Wrapup Blocking

Lieven Clement

Note, that this dataset has been acquired with DDA (data dependent acquisition). It was searched with MaxQuant. The analysis starts from the peptides.txt file, which is a table in the output directory of MaxQuant with the data summarized at the peptide-level.

The preprocessing steps are slightly different. The modeling concepts, however, are also applicable to DIA data.

Movie clip

Import Data and Preprocessing

Data

Click to see code

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

library("BiocFileCache")
bfc <- BiocFileCache()
peptidesFile <- bfcrpath(bfc,"https://github.com/statOmics/msqrob2data/raw/refs/heads/main/d
peptidesFile2 <- bfcrpath(bfc,"https://github.com/statOmics/msqrob2data/raw/refs/heads/main/
peptidesFile3 <- bfcrpath(bfc,"https://github.com/statOmics/msqrob2data/raw/refs/heads/main/

ecols <- grep("Intensity\\.", names(read.delim(peptidesFile)))
qf <- readQFeatures(
  assayData = read.delim(peptidesFile),
```

```
  fnames = 1,
  quantCols = ecols,
  name = "peptides_raw")

ecols2 <- grep("Intensity\\.", names(read.delim(peptidesFile2)))
qf2 <- readQFeatures(
  assayData = read.delim(peptidesFile2),
  fnames = 1,
  quantCols = ecols2,
  name = "peptides_raw")

ecols3 <- grep("Intensity\\.", names(read.delim(peptidesFile3)))
qf3 <- readQFeatures(
  assayData = read.delim(peptidesFile3),
  fnames = 1,
  quantCols = ecols3,
  name = "peptides_raw")

### Design
colData(qf)$celltype <- substr(
  colnames(qf[["peptides_raw"]]),
  11,
  14) %>%
  unlist %>%
  as.factor

colData(qf)$mouse <- qf[[1]] %>%
  colnames %>%
  strsplit(split="[.]")  %>%
  sapply(function(x) x[3]) %>%
  as.factor

colData(qf2)$celltype <- substr(
  colnames(qf2[["peptides_raw"]]),
  11,
  14) %>%
  unlist %>%
  as.factor

colData(qf2)$mouse <- qf2[[1]] %>%
  colnames %>%
  strsplit(split="[.]")  %>%
```

```
  sapply(function(x) x[3]) %>%
  as.factor

colData(qf3)$celltype <- substr(
  colnames(qf3[["peptides_raw"]]),
  11,
  14) %>%
  unlist %>%
  as.factor

colData(qf3)$mouse <- qf3[[1]] %>%
  colnames %>%
  strsplit(split="[.]")  %>%
  sapply(function(x) x[3]) %>%
  as.factor
```

Preprocessing

Log-transform

Click to see code to log-transform the data

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

```
qf <- zeroIsNA(qf, "peptides_raw") # convert 0 to NA

qf2 <- zeroIsNA(qf2, "peptides_raw") # convert 0 to NA

qf3 <- zeroIsNA(qf3, "peptides_raw") # convert 0 to NA
```

- Logtransform data with base 2

```
qf <- logTransform(qf, base = 2, i = "peptides_raw", name = "peptides_log")

qf2 <- logTransform(qf2, base = 2, i = "peptides_raw", name = "peptides_log")

qf3 <- logTransform(qf3, base = 2, i = "peptides_raw", name = "peptides_log")
```

Filtering

Click to see details on filtering

1. Handling overlapping protein groups

#Only use proteotypic proteins

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

qf2 <-  filterFeatures(
  qf2, ~ Proteins != "" #& ## Remove failed protein inference
  #  !grepl(";", Proteins)
  )

qf3 <-  filterFeatures(
  qf3, ~ Proteins != "" # & ## Remove failed protein inference
    #!grepl(";", Proteins)
  )
```

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.

