

# Statistical Methods for Quantitative MS-based Proteomics: Peptide-level Models for Summarization and Inference

Lieven Clement

[statOmics](#), Ghent University

## Contents

<b>1</b>	<b>Subset of CPTAC study: A vs B comparison in lab 3</b>	<b>1</b>
1.1	LFQ . . . . .	1
1.2	Median & robust summarization . . . . .	18
1.3	Comparison summarization methods . . . . .	74
<b>2</b>	<b>Full CPTAC study</b>	<b>76</b>
2.1	Read data . . . . .	76
2.2	Design . . . . .	77
2.3	Preprocessing . . . . .	78
2.4	Normalization . . . . .	79
<b>3</b>	<b>Peptide-level models</b>	<b>79</b>
3.1	Summarization . . . . .	79
3.2	Estimation of differential abundance using peptide level model . . . . .	91
	<b>References</b>	<b>91</b>

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

```
library(tidyverse)
library(limma)
library(QFeatures)
library(msqrob2)
library(plotly)
library(gridExtra)
library(data.table)
```

## 1 Subset of CPTAC study: A vs B comparison in lab 3

### 1.1 LFQ

Click to see background and code

1. Import data

```
proteinsTable <- fread("https://raw.githubusercontent.com/statOmics/PDA/data/quantification/cptacAvsB_1")
int64 <- which(sapply(proteinsTable,class) == "integer64")
for (j in int64) proteinsTable[[j]] <- as.numeric(proteinsTable[[j]])

quantCols <- grep("LFQ intensity ", names(proteinsTable))
```

```

peLFQ <- readQFeatures(
  assayData = proteinsTable,
  fnames = 1,
  quantCols = quantCols,
  name = "proteinRaw"
)

## Checking arguments.
## Loading data as a 'SummarizedExperiment' object.
## Formatting sample annotations (colData).
## Formatting data as a 'QFeatures' object.
## Setting assay rownames.
rm(proteinsTable)
gc()

##          used (Mb) gc trigger (Mb) limit (Mb) max used (Mb)
## Ncells  8043037 429.6   12731717 680.0          NA 12731717 680.0
## Vcells 19082366 145.6   33242060 253.7        16384 33241985 253.7
gc()

##          used (Mb) gc trigger (Mb) limit (Mb) max used (Mb)
## Ncells  8042967 429.6   12731717 680.0          NA 12731717 680.0
## Vcells 19075712 145.6   33242060 253.7        16384 33241985 253.7

cond <- which(
  strsplit(colnames(peLFQ)[[1]][1], split = "")[[1]] == "A") # find where condition is stored

colData(peLFQ)$condition <- substr(colnames(peLFQ), cond, cond) |>
  unlist() |>
  as.factor()

```

## 2. Preprocessing

```

peLFQ <- zeroIsNA(peLFQ, "proteinRaw") # convert 0 to NA

peLFQ <- logTransform(peLFQ, base = 2, i = "proteinRaw", name = "proteinLog")

peLFQ <- filterFeatures(peLFQ, ~ Reverse != "+")

## 'Reverse' found in 2 out of 2 assay(s).
peLFQ <- filterFeatures(peLFQ, ~ Potential.contaminant != "+")

## 'Potential.contaminant' found in 2 out of 2 assay(s).
peLFQ <- normalize(peLFQ,
  i = "proteinLog",
  name = "protein",
  method = "center.median")

# We want to have at least two observed protein intensities for each group so we set the minimum number
nObs <- 4
n <- ncol(peLFQ[["protein"]])

```

```
pNA <- (n-nObs)/n
peLFQ <- filterNA(peLFQ, pNA = pNA, i = "protein")
```

### 3. Modeling and Inference

```
peLFQ <- msqrob(object = peLFQ, i = "protein", formula = ~condition)
```

[illegible]







[illegible]

[illegible]





[illegible]

[illegible]

[illegible]



[illegible]



[illegible]



[illegible]

```
L <- makeContrast("conditionB=0", parameterNames = c("conditionB"))
peLFQ <- hypothesisTest(object = peLFQ, i = "protein", contrast = L)

volcanoLFQ <- ggplot(rowData(peLFQ[["protein"]])$conditionB,
  aes(x = logFC, y = -log10(pval), color = adjPval < 0.05)) +
  geom_point(cex = 2.5) +
  scale_color_manual(values = alpha(c("black", "red"), 0.5)) +
  theme_minimal() +
  ggtitle(paste0("maxLFQ: TP = ", sum(rowData(peLFQ[["protein"]])$conditionB$adjPval<0.05&grepl(rownames
```

## 1.2 Median & robust summarization

Click to see background and code

### 1. Import Data

```
peptidesTable <- fread("https://raw.githubusercontent.com/statOmics/SGA2020/data/quantification/cptacAv
int64 <- which(sapply(peptidesTable,class) == "integer64")
for (j in int64) peptidesTable[[j]] <- as.numeric(peptidesTable[[j]])

quantCols <- grep("Intensity ", names(peptidesTable))

pe <- readQFeatures(
  assayData = peptidesTable,
  fnames = 1,
  quantCols = quantCols,
  name = "peptideRaw")

## Checking arguments.
## Loading data as a 'SummarizedExperiment' object.
## Formatting sample annotations (colData).
## Formatting data as a 'QFeatures' object.
## Setting assay rownames.

rm(peptidesTable)
gc()

##          used (Mb) gc trigger (Mb) limit (Mb) max used (Mb)
## Ncells  8068534 431.0  12731717 680.0      NA 12731717 680.0
## Vcells 16324116 124.6   33242060 253.7    16384 33241985 253.7

gc()

##          used (Mb) gc trigger (Mb) limit (Mb) max used (Mb)
## Ncells  8068466 431.0  12731717 680.0      NA 12731717 680.0
## Vcells 16317573 124.5   33242060 253.7    16384 33241985 253.7

cond <- which(
  strsplit(colnames(pe)[[1]][1], split = "")[[1]] == "A") # find where condition is stored

colData(pe)$condition <- substr(colnames(pe), cond, cond) |>
  unlist() |>
  as.factor()
```

## 2. Preprocessing

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

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

pe <- filterFeatures(
  pe, ~ Proteins != "" & ## Remove failed protein inference
    !grepl(";", Proteins)) ## Remove protein groups

## 'Proteins' found in 2 out of 2 assay(s).
pe <- filterFeatures(pe, ~Reverse != "+")

## 'Reverse' found in 2 out of 2 assay(s).
pe <- filterFeatures(pe, ~Potential.contaminant != "+")

## 'Potential.contaminant' found in 2 out of 2 assay(s).
# We filter out all peptides that have been seen in less than three samples
nObs <- 3
n <- ncol(pe[["peptideLog"]])
pNA <- (n-nObs)/n
pe <- filterNA(pe, pNA = pNA, i = "peptideLog")
nrow(pe[["peptideLog"]])

## [1] 5910

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

pe <- aggregateFeatures(pe,
  i = "peptideNorm",
  fcol = "Proteins",
  na.rm = TRUE,
  name = "proteinMedian",
  fun = matrixStats::colMedians)

## 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.

## Aggregated: 1/1

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

## 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.

## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): 'rlm' failed to converge in 20
```















[illegible]























[illegible]

[illegible]









```
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): 'rlm' failed to converge in 20
## steps
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): 'rlm' failed to converge in 20
## steps
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): 'rlm' failed to converge in 20
## steps
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): 'rlm' failed to converge in 20
## steps
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Warning in rlm.default(X, expression, ...): some of ... do not match
## Aggregated: 1/1
```

*# Comparisons are only valid for proteins for which we observed a protein intensity twice in each group*

```
nObs <- 4
n <- ncol(pe[["proteinMedian"]])
pNA <- (n-nObs)/n
pe <- filterNA(pe, pNA = pNA, i = "proteinMedian")

n <- ncol(pe[["proteinRobust"]])
pNA <- (n-nObs)/n
pe <- filterNA(pe, pNA = pNA, i = "proteinRobust")
```

### 3. Modeling and inference

```
pe <- msqrob(object = pe, i = "proteinMedian", formula = ~condition)
```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible][illegible]

[illegible]





[illegible]

[illegible]

[illegible]





[illegible]

[illegible]



[illegible]

