Analysis and Exploration of Proteomics Data

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

Proteomics Data Analysis

Below are shown the steps typically used to transform, process and analyze proteomics data. The analysis steps have been applied over the *EGF-driven protein synthesis* case-study data from D.A. Rothenberg et al. A Proteomics Approach to Profiling the Temporal Translational Response to Stress and Growth. iScience. 2018; 9:367-381.

We mainly rely DEP R-package (Zhang et.al.) for integrated analysis workflow for robust and reproducible analysis of mass spectrometry proteomics data for differential protein expression or differential enrichment.

Loading of R-Packages

We start by loading the R-packages we need to use for our analysis:

```
library("DEP")
library("readr")
library("vsn")
library("dplyr")
library("tidyr")
library("limma")
library("ggplot2")
library("ggrepel")
library("knitr")
library("tidyverse")
```

Loading of the data and initial processing

We load the raw protein intensities:

```
data <- read.delim("../Data/protein_intensities.txt")
colnames(data)
```

```
##  [1] "Gene"         "Accession"    "Sequence"     "PBS_30_Rep1"  "PBS_60_Rep1"
##  [6] "PBS_90_Rep1"  "PBS_120_Rep1" "PBS_150_Rep1" "EGF_30_Rep1"  "EGF_60_Rep1"
## [11] "EGF_90_Rep1"  "EGF_120_Rep1" "EGF_150_Rep1" "PBS_30_Rep2"  "PBS_60_Rep2"
## [16] "PBS_90_Rep2"  "PBS_120_Rep2" "PBS_150_Rep2" "EGF_30_Rep2"  "EGF_60_Rep2"
## [21] "EGF_90_Rep2"  "EGF_120_Rep2" "EGF_150_Rep2" "PBS_30_Rep3"  "PBS_60_Rep3"
## [26] "PBS_90_Rep3"  "PBS_120_Rep3" "PBS_150_Rep3" "EGF_30_Rep3"  "EGF_60_Rep3"
## [31] "EGF_90_Rep3"  "EGF_120_Rep3" "EGF_150_Rep3" "PBS_30_Rep4"  "PBS_60_Rep4"
## [36] "PBS_90_Rep4"  "PBS_120_Rep4" "PBS_150_Rep4" "EGF_30_Rep4"  "EGF_60_Rep4"
## [41] "EGF_90_Rep4"  "EGF_120_Rep4" "EGF_150_Rep4"
```

```
head(data)
```

```
##     Gene   Accession                     Sequence PBS_30_Rep1 PBS_60_Rep1
```

```
## 1  HSPE1  CH10_HUMAN                   GGEIQPVSVK       98880      133900
## 2  HSPE1  CH10_HUMAN                VLQATVVAVGSGSK      215980      282280
## 3 EFTUD2  U5S1_HUMAN                 IAVEPVNPSELPK       20970       37250
## 4  YWHAB 1433B_HUMAN                       SELVQK      157680      215370
## 5  YWHAB 1433B_HUMAN TAFDEAIAELDTLNEESYKDSTLIMQLLR       44780       65620
## 6  YWHAB 1433B_HUMAN                   YLSEVASGDNK       42580       57410
##   PBS_90_Rep1 PBS_120_Rep1 PBS_150_Rep1 EGF_30_Rep1 EGF_60_Rep1 EGF_90_Rep1
## 1       92870       111600       147600      132300      190600      104200
## 2      190020       220860       258580      261080      354470      234470
## 3       22680        27430        33200       29590       45670       32610
## 4      172330       178840       232800      172670      250300      168350
## 5       46350        54980        71120       46100       84290       56390
## 6       44390        50640        56210       44140       75060       50310
##   EGF_120_Rep1 EGF_150_Rep1 PBS_30_Rep2 PBS_60_Rep2 PBS_90_Rep2 PBS_120_Rep2
## 1       137000       138900          NA          NA          NA           NA
## 2       273450       298530          NA          NA          NA           NA
## 3        36300        45390       71200       72900       84900        75100
## 4       213260       225100          NA          NA          NA           NA
## 5        67730        81740          NA          NA          NA           NA
## 6        50150        55100          NA          NA          NA           NA
##   PBS_150_Rep2 EGF_30_Rep2 EGF_60_Rep2 EGF_90_Rep2 EGF_120_Rep2 EGF_150_Rep2
## 1           NA          NA          NA          NA           NA           NA
## 2           NA          NA          NA          NA           NA           NA
## 3        76400       85800       85300      104000       108000        85000
## 4           NA          NA          NA          NA           NA           NA
## 5           NA          NA          NA          NA           NA           NA
## 6           NA          NA          NA          NA           NA           NA
##   PBS_30_Rep3 PBS_60_Rep3 PBS_90_Rep3 PBS_120_Rep3 PBS_150_Rep3 EGF_30_Rep3
## 1          NA          NA          NA           NA           NA          NA
## 2          NA          NA          NA           NA           NA          NA
## 3          NA          NA          NA           NA           NA          NA
## 4          NA          NA          NA           NA           NA          NA
## 5          NA          NA          NA           NA           NA          NA
## 6          NA          NA          NA           NA           NA          NA
##   EGF_60_Rep3 EGF_90_Rep3 EGF_120_Rep3 EGF_150_Rep3 PBS_30_Rep4 PBS_60_Rep4
## 1          NA          NA           NA           NA      637000      787000
## 2          NA          NA           NA           NA      142000      171000
## 3          NA          NA           NA           NA      214000      266000
## 4          NA          NA           NA           NA          NA          NA
## 5          NA          NA           NA           NA       23000       29900
## 6          NA          NA           NA           NA          NA          NA
##   PBS_90_Rep4 PBS_120_Rep4 PBS_150_Rep4 EGF_30_Rep4 EGF_60_Rep4 EGF_90_Rep4
## 1      644000       858000      1090000      792000      981000      994000
## 2      158000       194000       213000      167000      208000      189000
## 3      234000       264000       319000      255000      305000      266000
## 4          NA           NA           NA          NA          NA          NA
## 5       29500        21700        36600       30900       35400       31200
## 6          NA           NA           NA          NA          NA          NA
##   EGF_120_Rep4 EGF_150_Rep4
## 1      1060000       961000
## 2       211000       168000
## 3       297000       252000
## 4           NA           NA
## 5        28100        28800
```

```
## 6           NA           NA
```