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

qf2 <- filterFeatures(qf2,~Reverse != "+")
qf2 <- filterFeatures(qf2,~ Potential.contaminant != "+")

qf3 <- filterFeatures(qf3,~Reverse != "+")
qf3 <- filterFeatures(qf3,~ Potential.contaminant != "+")
```

3. Drop peptides that were only identified in one sample

We keep peptides that were observed at least twice.

```r
nObs <- 3
n1 <- ncol(qf[["peptides_log"]])
n2 <- ncol(qf2[["peptides_log"]])
n3 <- ncol(qf3[["peptides_log"]])

(qf <- filterNA(qf, i = "peptides_log", pNA = (n1 - nObs) / n1))
```

```
An instance of class QFeatures (type: bulk) with 2 sets:

 [1] peptides_raw: SummarizedExperiment with 52720 rows and 8 columns
 [2] peptides_log: SummarizedExperiment with 40822 rows and 8 columns
```

```r
(qf2 <- filterNA(qf2, i = "peptides_log", pNA = (n2 - nObs) / n2))
```

```
An instance of class QFeatures (type: bulk) with 2 sets:

 [1] peptides_raw: SummarizedExperiment with 52720 rows and 8 columns
 [2] peptides_log: SummarizedExperiment with 39932 rows and 8 columns
```

```r
(qf3 <- filterNA(qf3, i = "peptides_log", pNA = (n3 - nObs) / n3))
```

```
An instance of class QFeatures (type: bulk) with 2 sets:

 [1] peptides_raw: SummarizedExperiment with 52720 rows and 14 columns
 [2] peptides_log: SummarizedExperiment with 46201 rows and 14 columns
```

Normalization

Click to see code to normalize the data

```r
qf <- sweep( #4. Subtract log2 norm factor column-by-column (MARGIN = 2)
  qf,
  MARGIN = 2,
  STATS = nfLogMedianOfRatios(qf, i="peptides_log"), #1.
  i = "peptides_log",
  name = "peptides_norm"
)

qf2 <- sweep( #4. Subtract log2 norm factor column-by-column (MARGIN = 2)
```

```
  qf2,
  MARGIN = 2,
  STATS = nfLogMedianOfRatios(qf2, i="peptides_log"), #1.
  i = "peptides_log",
  name = "peptides_norm"
)

qf3 <- sweep( #4. Subtract log2 norm factor column-by-column (MARGIN = 2)
  qf3,
  MARGIN = 2,
  STATS = nfLogMedianOfRatios(qf3, i="peptides_log"), #1.
  i = "peptides_log",
  name = "peptides_norm"
)
```

Summarization

Click to see code to summarize the data

```
qf <- aggregateFeatures(qf,
 i = "peptides_norm",
 fcol = "Proteins",
 name = "proteins",
 fun = function(X) iq::maxLFQ(X)$estimate
)
```

```
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
qf2 <- aggregateFeatures(qf2,
 i = "peptides_norm",
 fcol = "Proteins",
 name = "proteins",
 fun = function(X) iq::maxLFQ(X)$estimate
 )
```

```
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

```
qf3 <- aggregateFeatures(qf3,
 i = "peptides_norm",
 fcol = "Proteins",
 name = "proteins",
 fun = function(X) iq::maxLFQ(X)$estimate
 )
```

```
  |
  |                                                                      |   0%
  |
  |======================================================================| 100%
```

Data Exploration: what is impact of blocking?