```

## Warning: 'rlm' failed to converge in 1 steps
## Warning: 'rlm' failed to converge in 1 steps
## Warning: 'rlm' failed to converge in 1 steps

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

volcanoMedian <- ggplot(rowData(pe[["proteinMedian"]])$conditionB,
  aes(x = logFC, y = -log10(pval), color = adjPval < 0.05)) +
  geom_point(cex = 2.5) +
  scale_color_manual(values = alpha(c("black", "red"), 0.5)) +
  theme_minimal() +
  ggtitle(paste0("Median: TP = ", sum(rowData(pe[["proteinMedian"]])$conditionB$adjPval < 0.05 & grepl(rownames(pe), "proteinMedian"))))

volcanoRobust <- ggplot(rowData(pe[["proteinRobust"]])$conditionB,
  aes(x = logFC, y = -log10(pval), color = adjPval < 0.05)) +
  geom_point(cex = 2.5) +
  scale_color_manual(values = alpha(c("black", "red"), 0.5)) +
  theme_minimal() +
  ggtitle(paste0("Robust: TP = ", sum(rowData(pe[["proteinRobust"]])$conditionB$adjPval < 0.05 & grepl(rownames(pe), "proteinRobust"))))

ylims <- c(0,
  ceiling(max(c(-log10(rowData(peLFQ[["protein"]])$conditionB$pval),
    -log10(rowData(pe[["proteinMedian"]])$conditionB$pval),
    -log10(rowData(pe[["proteinRobust"]])$conditionB$pval)),
    na.rm=TRUE))
)

xlims <- max(abs(c(rowData(peLFQ[["protein"]])$conditionB$logFC,
  rowData(pe[["proteinMedian"]])$conditionB$logFC,
  rowData(pe[["proteinRobust"]])$conditionB$logFC)),
  na.rm=TRUE) * c(-1,1)

compBoxPlot <- rbind(rowData(peLFQ[["protein"]])$conditionB |> mutate(method="maxLFQ") |> rownames_to_column(var="protein"),
  rowData(pe[["proteinMedian"]])$conditionB |> mutate(method="median") |> rownames_to_column(var="protein"),
  rowData(pe[["proteinRobust"]])$conditionB |> mutate(method="robust") |> rownames_to_column(var="protein"))
compBoxPlot <- mutate(compBoxPlot, ups = grepl(protein, pattern="UPS"))
compBoxPlot <- ggplot(aes(x = method, y = logFC, fill = ups)) +
  geom_boxplot() +
  geom_hline(yintercept = log2(0.74 / .25), color = "#00BFC4") +
  geom_hline(yintercept = 0, color = "#F8766D")

```

### 1.3 Comparison summarization methods

```

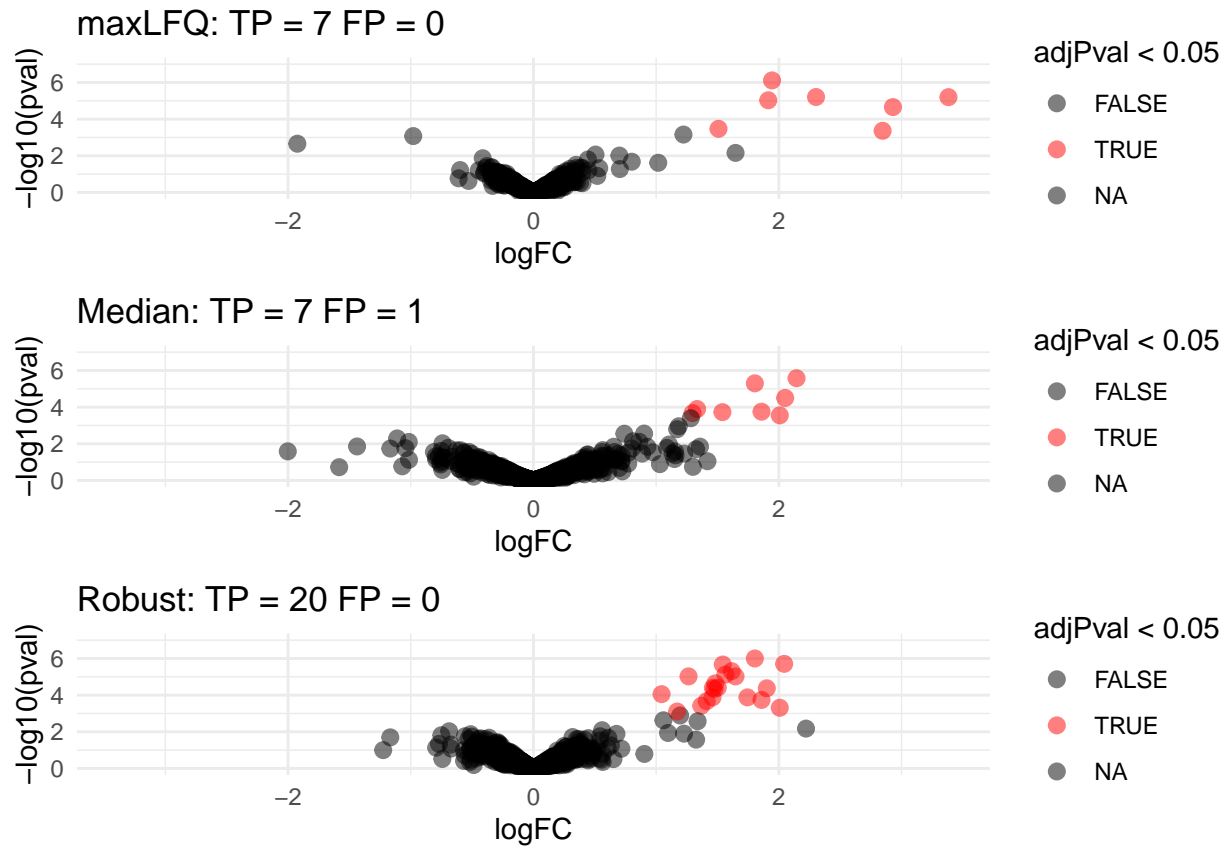
grid.arrange(volcanoLFQ + xlim(xlims) + ylim(ylims),
  volcanoMedian + xlim(xlims) + ylim(ylims),
  volcanoRobust + xlim(xlims) + ylim(ylims),
  ncol=1)

## Warning: Removed 41 rows containing missing values or values outside the scale range
## (`geom_point()`).

## Warning: Removed 30 rows containing missing values or values outside the scale range
## (`geom_point()`).

## Removed 30 rows containing missing values or values outside the scale range
## (`geom_point()`).

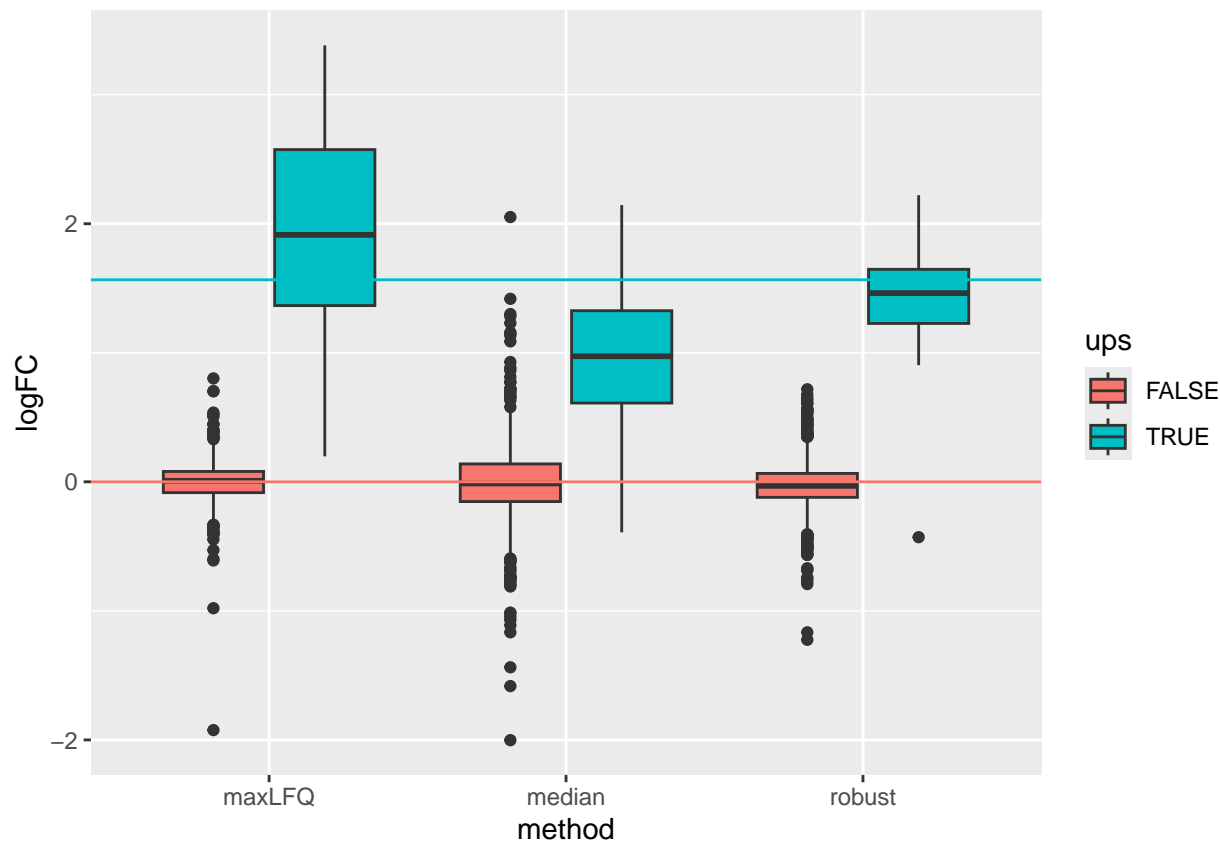
```



- Robust summarization: highest power and still good FDR control:  $FDP = \frac{0}{20} = 0$ .

compBoxPlot