We then search for duplicated protein identifiers on the data table (Accesion ID), summarize them and then we make unique names using the annotation in the "Accession" column as primary names and the annotation in "Sequence" as to identify for individual quantified peptide sequences.

```
# Are there any duplicated protein names?
data$Accession %>% duplicated() %>% any()
```

```
## [1] TRUE
```

```
# Make a table of duplicated protein names
data %>% group_by(Accession) %>% summarize(frequency = n()) %>%
  arrange(desc(frequency)) %>% filter(frequency > 1)
```

```
## # A tibble: 949 x 2
##    Accession   frequency
##    <chr>           <int>
##  1 PLEC_HUMAN        115
##  2 AHNK_HUMAN         86
##  3 FLNA_HUMAN         59
##  4 MYH9_HUMAN         52
##  5 FLNB_HUMAN         44
##  6 TLN1_HUMAN         43
##  7 VINC_HUMAN         34
##  8 MUC16_HUMAN        32
##  9 FAS_HUMAN          31
## 10 KI67_HUMAN         29
## # ... with 939 more rows
```

```
# Make unique names using the annotation in the "Accession" and "Sequence" column.
data_unique <- make_unique(data, "Accession", "Sequence", delim = ";")
head(data_unique)
```

```
##     Gene   Accession                     Sequence PBS_30_Rep1 PBS_60_Rep1
## 1  HSPE1  CH10_HUMAN                  GGEIQPVSVK       98880      133900
## 2  HSPE1  CH10_HUMAN                VLQATVVAVGSGSK      215980      282280
## 3 EFTUD2  U5S1_HUMAN                 IAVEPVNPSELPK       20970       37250
## 4  YWHAB 1433B_HUMAN                      SELVQK      157680      215370
## 5  YWHAB 1433B_HUMAN TAFDEAIAELDTLNEESYKDSTLIMQLLR       44780       65620
## 6  YWHAB 1433B_HUMAN                   YLSEVASGDNK       42580       57410
##   PBS_90_Rep1 PBS_120_Rep1 PBS_150_Rep1 EGF_30_Rep1 EGF_60_Rep1 EGF_90_Rep1
## 1       92870       111600       147600      132300      190600      104200
## 2      190020       220860       258580      261080      354470      234470
## 3       22680        27430        33200       29590       45670       32610
## 4      172330       178840       232800      172670      250300      168350
## 5       46350        54980        71120       46100       84290       56390
## 6       44390        50640        56210       44140       75060       50310
##   EGF_120_Rep1 EGF_150_Rep1 PBS_30_Rep2 PBS_60_Rep2 PBS_90_Rep2 PBS_120_Rep2
## 1       137000       138900          NA          NA          NA           NA
## 2       273450       298530          NA          NA          NA           NA
## 3        36300        45390       71200       72900       84900        75100
## 4       213260       225100          NA          NA          NA           NA
## 5        67730        81740          NA          NA          NA           NA
## 6        50150        55100          NA          NA          NA           NA
##   PBS_150_Rep2 EGF_30_Rep2 EGF_60_Rep2 EGF_90_Rep2 EGF_120_Rep2 EGF_150_Rep2
## 1           NA          NA          NA          NA           NA           NA
```

```
## 2           NA          NA          NA          NA          NA          NA
## 3        76400       85800       85300      104000      108000       85000
## 4           NA          NA          NA          NA          NA          NA
## 5           NA          NA          NA          NA          NA          NA
## 6           NA          NA          NA          NA          NA          NA
##   PBS_30_Rep3 PBS_60_Rep3 PBS_90_Rep3 PBS_120_Rep3 PBS_150_Rep3 EGF_30_Rep3
## 1          NA          NA          NA           NA          NA          NA
## 2          NA          NA          NA           NA          NA          NA
## 3          NA          NA          NA           NA          NA          NA
## 4          NA          NA          NA           NA          NA          NA
## 5          NA          NA          NA           NA          NA          NA
## 6          NA          NA          NA           NA          NA          NA
##   EGF_60_Rep3 EGF_90_Rep3 EGF_120_Rep3 EGF_150_Rep3 PBS_30_Rep4 PBS_60_Rep4
## 1          NA          NA           NA           NA      637000      787000
## 2          NA          NA           NA           NA      142000      171000
## 3          NA          NA           NA           NA      214000      266000
## 4          NA          NA           NA           NA          NA          NA
## 5          NA          NA           NA           NA       23000       29900
## 6          NA          NA           NA           NA          NA          NA
##   PBS_90_Rep4 PBS_120_Rep4 PBS_150_Rep4 EGF_30_Rep4 EGF_60_Rep4 EGF_90_Rep4
## 1      644000       858000      1090000      792000      981000      994000
## 2      158000       194000       213000      167000      208000      189000
## 3      234000       264000       319000      255000      305000      266000
## 4          NA           NA           NA          NA          NA          NA
## 5       29500        21700        36600       30900       35400       31200
## 6          NA           NA           NA          NA          NA          NA
##   EGF_120_Rep4 EGF_150_Rep4          name                          ID
## 1      1060000       961000    CH10_HUMAN                 GGEIQPVSVK
## 2       211000       168000  CH10_HUMAN.1                VLQATVVAVGSGSK
## 3       297000       252000    U5S1_HUMAN                 IAVEPVNPSELPK
## 4           NA           NA   1433B_HUMAN                       SELVQK
## 5        28100        28800 1433B_HUMAN.1 TAFDEAIAELDTLNEESYKDSTLIMQLLR
## 6           NA           NA 1433B_HUMAN.2                   YLSEVASGDNK
```

```
# Are there any duplicated names?
data$name %>% duplicated() %>% any()
```

```
## [1] FALSE
```