Click to see code

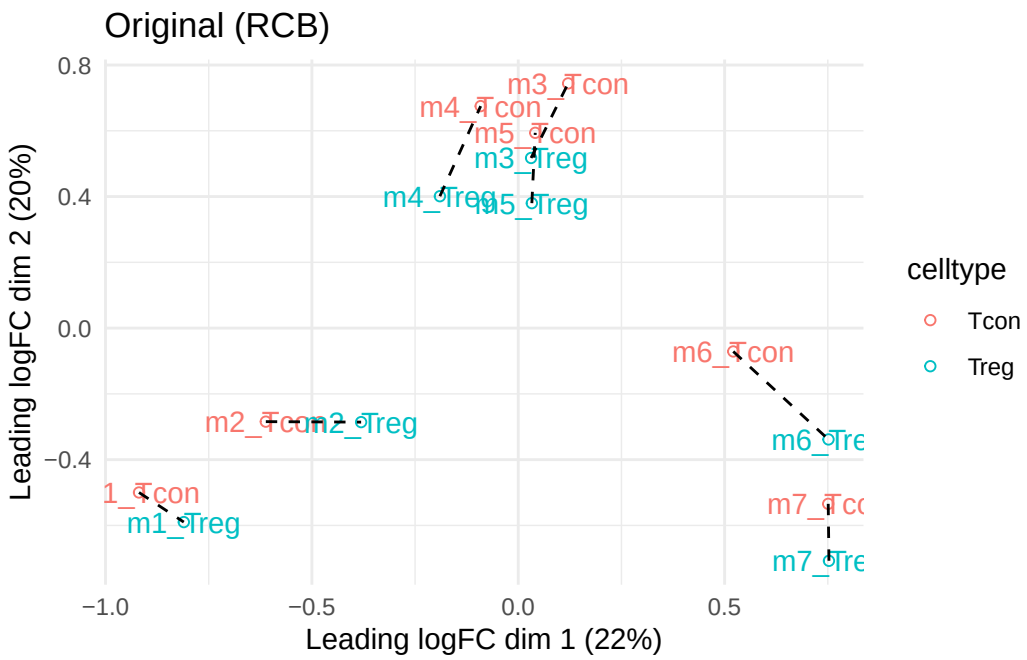
```
levels(colData(qf3)$mouse) <- paste0("m",1:7)
mdsObj3 <- plotMDS(assay(qf3[["proteins"]]), plot = FALSE)
mdsOrig <- colData(qf3) %>%
  as.data.frame %>%
  mutate(mds1 = mdsObj3$x,
         mds2 = mdsObj3$y,
         lab = paste(mouse,celltype,sep="_")) %>%
  ggplot(aes(x = mds1, y = mds2, label = lab, color = celltype, group = mouse)) +
  geom_text(show.legend = FALSE) +
  geom_point(shape = 21) +
  geom_line(color = "black", linetype = "dashed") +
  xlab(
    paste0(
      mdsObj3$axislabel,
      " ",
      1,
      " (",
      paste0(
```

```
        round(mdsObj3$var.explained[1] *100,0),
        "%"
        ),
      ")"
      )
    ) +
  ylab(
    paste0(
      mdsObj3$axislabel,
      " ",
      2,
      " (",
      paste0(
        round(mdsObj3$var.explained[2] *100,0),
        "%"
        ),
      ")"
      )
    ) +
  ggtitle("Original (RCB)")+
  theme_minimal()

levels(colData(qf)$mouse) <- paste0("m",1:4)
mdsObj <- plotMDS(assay(qf[["proteins"]]), plot = FALSE)
mdsRCB <- colData(qf) %>%
  as.data.frame %>%
  mutate(mds1 = mdsObj$x,
         mds2 = mdsObj$y,
         lab = paste(mouse,celltype,sep="_")) %>%
  ggplot(aes(x = mds1, y = mds2, label = lab, color = celltype, group = mouse)) +
  geom_text(show.legend = FALSE) +
  geom_point(shape = 21) +
  geom_line(color = "black", linetype = "dashed") +
  xlab(
    paste0(
      mdsObj$axislabel,
      " ",
      1,
      " (",
      paste0(
        round(mdsObj$var.explained[1] *100,0),
        "%"
```