```
## Warning: Removed 101 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```



- Median: biased logFC estimates for spike-in proteins
- maxLFQ: more variable logFC estimates for spike-in proteins

## 2 Full CPTAC study

### 2.1 Read data

Click to see background and code

1. We use a peptides.txt file from MS-data quantified with maxquant that contains MS1 intensities summarized at the peptide level.

```
peptidesTable <- fread("https://raw.githubusercontent.com/statOmics/PDA/data/quantification/fullCptacData/peptides.txt")
int64 <- which(sapply(peptidesTable, class) == "integer64")
for (j in int64) peptidesTable[[j]] <- as.numeric(peptidesTable[[j]])
```

2. Maxquant stores the intensity data for the different samples in columns that start with Intensity. We can retrieve the column names with the intensity data with the code below:

```
quantCols <- grep("Intensity ", names(peptidesTable))
```

3. Read the data and store it in QFeatures object

```
pe <- readQFeatures(
  assayData = peptidesTable,
  fnames = 1,
  quantCols = quantCols,
  name = "peptideRaw")
```

```
## Checking arguments.
## Loading data as a 'SummarizedExperiment' object.
## Formatting sample annotations (colData).
## Formatting data as a 'QFeatures' object.
## Setting assay rownames.
rm(peptidesTable)
gc()

##          used (Mb) gc trigger (Mb) limit (Mb) max used (Mb)
## Ncells  8068751 431.0   12731717 680.0         NA 12731717 680.0
## Vcells 17464472 133.3   33242060 253.7        16384 33241985 253.7

gc()

##          used (Mb) gc trigger (Mb) limit (Mb) max used (Mb)
## Ncells  8068604 431.0   12731717 680.0         NA 12731717 680.0
## Vcells 17459607 133.3   33242060 253.7        16384 33241985 253.7
```

## 2.2 Design

Click to see background and code

```
pe |> colnames()
```

```
## CharacterList of length 1
## [["peptideRaw"]] Intensity 6A_1 Intensity 6A_2 ... Intensity 6E_9
```

- Note, that the sample names include the spike-in condition.
- They also end on a number.
  - 1-3 is from lab 1,
  - 4-6 from lab 2 and
  - 7-9 from lab 3.
- We update the colData with information on the design

```
colData(pe)$lab <- rep(
  rep(
    paste0("lab",1:3),
    each=3),5) |>
as.factor()

colData(pe)$condition <- pe[["peptideRaw"]] |>
  colnames() |>
  substr(12,12) |>
  as.factor()

colData(pe)$spikeConcentration <- rep(
  c(A = 0.25, B = 0.74, C = 2.22, D = 6.67, E = 20),
  each = 9)
```

- We explore the colData

```
colData(pe)
```

```
## DataFrame with 45 rows and 3 columns
##           lab condition spikeConcentration
##           <factor>   <factor>           <numeric>
## Intensity 6A_1      lab1         A             0.25
## Intensity 6A_2      lab1         A             0.25
## Intensity 6A_3      lab1         A             0.25
## Intensity 6A_4      lab2         A             0.25
## Intensity 6A_5      lab2         A             0.25
## ...           ...           ...           ...
## Intensity 6E_5      lab2         E             20
## Intensity 6E_6      lab2         E             20
## Intensity 6E_7      lab3         E             20
## Intensity 6E_8      lab3         E             20
## Intensity 6E_9      lab3         E             20
```

## 2.3 Preprocessing

### 2.3.1 Log-transform

Click to see code to log-transform the data

- 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
```

- Logtransform data with base 2

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

### 2.3.2 Filtering

Click to see code to filter the data

1. Remove peptides that map to multiple proteins

We remove PSMs that could not be mapped to a protein or that map to multiple proteins (the protein identifier contains multiple identifiers separated by a ;).

```
pe <- filterFeatures(
  pe, ~ Proteins != "" & ## Remove failed protein inference
    !grepl(";", Proteins)) ## Remove protein groups
```

```
## 'Proteins' found in 2 out of 2 assay(s).
```

2. Remove reverse sequences (decoys) and contaminants

We now remove the contaminants, peptides that map to decoy sequences, and proteins which were only identified by peptides with modifications.

```
pe <- filterFeatures(pe, ~Reverse != "+")
```

```
## 'Reverse' found in 2 out of 2 assay(s).
```

```
pe <- filterFeatures(pe, ~ Potential.contaminant != "+")
```

```
## 'Potential.contaminant' found in 2 out of 2 assay(s).
```

3. Drop peptides that were identified less than three samples

We keep peptides that were observed at least three times. We tolerate the following proportion of NAs:  $pNA = (n-3)/n$ .

```
nObs <- 3
n <- ncol(pe[["peptideLog"]])
pNA <- (n-nObs)/n
pe <- filterNA(pe, pNA = pNA, i = "peptideLog")
nrow(pe[["peptideLog"]])
```