Experimental Design

We make an experimental design matrix where we indicate the condition and replicate ID's for each sample.

```
# We show the sample ID's
labels <- colnames(data_unique)[4:43]
print(labels)
```

```
##  [1] "PBS_30_Rep1"  "PBS_60_Rep1"  "PBS_90_Rep1"  "PBS_120_Rep1" "PBS_150_Rep1"
##  [6] "EGF_30_Rep1"  "EGF_60_Rep1"  "EGF_90_Rep1"  "EGF_120_Rep1" "EGF_150_Rep1"
## [11] "PBS_30_Rep2"  "PBS_60_Rep2"  "PBS_90_Rep2"  "PBS_120_Rep2" "PBS_150_Rep2"
## [16] "EGF_30_Rep2"  "EGF_60_Rep2"  "EGF_90_Rep2"  "EGF_120_Rep2" "EGF_150_Rep2"
## [21] "PBS_30_Rep3"  "PBS_60_Rep3"  "PBS_90_Rep3"  "PBS_120_Rep3" "PBS_150_Rep3"
## [26] "EGF_30_Rep3"  "EGF_60_Rep3"  "EGF_90_Rep3"  "EGF_120_Rep3" "EGF_150_Rep3"
## [31] "PBS_30_Rep4"  "PBS_60_Rep4"  "PBS_90_Rep4"  "PBS_120_Rep4" "PBS_150_Rep4"
## [36] "EGF_30_Rep4"  "EGF_60_Rep4"  "EGF_90_Rep4"  "EGF_120_Rep4" "EGF_150_Rep4"
```

```r
# We setup the experimental design matrix
experimental_design <- matrix(data = , nrow = length(labels), ncol = 3)
experimental_design[, 1] <- labels
experimental_design[, 2] <- sapply(strsplit(x = labels, split = "_Rep", fixed = TRUE),
                                   "[", 1)
experimental_design[, 3] <- sapply(strsplit(x = labels, split = "_", fixed = TRUE),
                                   "[", 3)
colnames(experimental_design) <- c("label", "condition", "replicate")
experimental_design <- as.data.frame(experimental_design)
print(experimental_design)
```

```
##           label condition replicate
## 1   PBS_30_Rep1    PBS_30      Rep1
## 2   PBS_60_Rep1    PBS_60      Rep1
## 3   PBS_90_Rep1    PBS_90      Rep1
## 4  PBS_120_Rep1   PBS_120      Rep1
## 5  PBS_150_Rep1   PBS_150      Rep1
## 6   EGF_30_Rep1    EGF_30      Rep1
## 7   EGF_60_Rep1    EGF_60      Rep1
## 8   EGF_90_Rep1    EGF_90      Rep1
## 9  EGF_120_Rep1   EGF_120      Rep1
## 10 EGF_150_Rep1   EGF_150      Rep1
## 11  PBS_30_Rep2    PBS_30      Rep2
## 12  PBS_60_Rep2    PBS_60      Rep2
## 13  PBS_90_Rep2    PBS_90      Rep2
## 14 PBS_120_Rep2   PBS_120      Rep2
## 15 PBS_150_Rep2   PBS_150      Rep2
## 16  EGF_30_Rep2    EGF_30      Rep2
## 17  EGF_60_Rep2    EGF_60      Rep2
## 18  EGF_90_Rep2    EGF_90      Rep2
## 19 EGF_120_Rep2   EGF_120      Rep2
## 20 EGF_150_Rep2   EGF_150      Rep2
## 21  PBS_30_Rep3    PBS_30      Rep3
## 22  PBS_60_Rep3    PBS_60      Rep3
## 23  PBS_90_Rep3    PBS_90      Rep3
## 24 PBS_120_Rep3   PBS_120      Rep3
## 25 PBS_150_Rep3   PBS_150      Rep3
## 26  EGF_30_Rep3    EGF_30      Rep3
## 27  EGF_60_Rep3    EGF_60      Rep3
## 28  EGF_90_Rep3    EGF_90      Rep3
## 29 EGF_120_Rep3   EGF_120      Rep3
## 30 EGF_150_Rep3   EGF_150      Rep3
## 31  PBS_30_Rep4    PBS_30      Rep4
## 32  PBS_60_Rep4    PBS_60      Rep4
## 33  PBS_90_Rep4    PBS_90      Rep4
## 34 PBS_120_Rep4   PBS_120      Rep4
## 35 PBS_150_Rep4   PBS_150      Rep4
## 36  EGF_30_Rep4    EGF_30      Rep4
## 37  EGF_60_Rep4    EGF_60      Rep4
## 38  EGF_90_Rep4    EGF_90      Rep4
## 39 EGF_120_Rep4   EGF_120      Rep4
## 40 EGF_150_Rep4   EGF_150      Rep4
```

Data Processing

Summarize Data We **summarize** the raw data into an object format recognizable by the *DEP* R-package.

```
# Generate a SummarizedExperiment object using an experimental design
Int_columns <- 4:43 # get Intensity column numbers
data_se <- make_se(data_unique, Int_columns, as.data.frame(experimental_design))
data_se
```

```
## class: SummarizedExperiment
## dim: 5669 40
## metadata(0):
## assays(1): ''
## rownames(5669): CH10_HUMAN CH10_HUMAN.1 ... TRUA_HUMAN RTCB_HUMAN
## rowData names(5): Gene Accession Sequence name ID
## colnames(40): PBS_30_Rep1 PBS_60_Rep1 ... EGF_120_Rep4 EGF_150_Rep4
## colData names(4): label ID condition replicate
```

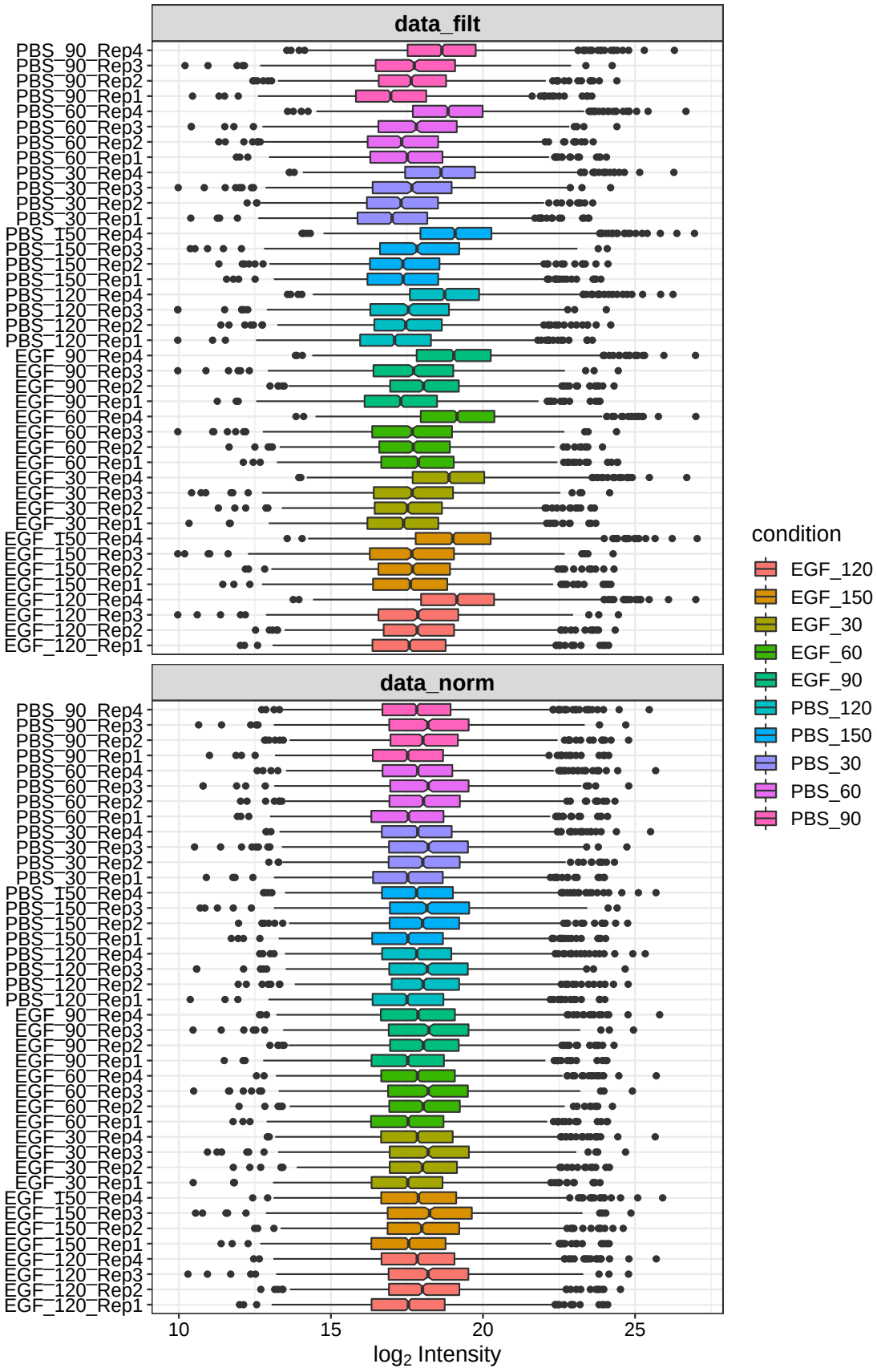
Filtering We **filter** the proteomics dataset based on missing values. The dataset is filtered for proteins that have a maximum of 2 missing values in at least one condition.

```
# Filter for proteins that are identified in 2 out of 4 replicates of at least
# one condition
data_filt <- filter_missval(data_se, thr = 2)
data_filt
```

```
## class: SummarizedExperiment
## dim: 2108 40
## metadata(0):
## assays(1): ''
## rownames(2108): 1433B_HUMAN.1 1433E_HUMAN ... ZFR_HUMAN.4 ZN622_HUMAN.2
## rowData names(5): Gene Accession Sequence name ID
## colnames(40): PBS_30_Rep1 PBS_60_Rep1 ... EGF_120_Rep4 EGF_150_Rep4
## colData names(4): label ID condition replicate
```