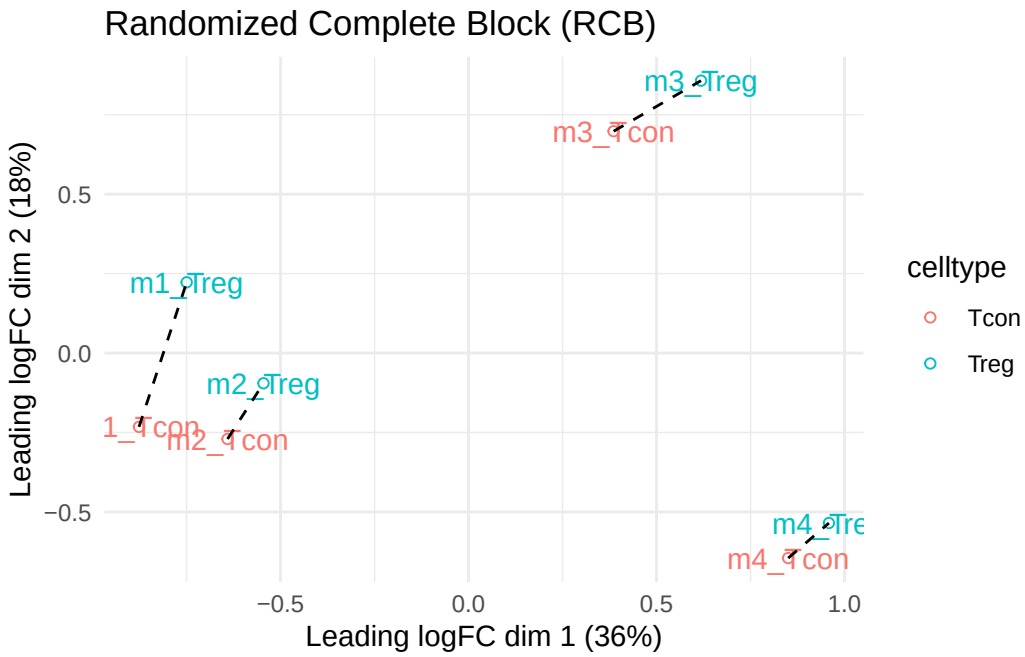
```
        ),
      ")"
      )
    ) +
  ylab(
    paste0(
      mdsObj$axislabel,
      " ",
      2,
      " (",
      paste0(
        round(mdsObj$var.explained[2] *100,0),
        "%"
        ),
      ")"
      )
    ) +
  ggtitle("Randomized Complete Block (RCB)")+
  theme_minimal()


levels(colData(qf2)$mouse) <- paste0("m",1:8)
mdsObj2 <- plotMDS(assay(qf2[["proteins"]]), plot = FALSE)
mdsCRD <- colData(qf2) %>%
  as.data.frame %>%
  mutate(mds1 = mdsObj2$x,
         mds2 = mdsObj2$y,
         lab = paste(mouse,celltype,sep="_")) %>%
  ggplot(aes(x = mds1, y = mds2, label = lab, color = celltype, group = mouse)) +
  geom_text(show.legend = FALSE) +
  geom_point(shape = 21) +
  xlab(
    paste0(
      mdsObj$axislabel,
      " ",
      1,
      " (",
      paste0(
        round(mdsObj2$var.explained[1] *100,0),
        "%"
        ),
      ")"
```

```
      )
    ) +
  ylab(
    paste0(
      mdsObj$axislabel,
      " ",
      2,
      " (",
      paste0(
        round(mdsObj2$var.explained[2] *100,0),
        "%"
        ),
      ")"
      )
    ) +
  ggtitle("Completely Randomized Design (CRD)") +
  theme_minimal()
```