```
## [1] 10091
```

We keep 10091 peptides upon filtering.

## 2.4 Normalization

Click to see R-code to normalize the data

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

## 3 Peptide-level models

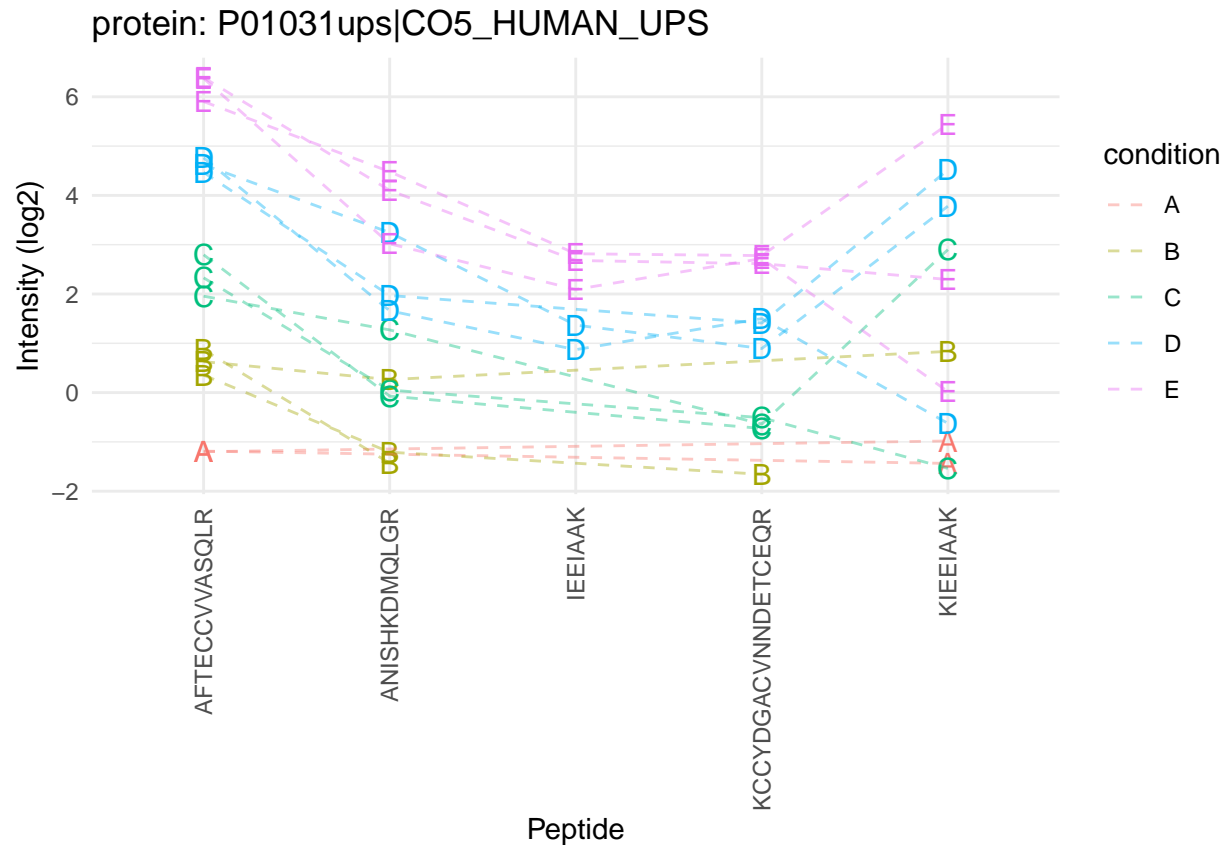
### 3.1 Summarization

Click to see code to make plot

```
prot <- "P01031ups|CO5_HUMAN_UPS"
data <- pe[["peptideNorm"]][
  rowData(pe[["peptideNorm"]])$Proteins == prot,
  colData(pe)$lab=="lab3"] |>
  assay() |>
  as.data.frame() |>
  rownames_to_column(var = "peptide") |>
  gather(sample, intensity, -peptide) |>
  mutate(condition = colData(pe)[sample,"condition"]) |>
  na.exclude()
sumPlot <- data |>
  ggplot(aes(x = peptide, y = intensity, color = condition, group = sample, label = condition), show.legend = FALSE) +
  geom_text(show.legend = FALSE) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  xlab("Peptide") +
  ylab("Intensity (log2)") +
  ggtitle(paste0("protein: ",prot))
```

Here, we will focus on the summarization of the intensities for protein P01031ups|CO5\_HUMAN\_UPS.

```
sumPlot +
  geom_line(linetype="dashed",alpha=.4)
```



### 3.1.1 Median summarization

We first evaluate median summarization for protein P01031ups|CO5\_HUMAN\_UPS.

Click to see code to make plot

```
dataHlp <- pe[["peptideNorm"]][
  rowData(pe[["peptideNorm"]])$Proteins == prot,
  colData(pe)$lab=="lab3"] |>
  assay()

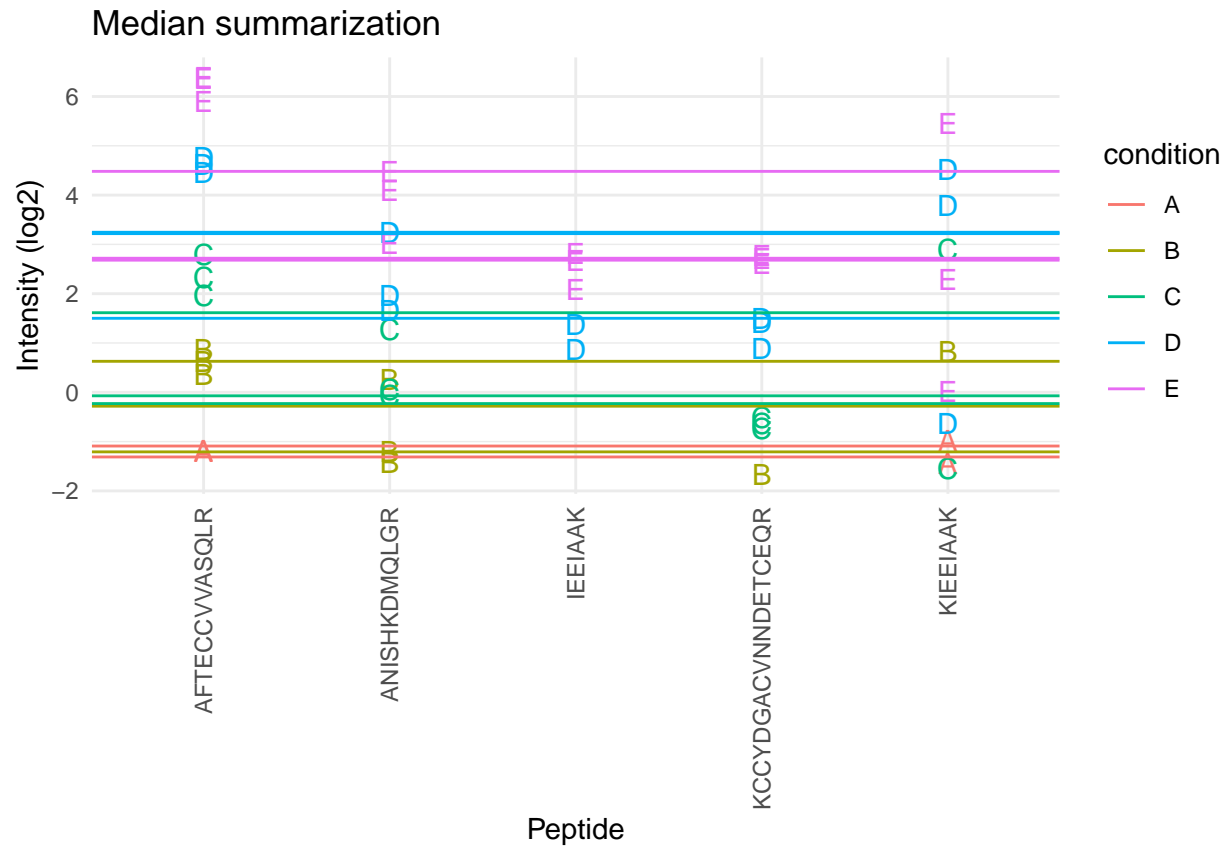
sumMedian <- data.frame(
  intensity= dataHlp
  |>
  colMedians(na.rm=TRUE)
,
  condition= colnames(dataHlp) |>
  substr(12,12) |>
  as.factor()
)

sumMedianPlot <- sumPlot +
  geom_hline(
    data = sumMedian,
    mapping = aes(yintercept=intensity,color=condition)) +
  ggtitle("Median summarization")
```