Normalization We perform a variance stabilizing **transformation/normalization** using the vsn-package. We then also see the differences in the data before and after **normalization**.

```
# Normalize the data
data_norm <- normalize_vsn(data_filt)

# Visualize normalization by boxplots for all samples before and after normalization
plot_normalization(data_filt, data_norm)
```


data_filt
PBS_90_Rep4
PBS_90_Rep3
PBS_90_Rep2
PBS_90_Rep1
PBS_60_Rep4
PBS_60_Rep3
PBS_60_Rep2
PBS_60_Rep1
PBS_30_Rep4
PBS_30_Rep3
PBS_30_Rep2
PBS_30_Rep1
PBS_150_Rep4
PBS_150_Rep3
PBS_150_Rep2
PBS_150_Rep1
PBS_120_Rep4
PBS_120_Rep3
PBS_120_Rep2
PBS_120_Rep1
EGF_90_Rep4
EGF_90_Rep3
EGF_90_Rep2
EGF_90_Rep1
EGF_60_Rep4
EGF_60_Rep3
EGF_60_Rep2
EGF_60_Rep1
EGF_30_Rep4
EGF_30_Rep3
EGF_30_Rep2
EGF_30_Rep1
EGF_150_Rep4
EGF_150_Rep3
EGF_150_Rep2
EGF_150_Rep1
EGF_120_Rep4
EGF_120_Rep3
EGF_120_Rep2
EGF_120_Rep1
data_norm
PBS_90_Rep4
PBS_90_Rep3
PBS_90_Rep2
PBS_90_Rep1
PBS_60_Rep4
PBS_60_Rep3
PBS_60_Rep2
PBS_60_Rep1
PBS_30_Rep4
PBS_30_Rep3
PBS_30_Rep2
PBS_30_Rep1
PBS_150_Rep4
PBS_150_Rep3
PBS_150_Rep2
PBS_150_Rep1
PBS_120_Rep4
PBS_120_Rep3
PBS_120_Rep2
PBS_120_Rep1
EGF_90_Rep4
EGF_90_Rep3
EGF_90_Rep2
EGF_90_Rep1
EGF_60_Rep4
EGF_60_Rep3
EGF_60_Rep2
EGF_60_Rep1
EGF_30_Rep4
EGF_30_Rep3
EGF_30_Rep2
EGF_30_Rep1
EGF_150_Rep4
EGF_150_Rep3
EGF_150_Rep2
EGF_150_Rep1
EGF_120_Rep4
EGF_120_Rep3
EGF_120_Rep2
EGF_120_Rep1
10
15
20
25
log2 Intensity
condition
EGF_120
EGF_150
EGF_30
EGF_60
EGF_90
PBS_120
PBS_150
PBS_30
PBS_60
PBS_90

Imputation There are still many missing values in the data. We perform **imputation** in order to impute missing values in the data-set and then visualize the effects which imputation has on the distribution of the data.

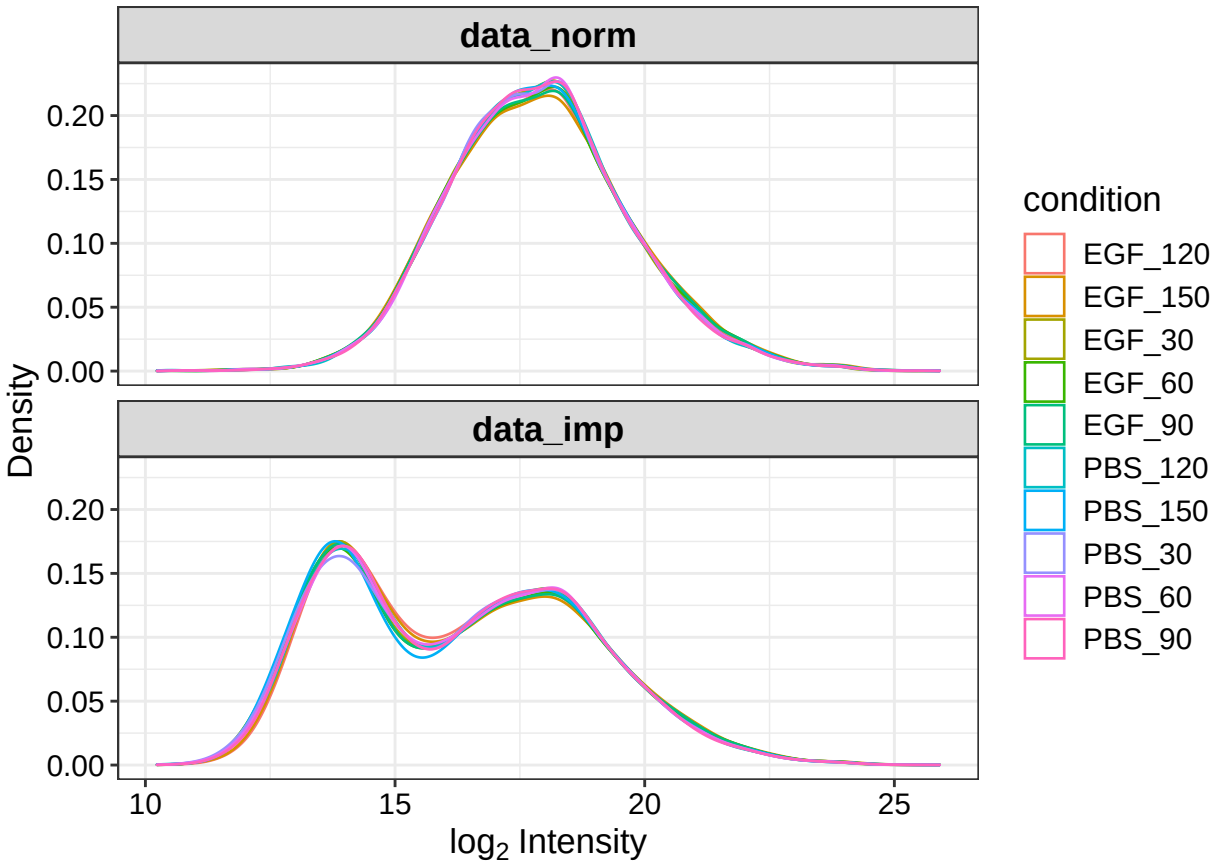
```
# Percentage of missing data
perc <- length(which(is.na(data_norm@assays@data@listData[[1]])))/
  (dim(data_norm@assays@data@listData[[1]])[1]*
     dim(data_norm@assays@data@listData[[1]])[2])
print(paste0(round(x = 100*perc, digits = 2), "%"))
```

```
FALSE [1] "38.21%"
```

```
# Impute missing data using random draws from a Gaussian distribution centered
# around a minimal value (for MNAR)
data_imp <- impute(data_norm, fun = "MinProb", q = 0.01)
```

```
FALSE [1] 0.8196504
```

```
# We can then visualize the effect of imputation on the data
plot_imputation(data_norm, data_imp)
```