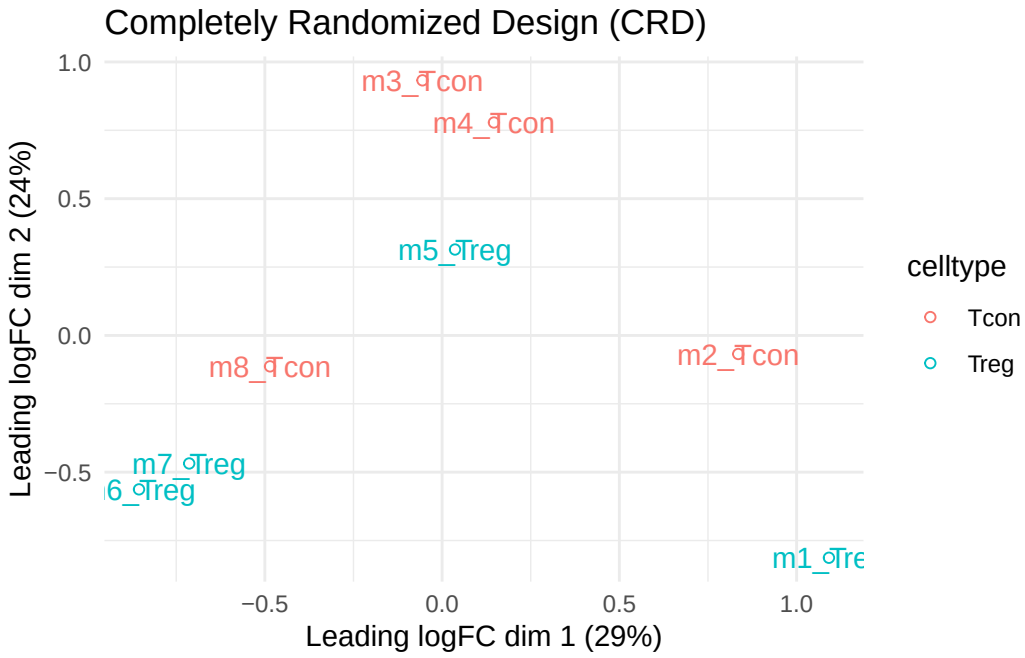
```
mdsOrig
```



```
mdsRCB
```




```
mdsCRD
```



- We observe that the leading fold change is according to mouse
- In the second dimension we see a separation according to cell-type
- With the Randomized Complete Block design (RCB) we can remove the mouse effect from the analysis!

Modeling and inference

RCB analysis

```
qf <- msqrob(
  object = qf,
  i = "proteins",
  formula = ~ celltype + mouse)
```

RCB wrong analysis

```
qf <- msqrob(
  object = qf,
  i = "proteins",
  formula = ~ celltype, modelColumnName = "wrongModel")
```

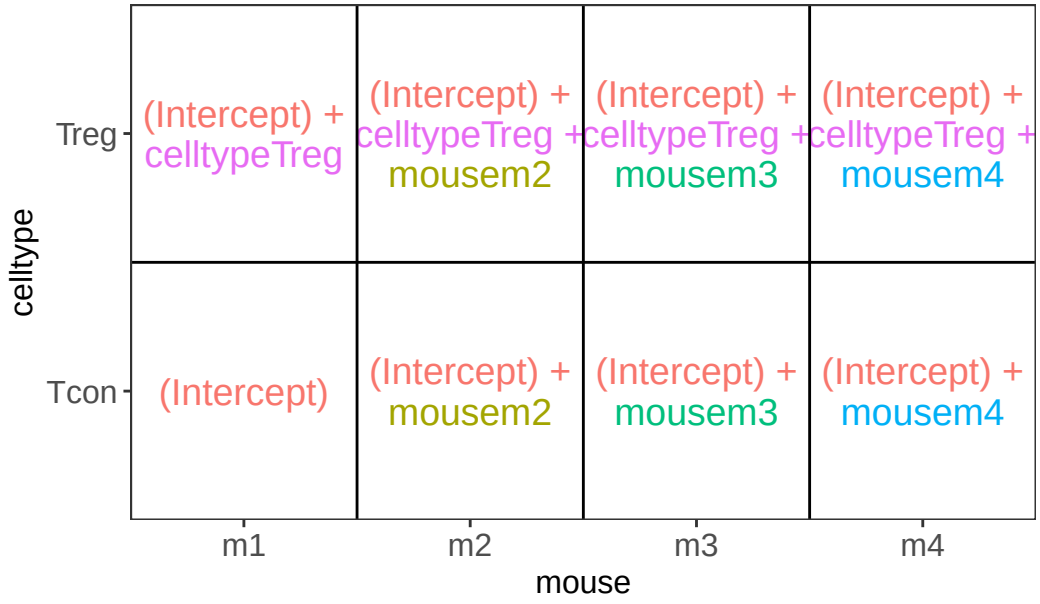
CRD analysis

```
qf2 <- msqrob(
  object = qf2,
  i = "proteins",
  formula = ~ celltype)
```