```
sumMedianPlot
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_hline()`).
```



- The sample medians are not a good estimate for the protein expression value.
- Indeed, they do not account for differences in peptide effects
- Peptides that ionize poorly are also picked up in samples with high spike-in concentration and not in samples with low spike-in concentration
- This introduces a bias.

### 3.1.2 Mean summarization

$$y_{ip} = \beta_i^{\text{sample}} + \epsilon_{ip}$$

Click to see code to make plot

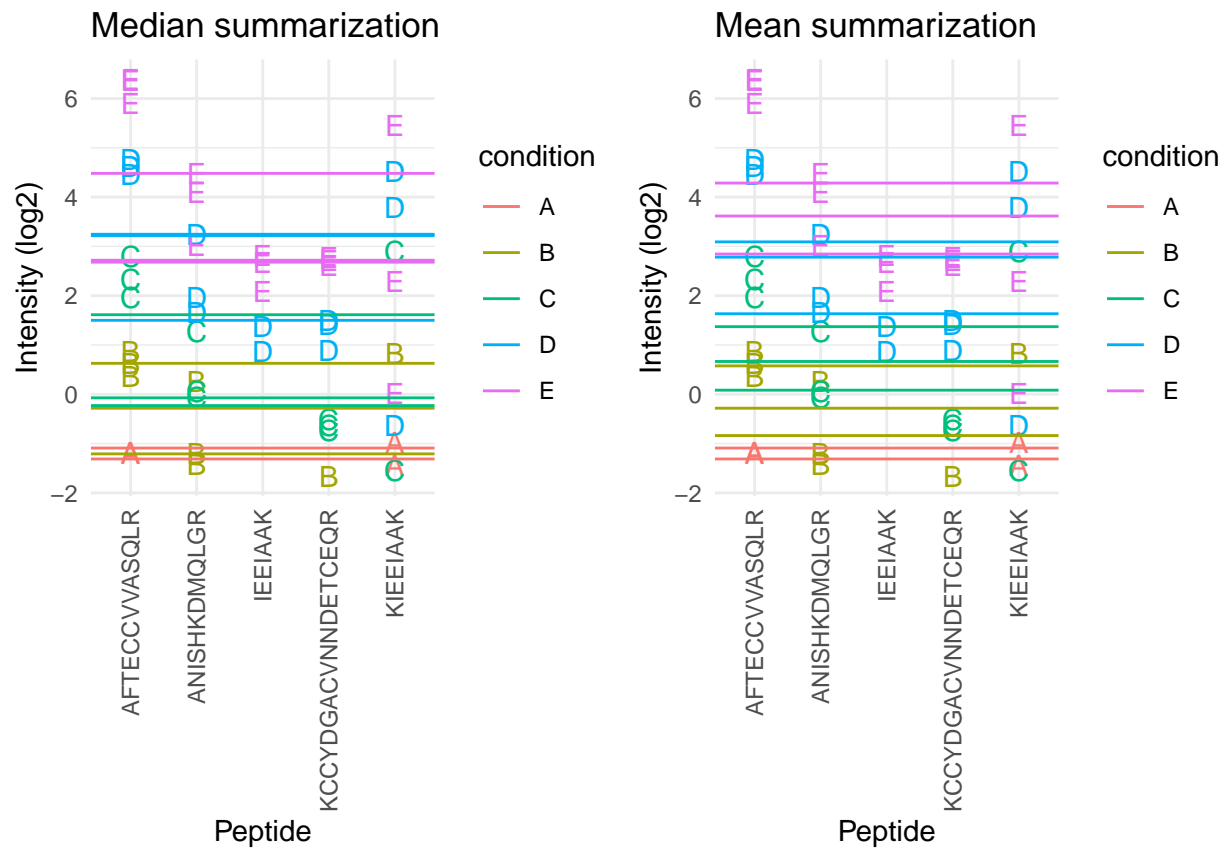
```
sumMeanMod <- lm(intensity ~ -1 + sample,data)
```

```
sumMean <- data.frame(
  intensity=sumMeanMod$coef[grep("sample",names(sumMeanMod$coef))],
  condition= names(sumMeanMod$coef)[grep("sample",names(sumMeanMod$coef))] |>
  substr(18,18) |>
  as.factor() )
```

```
sumMeanPlot <- sumPlot + geom_hline(
  data = sumMean,
  mapping = aes(yintercept=intensity,color=condition)) +
  ggtitle("Mean summarization")
```

```
grid.arrange(sumMedianPlot, sumMeanPlot, ncol=2)
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_hline()`).
```



### 3.1.3 Model based summarization

We can use a linear peptide-level model to estimate the protein expression value while correcting for the peptide effect, i.e.

$$y_{ip} = \beta_i^{\text{sample}} + \beta_p^{\text{peptide}} + \epsilon_{ip}$$

Click to see code to make plot

```
sumMeanPepMod <- lm(intensity ~ -1 + sample + peptide,data)
```

```
sumMeanPep <- data.frame(
  intensity = sumMeanPepMod$coef[grep("sample",names(sumMeanPepMod$coef))] +
    mean(data$intensity) -
    mean(sumMeanPepMod$coef[grep("sample",names(sumMeanPepMod$coef))]),
  condition = names(sumMeanPepMod$coef)[grep("sample",names(sumMeanPepMod$coef))])
```

```

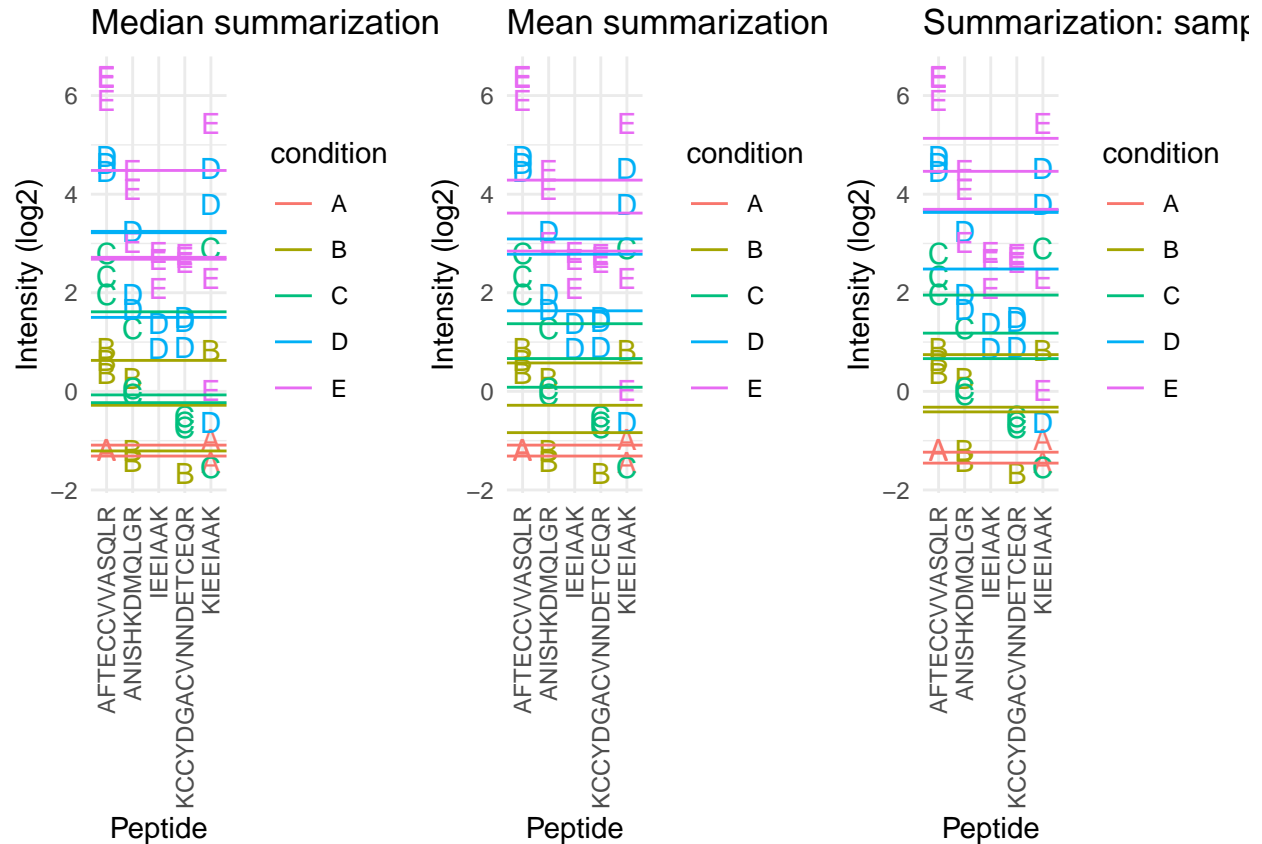
|> substr(18,18) |>
as.factor() )

fitLmPlot <- sumPlot + geom_line(
  data = data |>
    mutate(fit=sumMeanPepMod$fitted.values),
  mapping = aes(x=peptide, y=fit,color=condition, group=sample)) +
  ggtitle("fit: ~ sample + peptide")
sumLmPlot <- sumPlot + geom_hline(
  data = sumMeanPep,
  mapping = aes(yintercept=intensity,color=condition)) +
  ggtitle("Summarization: sample effect")

grid.arrange(sumMedianPlot, sumMeanPlot, sumLmPlot, nrow=1)

## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_hline()`).

```



- By correcting for the peptide species the protein expression values are much better separated and better reflect differences in abundance induced by the spike-in condition.
- Indeed, it shows that median expression and mean summarization that do not account for the peptide effect indeed overestimate the protein expression value in the small spike-in conditions and underestimate that in the large spike-in conditions.
- Still there seem to be some issues with samples that for which the expression values are not well

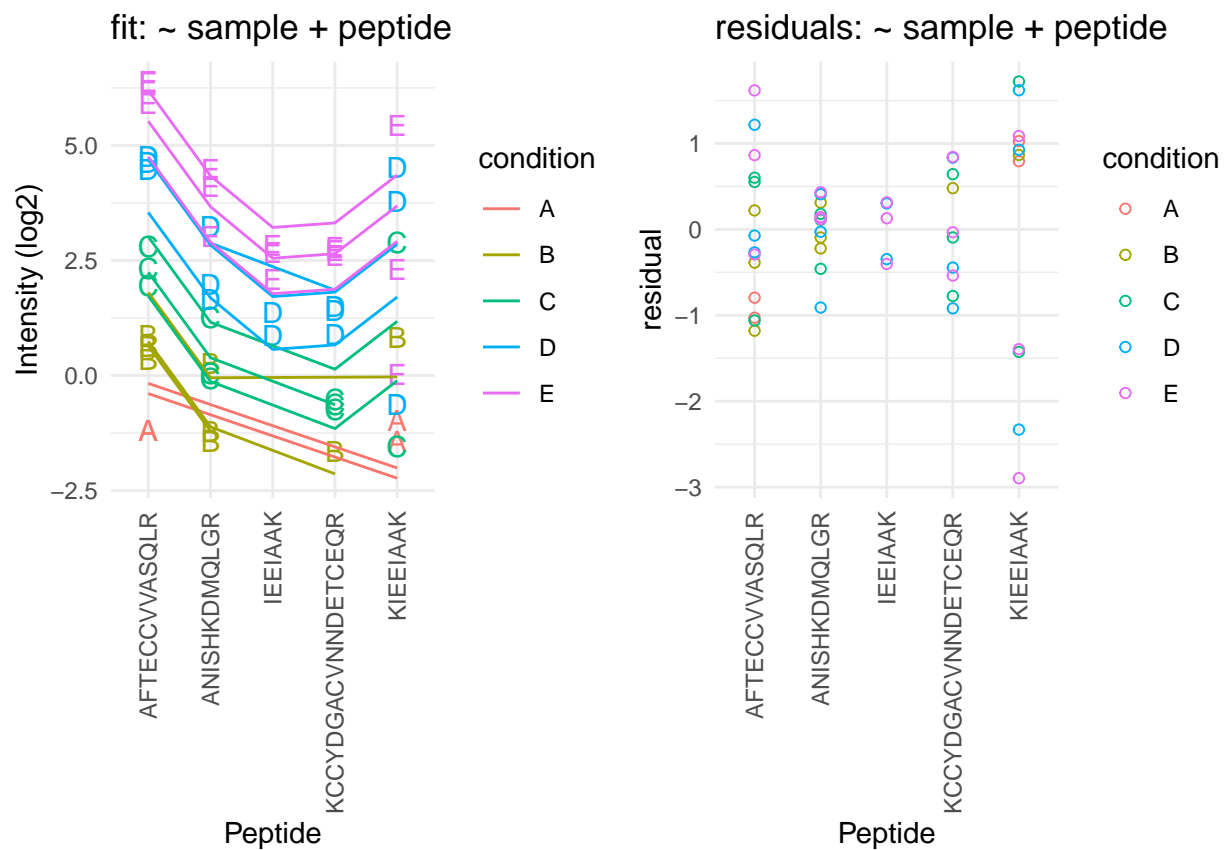
separated according to the spike-in condition.

A residual analysis clearly indicates potential issues:

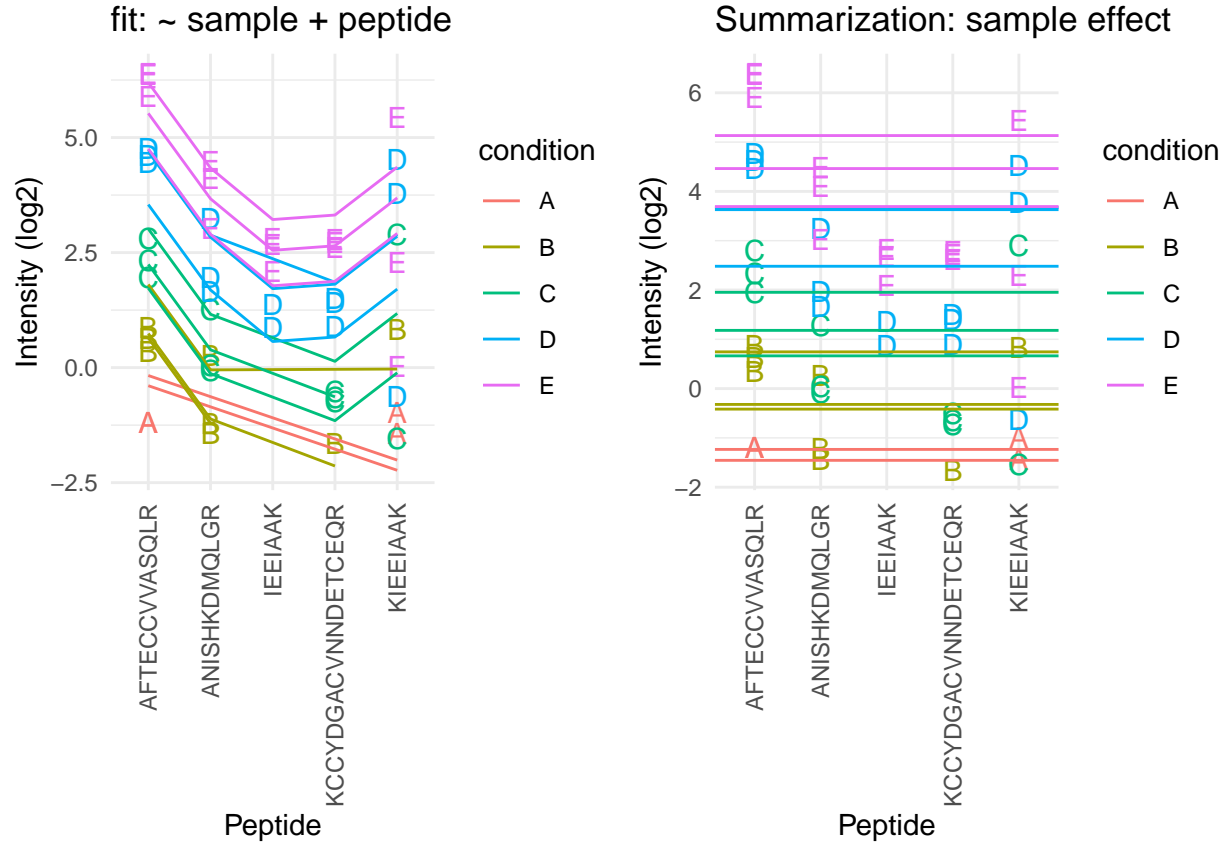
Click to see code to make plot

```
resPlot <- data |>
  mutate(res=sumMeanPepMod$residuals) |>
  ggplot(aes(x = peptide, y = res, color = condition, label = condition), show.legend = FALSE) +
  geom_point(shape=21) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
  xlab("Peptide") +
  ylab("residual") +
  ggtitle("residuals: ~ sample + peptide")

