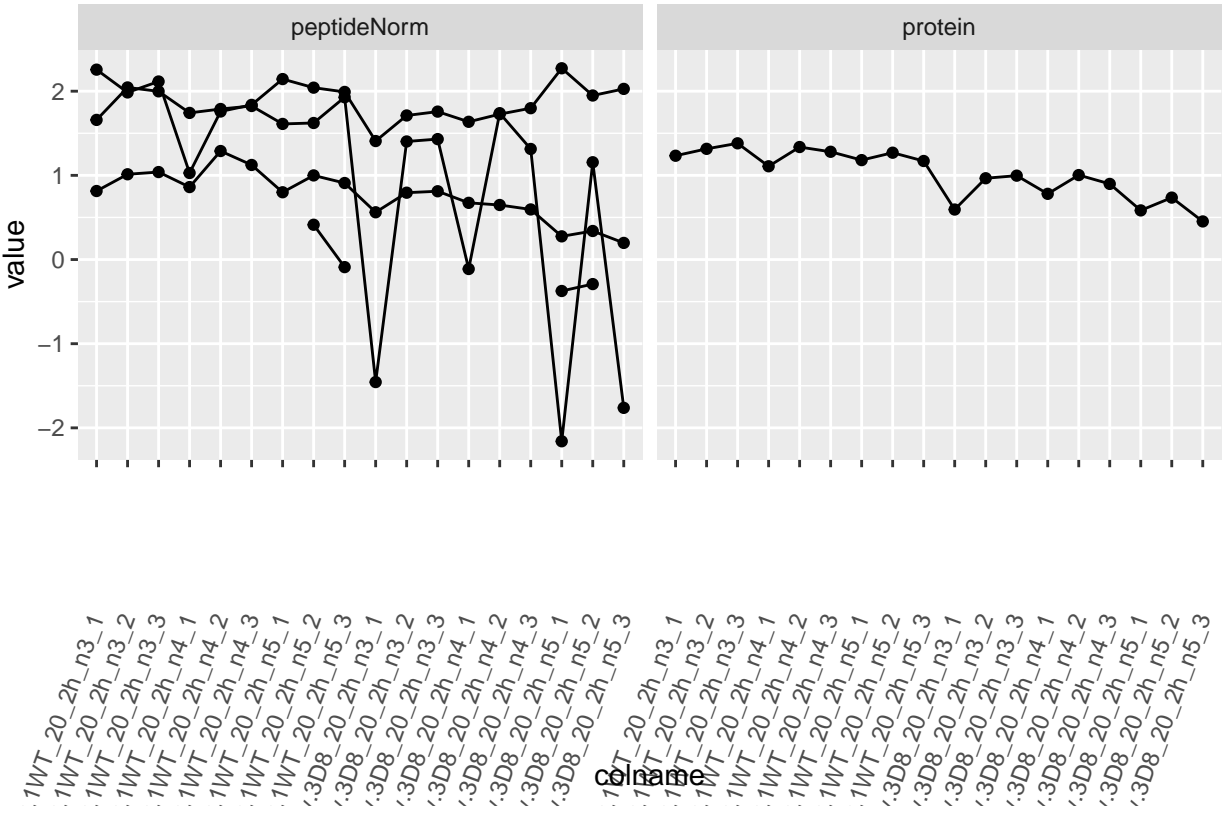


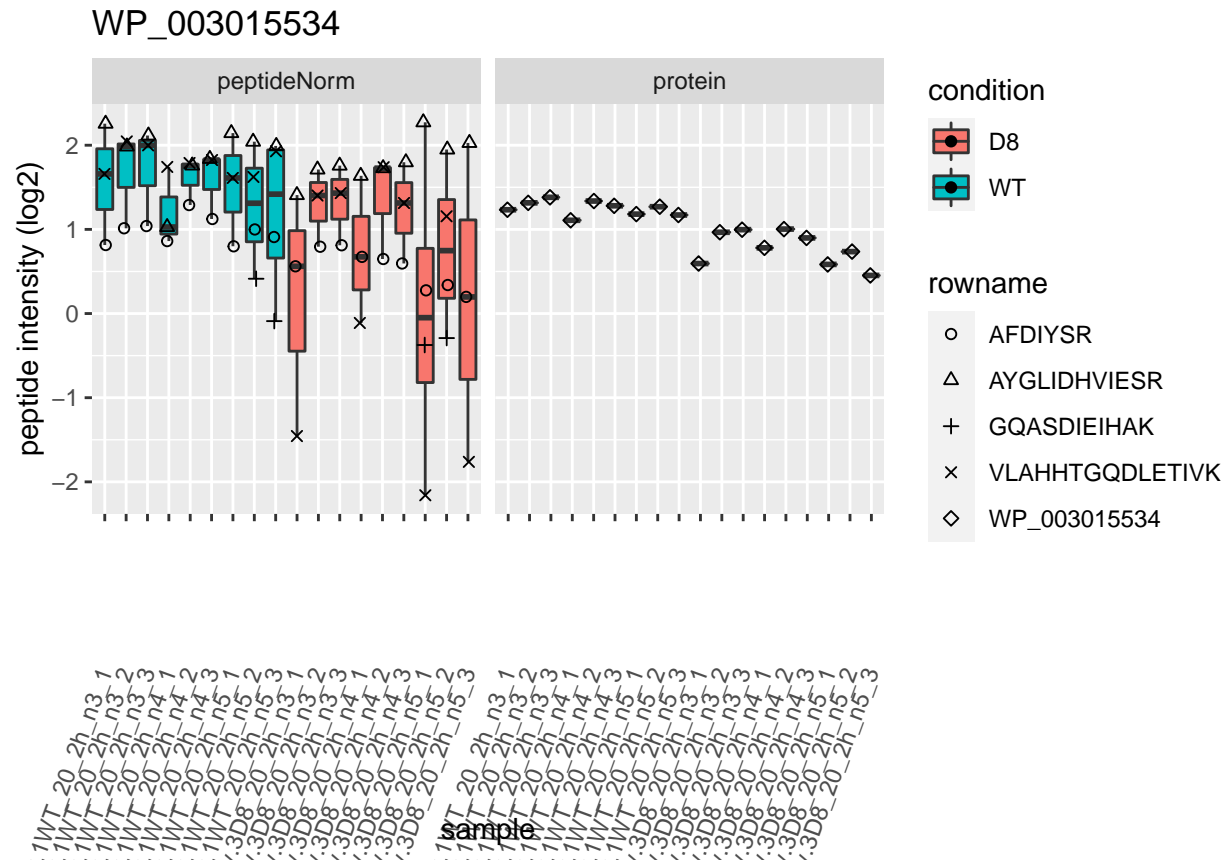
WP_003015002





WP_003015534





Note, that the yeast protein is only covered by 3 peptides. Only one peptide is picked up in condition A. This peptide is also only once observed in spike-in condition B. This puts a considerable burden upon the inference and could be avoided by more stringent filtering.