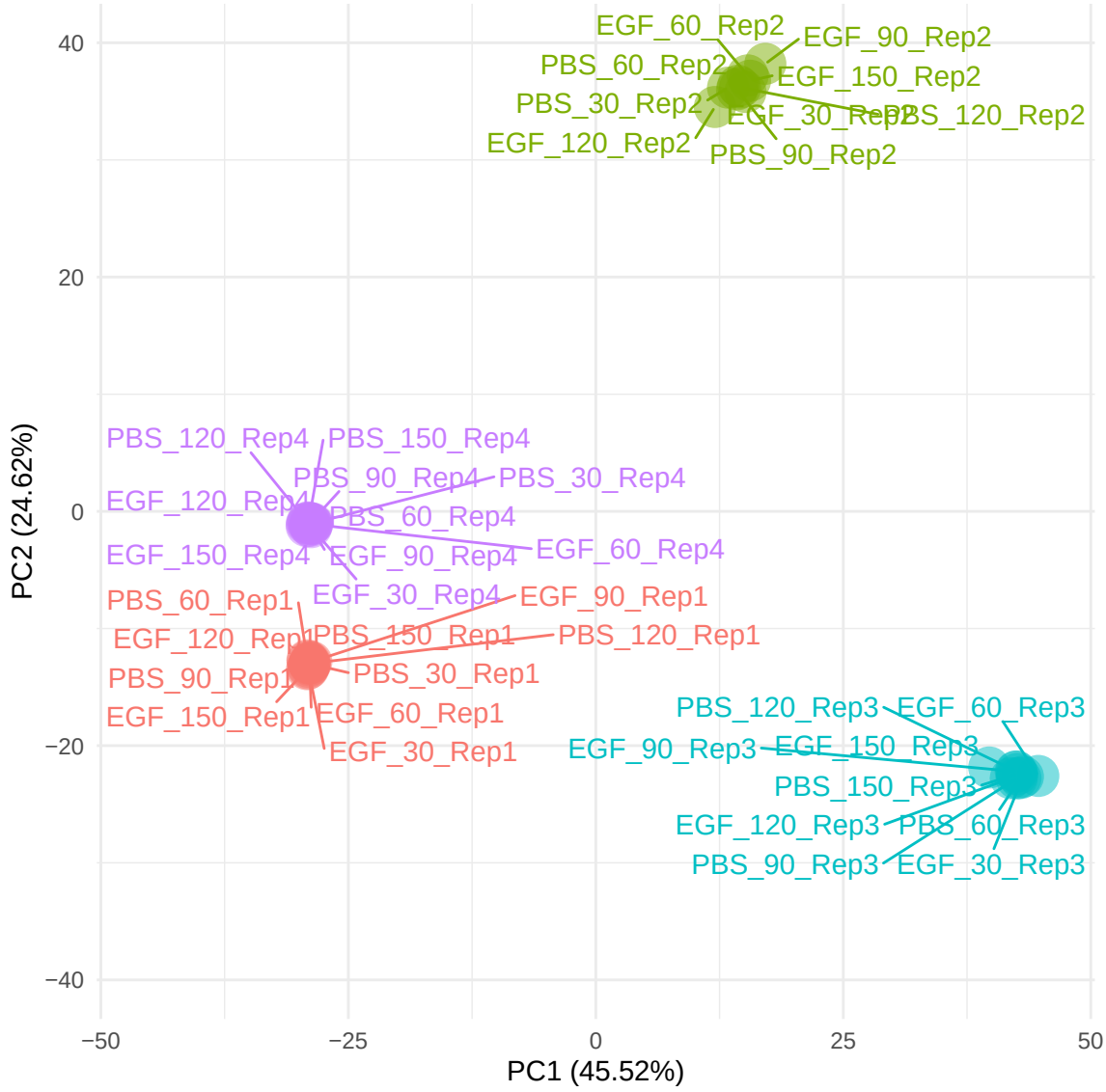
Interpretation We perform a **dimensionality reduction/PCA** analysis in order to enhance the interprability of the data by reducing it's complexity.

```
matrix_data <- data_imp@assays@data@listData[[1]]

# PCA plot based on condition
groups <- experimental_design$condition
```

```
names(groups) <- experimental_design$label

data.pca <- t(matrix_data)
data.pca <- cbind(data.pca, as.matrix(groups))
colnames(data.pca)[ncol(data.pca)] <- "Group"
data.pca <- as.data.frame(data.pca)
data.pca[, 1:(ncol(data.pca)-1)] <- lapply(data.pca[, 1:(ncol(data.pca)-1)],
                                           function(x) as.numeric(as.character(x)))
res.pca <- prcomp(data.pca[, -ncol(data.pca)], scale. = TRUE)
res.plot <-  as.data.frame(cbind(res.pca$x[, 1], res.pca$x[, 2],
                                 as.character(data.pca$Group), rownames(data.pca)))
res.plot[, 1:2] <- lapply(res.plot[, 1:2], function(x) as.numeric(as.character(x)))
res.plot[, 3:4] <- lapply(res.plot[, 3:4], function(x) as.character(x))
colnames(res.plot) <- c("pc1", "pc2", "Group", "sample")
percentages <- ((res.pca$sdev)^2 / sum(res.pca$sdev^2)*100)[1:2]
pp <- ggplot(res.plot, aes(x=pc1, y=pc2, color=Group)) +
  geom_point(size=7, alpha = 0.5) +
  scale_alpha_discrete(range=c(0.3, 1.0)) +
  theme_minimal() +
  xlab(paste0("PC1 (", round(x = percentages[1], digits = 2), "%)")) +
  ylab(paste0("PC2 (", round(x = percentages[2], digits = 2), "%)")) +
  xlim(c(-max(abs(res.pca$x[, 1])),max(abs(res.pca$x[, 1])))) +
  ylim(c(-max(abs(res.pca$x[, 2])),max(abs(res.pca$x[, 2])))) +
  theme(legend.position = "none") +
  geom_text_repel(data = res.plot, aes(label=sample), max.overlaps = 100)
plot(pp)
```