grid.arrange(fitLmPlot, resPlot, nrow = 1)
```



```
grid.arrange(fitLmPlot, sumLmPlot, nrow = 1)
```



- The residual plot shows some large outliers for peptide KIEEIAAK.
- Indeed, in the original plot the intensities for this peptide do not seem to line up very well with the concentration.
- This induces a bias in the summarization for some of the samples (e.g. for D and E)

### 3.1.4 Robust summarization using a peptide-level linear model

$$y_{ip} = \beta_i^{\text{sample}} + \beta_p^{\text{peptide}} + \epsilon_{ip}$$

- Ordinary least squares: estimate  $\beta$  that minimizes

$$\text{OLS} : \sum_{i,p} \epsilon_{ip}^2 = \sum_{i,p} (y_{ip} - \beta_i^{\text{sample}} - \beta_p^{\text{peptide}})^2$$

We replace OLS by M-estimation with loss function

$$\sum_{i,p} w_{ip} \epsilon_{ip}^2 = \sum_{i,p} w_{ip} (y_{ip} - \beta_i^{\text{sample}} - \beta_p^{\text{peptide}})^2$$

- Iteratively fit model with observation weights  $w_{ip}$  until convergence
- The weights are calculated based on standardized residuals

Click to see code to make plot

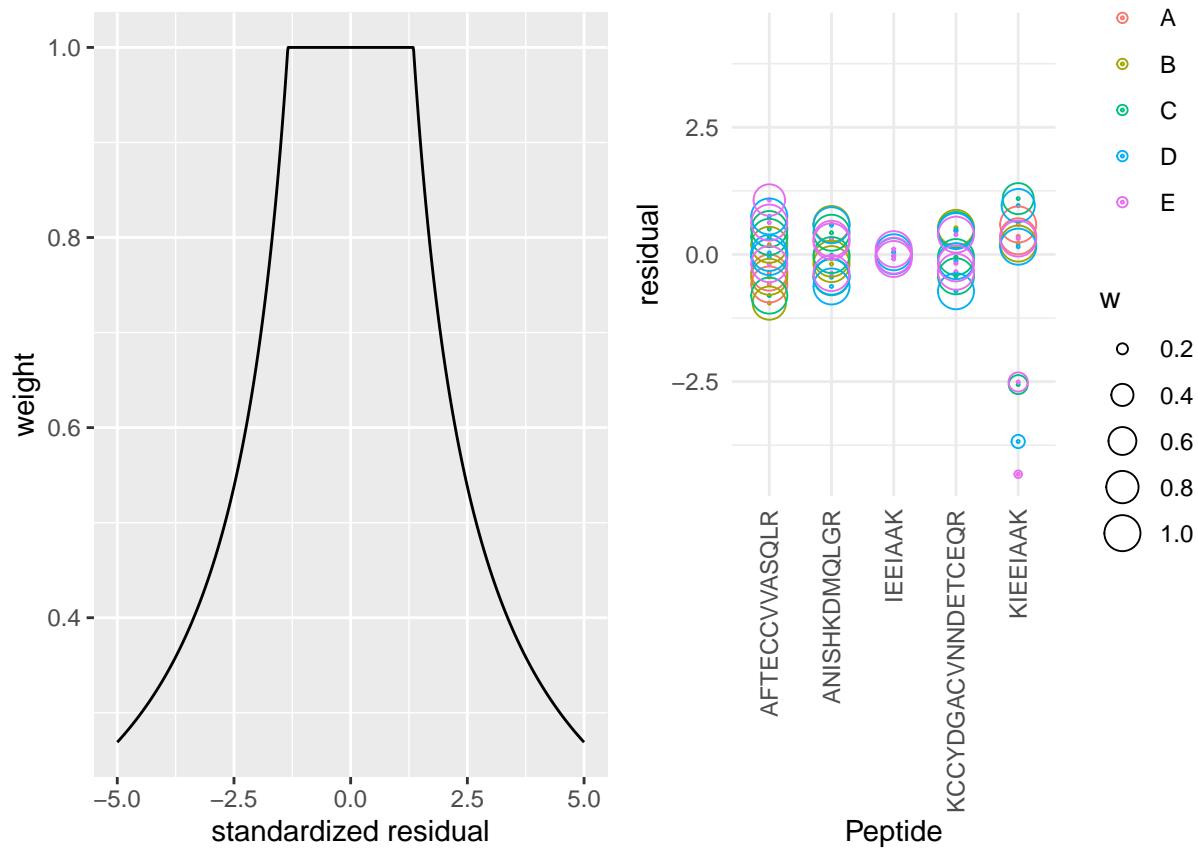
```
sumMeanPepRobMod <- MASS::rlm(intensity ~ -1 + sample + peptide, data)
resRobPlot <- data |>
  mutate(res = sumMeanPepRobMod$residuals,
         w = sumMeanPepRobMod$w) |>
  ggplot(aes(x = peptide, y = res, color = condition, label = condition, size = w), show.legend = FALSE) +
```

```

geom_point(shape=21,size=.2) +
geom_point(shape=21) +
theme_minimal() +
theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1)) +
xlab("Peptide") +
ylab("residual") +
ylim(c(-1,1)*max(abs(sumMeanPepRobMod$residuals)))
weightPlot <- data.frame(
  resid=seq(-5,5,.01),
  weight=MASS::psi.huber(seq(-5,5,.01))
) |>
ggplot(aes(resid,weight)) +
geom_line() +
xlab("standardized residual") +
ylab("weight")

```

```
grid.arrange(weightPlot,resRobPlot,nrow=1)
```



- We clearly see that the weights in the M-estimation procedure will down-weight errors associated with outliers for peptide KIEEIAAK.

Click to see code to make plot

```

sumMeanPepRob <- data.frame(
  intensity=sumMeanPepRobMod$coef[grep("sample",names(sumMeanPepRobMod$coef))] +
  mean(data$intensity) -
  mean(sumMeanPepRobMod$coef[grep("sample",names(sumMeanPepRobMod$coef))]),

```

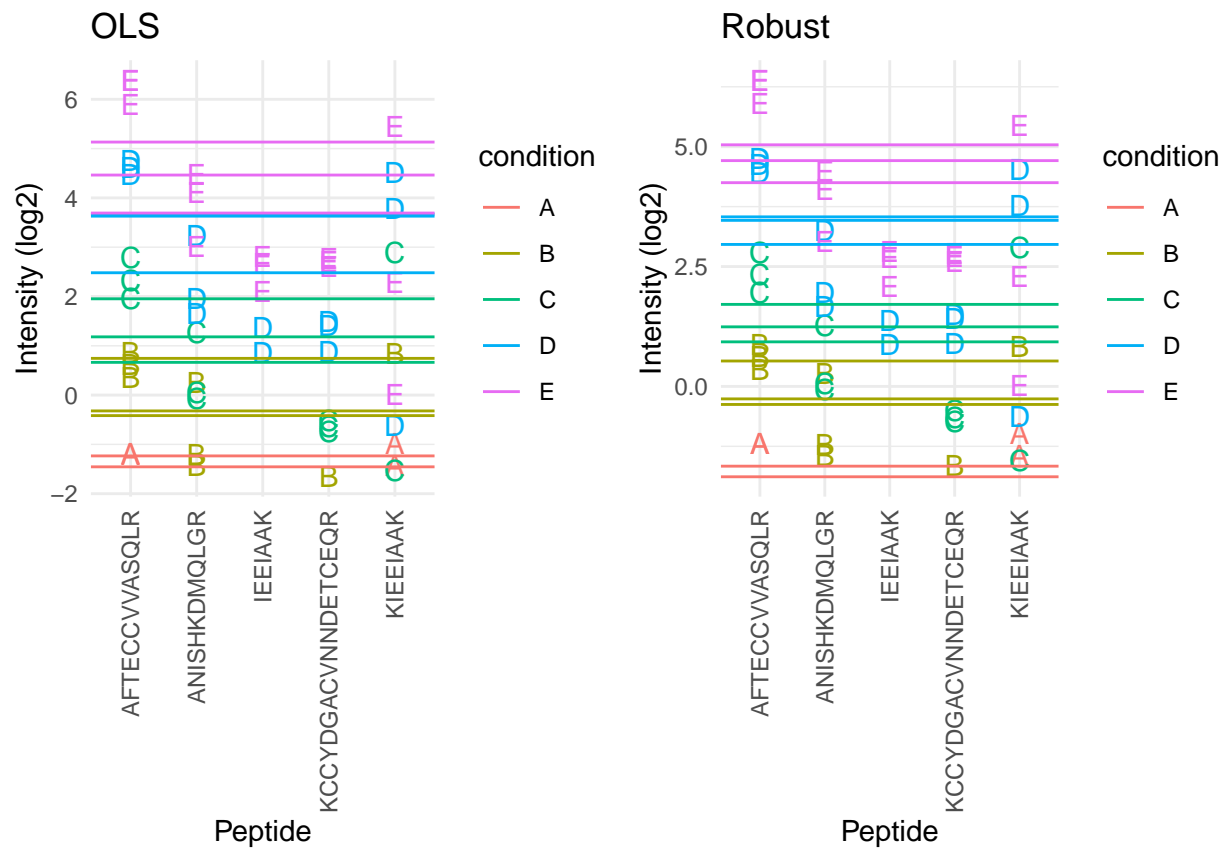
```