Inference

```
library(ExploreModelMatrix)
VisualizeDesign(colData(qf),~ celltype + mouse)$plotlist
```

```
[[1]]
```




```
VisualizeDesign(colData(qf2),~ celltype)$plotlist
```

```
[[1]]
```

```
L <- makeContrast("celltypeTreg = 0", parameterNames = c("celltypeTreg"))
qf <- hypothesisTest(object = qf, i = "proteins", contrast = L)
qf <- hypothesisTest(object = qf, i = "proteins", contrast = L, modelColumn = "wrongModel", r
qf2 <- hypothesisTest(object = qf2, i = "proteins", contrast = L)
```

Advantage of Blocking: comparison between designs

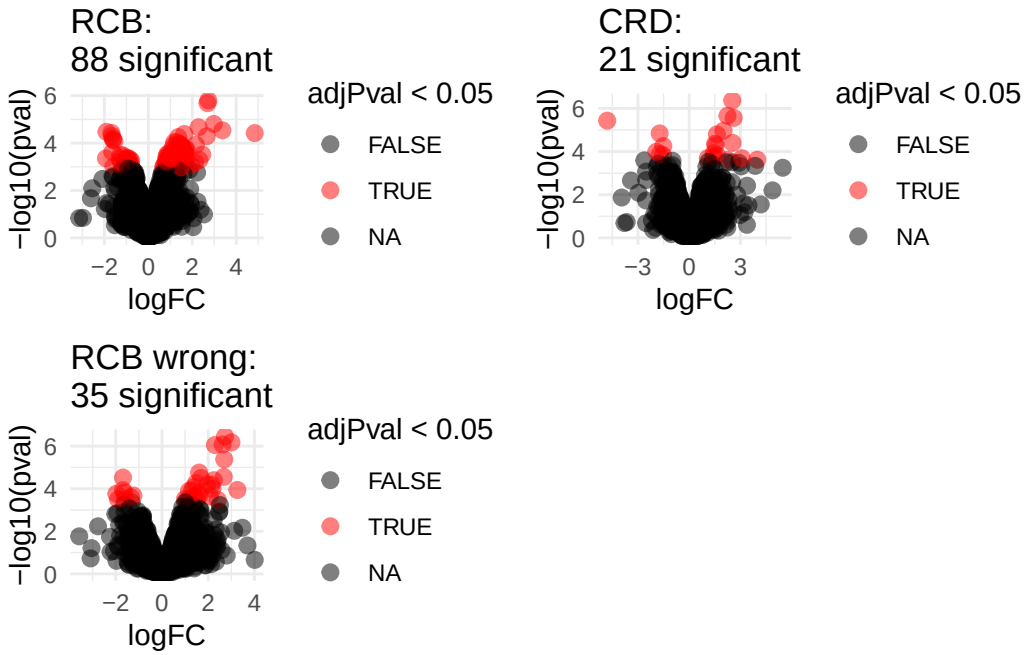
Volcano plots

Click to see code

```
volcanoRCB <- ggplot(
    rowData(qf[["proteins"]])$celltypeTreg,
    aes(x = logFC, y = -log10(pval), color = adjPval < 0.05)
) +
    geom_point(cex = 2.5) +
    scale_color_manual(values = alpha(c("black", "red"), 0.5)) +
    theme_minimal() +
    ggtitle(paste0("RCB: \n",
                sum(rowData(qf[["proteins"]])$celltypeTreg$adjPval<0.05,na.rm=TRUE),
            " significant"))

volcanoRCBwrong <- ggplot(
    rowData(qf[["proteins"]])$wrongcelltypeTreg,
    aes(x = logFC, y = -log10(pval), color = adjPval < 0.05)
) +
    geom_point(cex = 2.5) +
    scale_color_manual(values = alpha(c("black", "red"), 0.5)) +
    theme_minimal() +
    ggtitle(paste0("RCB wrong: \n",
                sum(rowData(qf[["proteins"]])$wrongcelltypeTreg$adjPval<0.05,na.rm=TRUE),
            " significant"))

volcanoCRD <- ggplot(
    rowData(qf2[["proteins"]])$celltypeTreg,