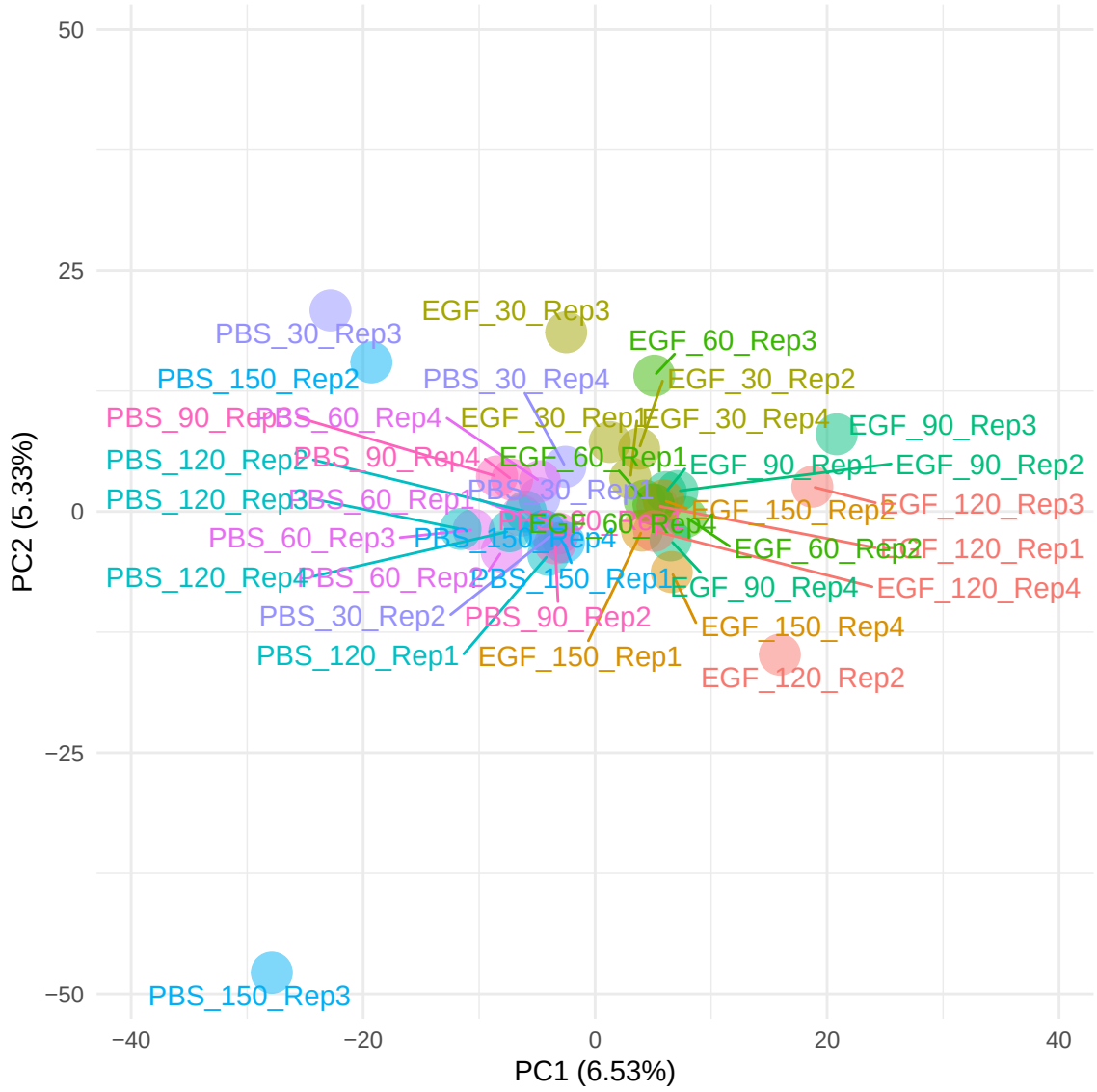

```
# PCA plot based on relpicate
groups <- experimental_design$replicate
names(groups) <- experimental_design$label

data.pca <- t(matrix_data)
data.pca <- cbind(data.pca, as.matrix(groups))
colnames(data.pca)[ncol(data.pca)] <- "Group"
data.pca <- as.data.frame(data.pca)
data.pca[, 1:(ncol(data.pca)-1)] <- lapply(data.pca[, 1:(ncol(data.pca)-1)],
                                           function(x) as.numeric(as.character(x)))
res.pca <- prcomp(data.pca[, -ncol(data.pca)], scale. = TRUE)
res.plot <-  as.data.frame(cbind(res.pca$x[, 1], res.pca$x[, 2],
                                 as.character(data.pca$Group), rownames(data.pca)))
res.plot[, 1:2] <- lapply(res.plot[, 1:2], function(x) as.numeric(as.character(x)))
res.plot[, 3:4] <- lapply(res.plot[, 3:4], function(x) as.character(x))
colnames(res.plot) <- c("pc1", "pc2", "Group", "sample")
percentages <- ((res.pca$sdev)^2 / sum(res.pca$sdev^2)*100)[1:2]
```

```
pp <- ggplot(res.plot, aes(x=pc1, y=pc2, color=Group)) +
  geom_point(size=7, alpha = 0.5) +
  scale_alpha_discrete(range=c(0.3, 1.0)) +
  theme_minimal() +
  xlab(paste0("PC1 (", round(x = percentages[1], digits = 2), "%)")) +
  ylab(paste0("PC2 (", round(x = percentages[2], digits = 2), "%)")) +
  xlim(c(-max(abs(res.pca$x[, 1])),max(abs(res.pca$x[, 1])))) +
  ylim(c(-max(abs(res.pca$x[, 2])),max(abs(res.pca$x[, 2])))) +
  theme(legend.position = "none") +
  geom_text_repel(data = res.plot, aes(label=sample), max.overlaps = 100)
plot(pp)
```