condition= names(sumMeanPepRobMod$coef)[grep("sample",names(sumMeanPepRobMod$coef))] |>
  substr(18,18) |> as.factor()
)

sumRlmPlot <- sumPlot +
  geom_hline(
    data=sumMeanPepRob,
    mapping=aes(yintercept=intensity,color=condition)) +
    ggtitle("Robust")

grid.arrange(sumLmPlot + ggtitle("OLS"), sumRlmPlot, nrow = 1)

```



- Robust regression results in a better separation between the protein expression values for the different samples according to their spike-in concentration.

### 3.1.5 Comparison summarization methods

- maxLFQ

**a**

&gt;P63208

MPSIKLQSSDGEIFEVDVEIAKQSVTIKTMLEDLGMDDEGDD  
 DPVPLPNVNAAILKKVIQWCTHHKDDPPPPEDDENKEKRTDD  
IPVWDQEFLKVDQGTLELILAANYLDIKGLLDVTCKTVANM  
IKGKTPEEIRKTFNIKNDFTEEEEAQVRKENQWCEEK

**b**

Peptide species	Sequence	Charge	Mod.
<b>P<sub>1</sub></b>	LQSSDGEIFEVDVEIAK	2	—
<b>P<sub>2</sub></b>	LQSSDGEIFEVDVEIAK	3	—
<b>P<sub>3</sub></b>	RTDDIPVWDQEFLK	2	—
<b>P<sub>4</sub></b>	TVANMIK	2	—
<b>P<sub>5</sub></b>	TVANMIK	2	Oxid.
<b>P<sub>6</sub></b>	TPEEIRK	3	—
<b>P<sub>7</sub></b>	NDFTEEEEAQVR	2	—

**c**

Sample	<b>P<sub>1</sub></b>	<b>P<sub>2</sub></b>	<b>P<sub>3</sub></b>	<b>P<sub>4</sub></b>	<b>P<sub>5</sub></b>	<b>P<sub>6</sub></b>	<b>P<sub>7</sub></b>
<b>A</b>		+				+	
<b>B</b>		+	+			+	
<b>C</b>	+	+	+	+		+	+
<b>D</b>	+	+		+		+	+



- MS-stats also uses a robust peptide level model to perform the summarization, however, they typically first impute missing values
- Proteus high-flyer method: mean of three peptides with highest intensity

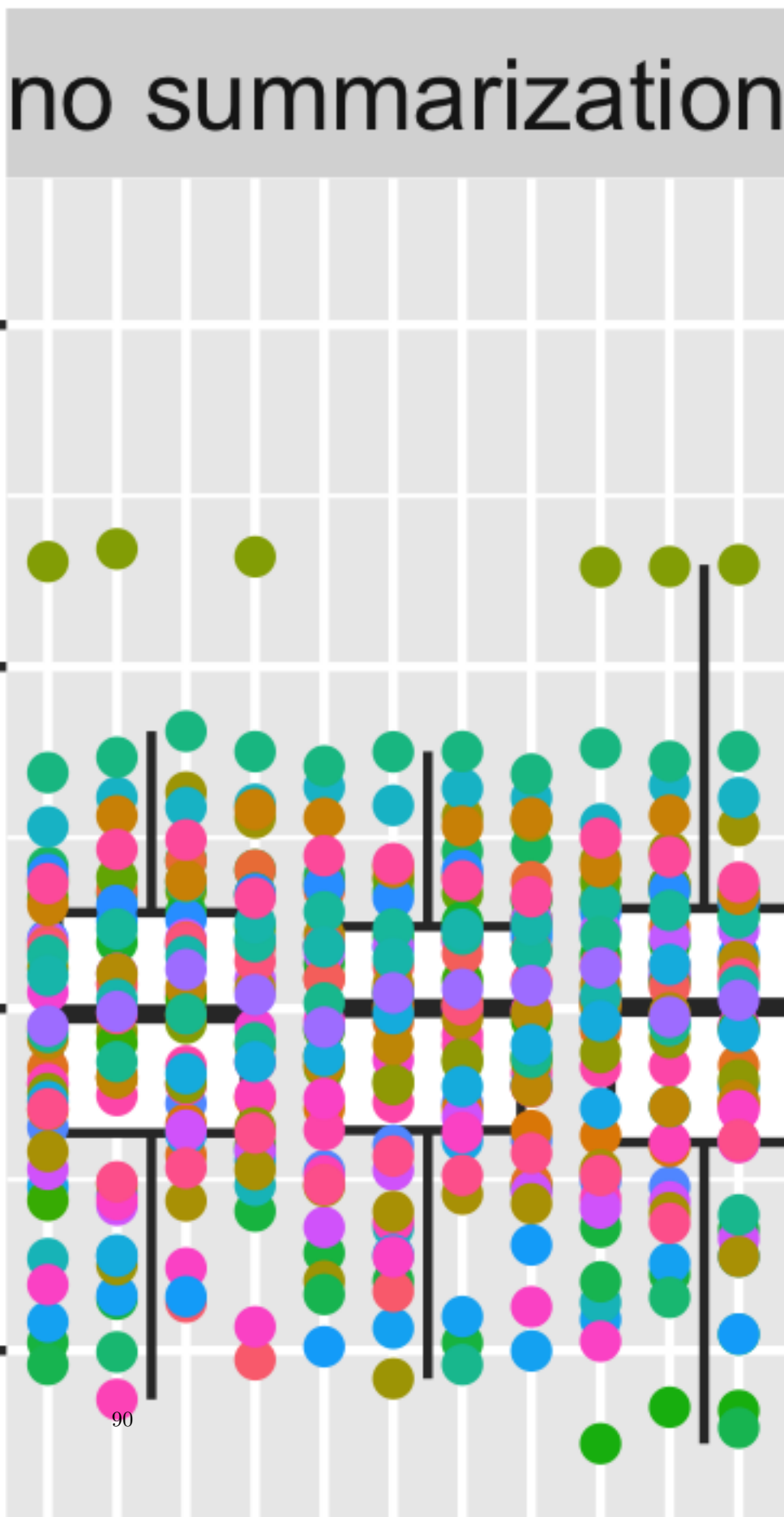
no summarization

30.0

27.5

25.0

22.5



- (Sticker et al. 2020)
- doi: <https://doi.org/10.1074/mcp.RA119.001624>
- [pdf](#)

### 3.2 Estimation of differential abundance using peptide level model

- Instead of summarising the data we can also directly model the data at the peptide-level.
- But, we will have to address the pseudo-replication.

$$y_{iclp} = \beta_0 + \beta_c^{\text{condition}} + \beta_l^{\text{lab}} + \beta_p^{\text{peptide}} + u_s^{\text{sample}} + \epsilon_{iclp}$$

- protein-level
  - $\beta_c^{\text{condition}}$ : spike-in condition  $c = b, \dots, e$
  - $\beta_l^{\text{lab}}$ : lab effect  $l = l_2 \dots l_3$
  - $u_r^{\text{run}} \sim N(0, \sigma_{\text{run}}^2) \rightarrow$  random effect addresses pseudo-replication
- peptide-level
  - $\beta_p^{\text{peptide}}$ : peptide effect
  - $\epsilon_{rp} \sim N(0, \sigma_\epsilon^2)$  within sample (run) error

- DA estimates:

$$\begin{aligned}\log_2 FC_{B-A} &= \beta_B^{\text{condition}} \\ \log_2 FC_{C-B} &= \beta_C^{\text{condition}} - \beta_B^{\text{condition}}\end{aligned}$$

- Mixed peptide-level models are implemented in msqrob2
- It has the advantages that
  1. it correctly addresses the difference levels of variability in the data
  2. it avoids summarization and therefore also accounts for the difference in the number of peptides that are observed in each sample
  3. more powerful analysis
- It has the disadvantage that
  1. protein summaries are no longer available for plotting
  2. it is difficult to correctly specify the degrees of freedom for the test-statistic leading to inference that is too liberal in experiments with small sample size
  3. sometimes sample level random effect variance are estimated to be zero, then the pseudo-replication is not addressed leading to inference that is too liberal for these specific proteins
  4. they are much more difficult to disseminate to users with limited background in statistics

Hence, for this course we opted to use peptide-level models for summarization, but not for directly inferring on the differential expression at the protein-level.

## References

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.