5 Session Info

With respect to reproducibility, it is highly recommended to include a session info in your script so that readers of your output can see your particular setup of R.

```
sessionInfo()

## R version 4.1.0 (2021-05-18)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Catalina 10.15.7
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] parallel stats4 stats graphics grDevices utils datasets
```

```

## [8] methods      base
##
## other attached packages:
## [1] BiocStyle_2.20.2      gridExtra_2.3
## [3] plotly_4.9.4.1        msqrob2_0.99.6
## [5] QFeatures_1.2.0       MultiAssayExperiment_1.18.0
## [7] SummarizedExperiment_1.22.0 Biobase_2.52.0
## [9] GenomicRanges_1.44.0  GenomeInfoDb_1.28.1
## [11] IRanges_2.26.0        S4Vectors_0.30.0
## [13] BiocGenerics_0.38.0   MatrixGenerics_1.4.0
## [15] matrixStats_0.59.0    limma_3.48.1
## [17] forcats_0.5.1         stringr_1.4.0
## [19] dplyr_1.0.7           purrr_0.3.4
## [21] readr_1.4.0           tidyr_1.1.3
## [23] tibble_3.1.2          ggplot2_3.3.5
## [25] tidyverse_1.3.1
##
## loaded via a namespace (and not attached):
## [1] minqa_1.2.4           colorspace_2.0-2      ellipsis_0.3.2
## [4] XVector_0.32.0        fs_1.5.0              clue_0.3-59
## [7] rstudioapi_0.13       farver_2.1.0          mzR_2.26.1
## [10] affyio_1.62.0         fansi_0.5.0           lubridate_1.7.10
## [13] xml2_1.3.2            codetools_0.2-18      splines_4.1.0
## [16] doParallel_1.0.16     ncdf4_1.17            impute_1.66.0
## [19] knitr_1.33            jsonlite_1.7.2        nloptr_1.2.2.2
## [22] broom_0.7.8           vsn_3.60.0            cluster_2.1.2
## [25] dbplyr_2.1.1          BiocManager_1.30.16   compiler_4.1.0
## [28] httr_1.4.2            backports_1.2.1       assertthat_0.2.1
## [31] Matrix_1.3-3          lazyeval_0.2.2        cli_3.0.0
## [34] htmltools_0.5.1.1     tools_4.1.0           affy_1.70.0
## [37] gtable_0.3.0          glue_1.4.2            GenomeInfoDbData_1.2.6
## [40] Rcpp_1.0.7            MALDIquant_1.19.3     cellranger_1.1.0
## [43] vctrs_0.3.8           preprocessCore_1.54.0 nlme_3.1-152
## [46] iterators_1.0.13      xfun_0.24             lme4_1.1-27.1
## [49] rvest_1.0.0           lifecycle_1.0.0       XML_3.99-0.6
## [52] zlibbioc_1.38.0       MASS_7.3-54           scales_1.1.1
## [55] MSnbase_2.18.0        pcaMethods_1.84.0     hms_1.1.0
## [58] ProtGenerics_1.24.0   AnnotationFilter_1.16.0 yaml_2.2.1
## [61] stringi_1.7.3         highr_0.9             foreach_1.5.1
## [64] boot_1.3-28           BiocParallel_1.26.1   rlang_0.4.11
## [67] pkgconfig_2.0.3       bitops_1.0-7          mzID_1.30.0
## [70] evaluate_0.14         lattice_0.20-44       labeling_0.4.2
## [73] htmlwidgets_1.5.3     tidyselect_1.1.1      bookdown_0.22
## [76] plyr_1.8.6            magrittr_2.0.1        R6_2.5.0
## [79] magick_2.7.2          generics_0.1.0        DelayedArray_0.18.0
## [82] DBI_1.1.1            pillar_1.6.1          haven_2.4.1
## [85] withr_2.4.2           MsCoreUtils_1.4.0     RCurl_1.98-1.3
## [88] msdata_0.32.0         modelr_0.1.8          crayon_1.4.1
## [91] utf8_1.2.1            rmarkdown_2.9         grid_4.1.0
## [94] readxl_1.3.1          data.table_1.14.0     reprex_2.0.0
## [97] digest_0.6.27         munsell_0.5.0         viridisLite_0.4.0

```

References

- Goeminne, L. J., K. Gevaert, and L. Clement. 2016. “Peptide-level Robust Ridge Regression Improves Estimation, Sensitivity, and Specificity in Data-dependent Quantitative Label-free Shotgun Proteomics.” *Mol Cell Proteomics* 15 (2): 657–68.
- Sticker, A., L. Goeminne, L. Martens, and L. Clement. 2020. “Robust Summarization and Inference in Proteome-wide Label-free Quantification.” *Mol Cell Proteomics* 19 (7): 1209–19.