    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("CRD: \n",
                   sum(rowData(qf2[["proteins"]])$celltypeTreg$adjPval<0.05,na.rm=TRUE),
            " significant"))
```

```
grid.arrange(volcanoRCB,volcanoCRD, volcanoRCBwrong,ncol=2)
```



Anova table: Q7TPR4, Alpha-actinin-1

Disclaimer: the Anova analysis is only for didactical purposes. In practice we assess the hypotheses using msqrob2.

- We illustrate the power gain of blocking using an Anova analysis on 1 protein.
- Note, that msqrob2 will perform a similar analysis, but, it uses robust regression and it uses an empirical Bayes estimator for the variance.

```
prot <- "Q7TPR4"
dataHlp <- colData(qf) %>%
  as.data.frame %>%
  mutate(intensity=assay(qf[["proteins"]])[prot,],
         intensityCRD=assay(qf2[["proteins"]])[prot,])
```

```
  anova(lm(intensity~ celltype + mouse, dataHlp))
```

```
Analysis of Variance Table

Response: intensity
          Df Sum Sq Mean Sq  F value   Pr(>F)
celltype   1 5.1780  5.1780 144.0491 0.001244 **
mouse      3 0.8749  0.2916   8.1127 0.059640 .
Residuals  3 0.1078  0.0359
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
  anova(lm(intensity~ celltype,dataHlp))
```

```
Analysis of Variance Table

Response: intensity
          Df Sum Sq Mean Sq F value   Pr(>F)
celltype   1 5.1780  5.1780  31.615 0.001352 **
Residuals  6 0.9827  0.1638
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
  anova(lm(intensityCRD~ celltype,dataHlp))
```