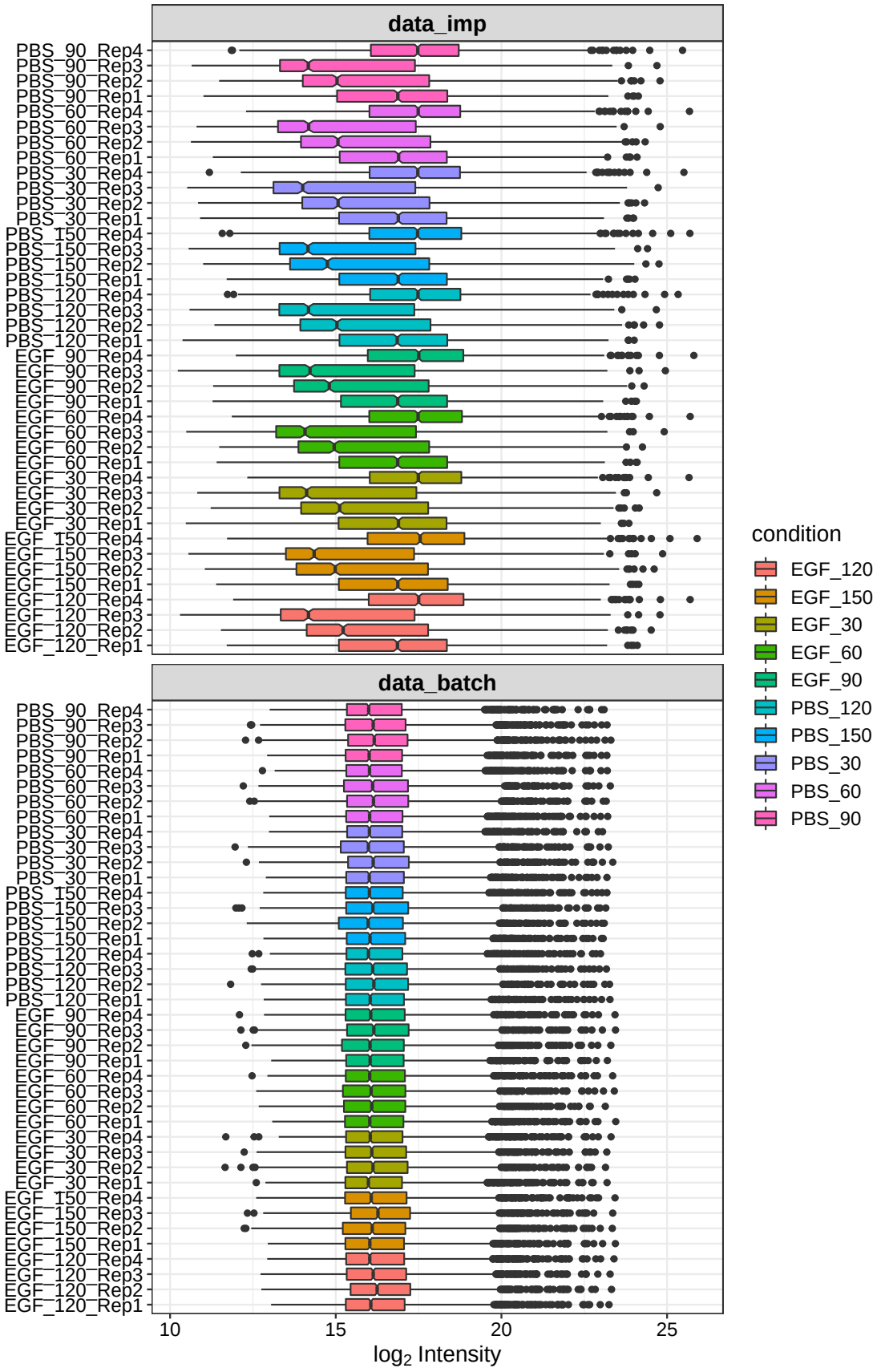
Batch Effect Removal There are **Batch Effects** across replicates which we need to remove. We remove the batch effects through the *removeBatchEffect()* function of the *limma* R-package. Next we verify the removal of batch effects with a **PCA** plot of the transformed data.

```r
# Remove Batch effects
batch_rem <- removeBatchEffect(x = matrix_data, batch = experimental_design$replicate)

# PCA plot based on condition after removing the batch effects
groups <- experimental_design$condition
names(groups) <- experimental_design$label

data.pca <- t(batch_rem)
data.pca <- cbind(data.pca, as.matrix(groups))
colnames(data.pca)[ncol(data.pca)] <- "Group"
data.pca <- as.data.frame(data.pca)
data.pca[, 1:(ncol(data.pca)-1)] <- lapply(data.pca[, 1:(ncol(data.pca)-1)],
                                           function(x) as.numeric(as.character(x)))
res.pca <- prcomp(data.pca[, -ncol(data.pca)], scale. = TRUE)
res.plot <-  as.data.frame(cbind(res.pca$x[, 1], res.pca$x[, 2],
                                 as.character(data.pca$Group), rownames(data.pca)))
res.plot[, 1:2] <- lapply(res.plot[, 1:2], function(x) as.numeric(as.character(x)))
res.plot[, 3:4] <- lapply(res.plot[, 3:4], function(x) as.character(x))
colnames(res.plot) <- c("pc1", "pc2", "Group", "sample")
percentages <- ((res.pca$sdev)^2 / sum(res.pca$sdev^2)*100)[1:2]
pp <- ggplot(res.plot, aes(x=pc1, y=pc2, color=Group)) +
  geom_point(size=7, alpha = 0.5) +
  scale_alpha_discrete(range=c(0.3, 1.0)) +
  #geom_path(arrow=arrow()) +
  theme_minimal() +
  xlab(paste0("PC1 (", round(x = percentages[1], digits = 2), "%)")) +
  ylab(paste0("PC2 (", round(x = percentages[2], digits = 2), "%)")) +
  xlim(c(-max(abs(res.pca$x[, 1])),max(abs(res.pca$x[, 1])))) +
  ylim(c(-max(abs(res.pca$x[, 2])),max(abs(res.pca$x[, 2])))) +
  theme(legend.position = "none") +
  geom_text_repel(data = res.plot, aes(label=sample), max.overlaps = 100)
plot(pp)
```



We create the *DEP Object* after removing the batch effects and **comparing the distribution of data** before and after batch effect removal.

```
# Create the DEP object after removing the batch effects
data_batch <- data_imp
data_batch@assays@data@listData[[1]] <- batch_rem

# Plot intensity distributions before and after batch correction
plot_normalization(data_imp, data_batch)
```


data_imp
PBS_90_Rep4
PBS_90_Rep3
PBS_90_Rep2
PBS_90_Rep1
PBS_60_Rep4
PBS_60_Rep3
PBS_60_Rep2
PBS_60_Rep1
PBS_30_Rep4
PBS_30_Rep3
PBS_30_Rep2
PBS_30_Rep1
PBS_150_Rep4
PBS_150_Rep3
PBS_150_Rep2
PBS_150_Rep1
PBS_120_Rep4
PBS_120_Rep3
PBS_120_Rep2
PBS_120_Rep1
EGF_90_Rep4
EGF_90_Rep3
EGF_90_Rep2
EGF_90_Rep1
EGF_60_Rep4
EGF_60_Rep3
EGF_60_Rep2
EGF_60_Rep1
EGF_30_Rep4
EGF_30_Rep3
EGF_30_Rep2
EGF_30_Rep1
EGF_150_Rep4
EGF_150_Rep3
EGF_150_Rep2
EGF_150_Rep1
EGF_120_Rep4
EGF_120_Rep3
EGF_120_Rep2
EGF_120_Rep1
data_batch
PBS_90_Rep4
PBS_90_Rep3
PBS_90_Rep2
PBS_90_Rep1
PBS_60_Rep4
PBS_60_Rep3
PBS_60_Rep2
PBS_60_Rep1
PBS_30_Rep4
PBS_30_Rep3
PBS_30_Rep2
PBS_30_Rep1
PBS_150_Rep4
PBS_150_Rep3
PBS_150_Rep2
PBS_150_Rep1
PBS_120_Rep4
PBS_120_Rep3
PBS_120_Rep2
PBS_120_Rep1
EGF_90_Rep4
EGF_90_Rep3
EGF_90_Rep2
EGF_90_Rep1
EGF_60_Rep4
EGF_60_Rep3
EGF_60_Rep2
EGF_60_Rep1
EGF_30_Rep4
EGF_30_Rep3
EGF_30_Rep2
EGF_30_Rep1
EGF_150_Rep4
EGF_150_Rep3
EGF_150_Rep2
EGF_150_Rep1
EGF_120_Rep4
EGF_120_Rep3
EGF_120_Rep2
EGF_120_Rep1
10
15
20
25
log2 Intensity
condition
EGF_120
EGF_150
EGF_30
EGF_60
EGF_90
PBS_120
PBS_150
PBS_30
PBS_60
PBS_90