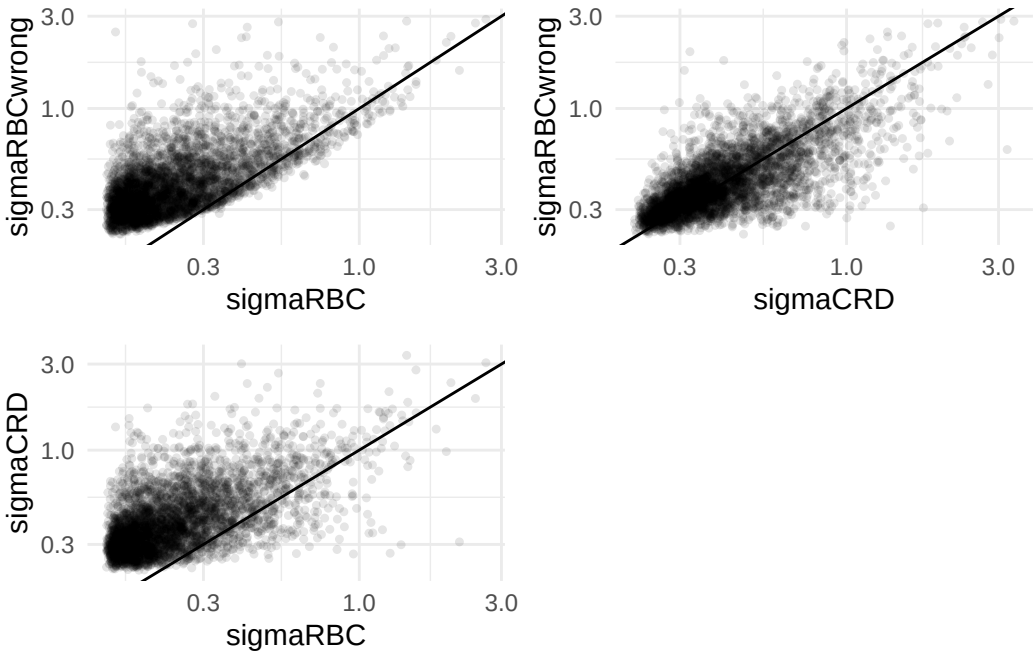
```
Analysis of Variance Table

Response: intensityCRD
          Df Sum Sq Mean Sq F value    Pr(>F)
celltype   1 5.8564  5.8564  43.269 0.0005922 ***
Residuals  6 0.8121  0.1354
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Comparison Empirical Bayes standard deviation in msqrob2

Click to see code

```
accessions <- rownames(qf[["proteins"]])[rownames(qf[["proteins"]])%in%rownames(qf2[["protein
dat <- data.frame(
sigmaRBC = sapply(rowData(qf[["proteins"]])$msqrobModels[accessions], getSigmaPosterior),
sigmaRBCwrong = sapply(rowData(qf[["proteins"]])$wrongModel[accessions], getSigmaPosterior),
sigmaCRD <- sapply(rowData(qf2[["proteins"]])$msqrobModels[accessions], getSigmaPosterior)
)

 plotRBCvsWrong <- ggplot(data = dat, aes(sigmaRBC, sigmaRBCwrong)) +
    geom_point(alpha = 0.1, shape = 20) +
    scale_x_log10() +
    scale_y_log10() +
    geom_abline(intercept=0,slope=1) +
   theme_minimal()
plotCRDvsWrong <- ggplot(data = dat, aes(sigmaCRD, sigmaRBCwrong)) +
    geom_point(alpha = 0.1, shape = 20) +
    scale_x_log10() +
    scale_y_log10() +
    geom_abline(intercept=0,slope=1) +   theme_minimal()

plotRBCvsCRD <- ggplot(data = dat, aes(sigmaRBC, sigmaCRD)) +
    geom_point(alpha = 0.1, shape = 20) +
    scale_x_log10() +
    scale_y_log10() +
    geom_abline(intercept=0,slope=1) +    theme_minimal()
```

```
grid.arrange(
  plotRBCvsWrong,
  plotCRDvsWrong,
  plotRBCvsCRD,
  nrow=2)
```



- We clearly observe that the standard deviation of the protein expression in the RCB is smaller for the majority of the proteins than that obtained with the CRD
- The standard deviation of the protein expression RCB where we perform a wrong analysis without considering the blocking factor according to mouse is much larger for the majority of the proteins than that obtained with the correct analysis.
- Indeed, when we ignore the blocking factor in the RCB design we do not remove the variability according to mouse from the analysis and the mouse effect is absorbed in the error term. The standard deviation then becomes very comparable to that observed in the completely randomised design where we could not remove the mouse effect from the analysis.
- Why are some of the standard deviations for the RCB with the correct analysis larger than that of the RCB with the incorrect analysis that ignored the mouse blocking factor?
- Can you think of a reason why it would not be useful to block on a particular factor?