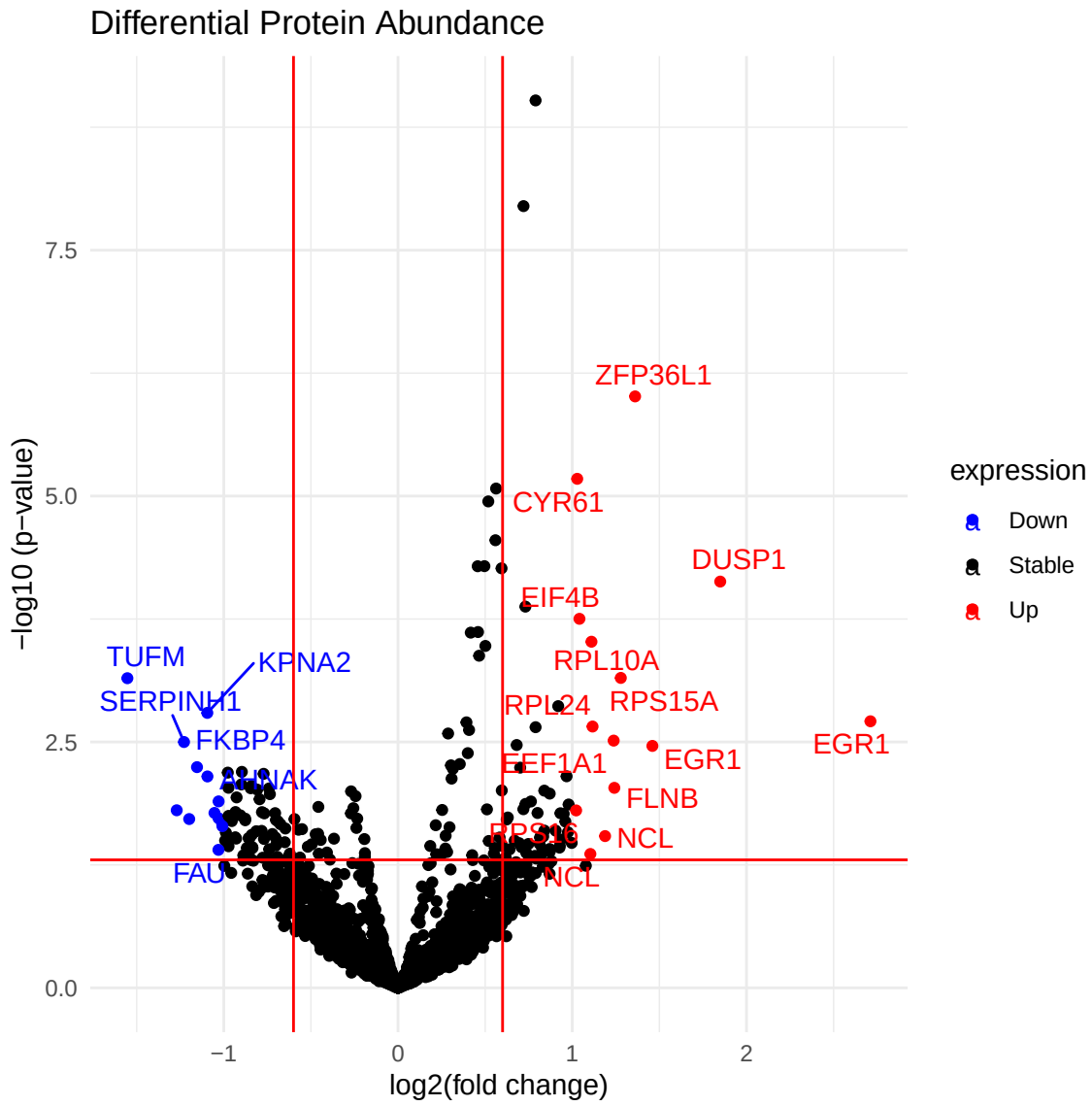
Differential Analysis

We perform **Differential Analysis** in order to identify the differentially abundant Proteins.

```
# Test manually defined comparisons for time-point 60.
data_diff_manual <- test_diff(data_batch, type = "manual",
                              test = c("EGF_60_vs_PBS_60"))

ttop <- as.data.frame(data_diff_manual@elementMetadata)
ttop <- ttop[, c(1:4, 10:12)]
head(ttop)
```

```
FALSE            name  Gene   Accession                      Sequence
FALSE 1 1433B_HUMAN.1 YWHAB 1433B_HUMAN TAFDEAIAELDTLNEESYKDSTLIMQLLR
FALSE 2   1433E_HUMAN YWHAE 1433E_HUMAN                   EAAENSLVAYK
FALSE 3 1433E_HUMAN.1 YWHAE 1433E_HUMAN                 VAGMDVELTVEER
FALSE 4   1433G_HUMAN YWHAG 1433G_HUMAN                    ATVVESSEK
FALSE 5 1433G_HUMAN.1 YWHAG 1433G_HUMAN                NVTELNEPLSNEER
FALSE 6 1433G_HUMAN.2 YWHAG 1433G_HUMAN                   YLAEVATGEK
FALSE   EGF_60_vs_PBS_60_diff EGF_60_vs_PBS_60_p.adj EGF_60_vs_PBS_60_p.val
FALSE 1           -0.28245197              0.9273398              0.4943292
FALSE 2           -0.49856379              0.8285225              0.1716924
FALSE 3            0.09324087              0.9482865              0.7358296
FALSE 4            0.05652220              0.9563284              0.8930352
FALSE 5           -0.24590727              0.9263987              0.4868471
FALSE 6           -0.06624042              0.9519506              0.8007577
```

```
ttop$expression = ifelse(ttop$EGF_60_vs_PBS_60_p.val < 0.05 &
                           abs(ttop$EGF_60_vs_PBS_60_diff) >= 1,
                       ifelse(ttop$EGF_60_vs_PBS_60_diff> 1 ,'Up','Down'),
                       'Stable')

ttop$label <- NA
ttop$label[which(ttop$expression!="Stable")] <-
  ttop$Gene[which(ttop$expression!="Stable")]

p <- ggplot(data=ttop, aes(x=EGF_60_vs_PBS_60_diff,
                           y=-log10(EGF_60_vs_PBS_60_p.val),
                           col=expression, label=label)) +
  geom_point() +
  theme_minimal() +
  geom_text_repel() +
  scale_color_manual(values=c("blue", "black", "red")) +
  geom_vline(xintercept=c(-0.6, 0.6), col="red") +
  geom_hline(yintercept=-log10(0.05), col="red") +
  labs(x="log2(fold change)",
       y="-log10 (p-value)",
       title="Differential Protein Abundance")
p
```


Differential Protein Abundance
−log10 (p−value)
7.5
5.0
2.5
0.0
−1
0
1
2
log2(fold change)
ZFP36L1
CYR61
DUSP1
EIF4B
RPL10A
RPS15A
RPL24
EGR1
EGR1
EEF1A1
FLNB
NCL
NCL
TUFM
KPNA2
SERPINH1
FKBP4
FAU
expression
Down
Stable
Up