Analytical pipeline for RIC and WCL proteomic analysis

Marko Noerenberg; marko.noerenberg@bioch.ox.ac.uk;

05 May 2020

Clear workspace

```
# Clear plots
if(!is.null(dev.list())) dev.off()
```

```
## null device
##           1
```

```
# Clear console
cat("\014")
```

```
# Clean workspace
rm(list=ls())
```

Load libraries and datasets

```
#install.package("ggrepel")
library(ggrepel)
```

```
## Warning: package 'ggrepel' was built under R version 3.6.3
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

```
library(VennDiagram)
```

```
## Loading required package: grid
```

```
## Loading required package: futile.logger
```

```
library(Biobase)
```

```
## Loading required package: BiocGenerics
```

```
## Loading required package: parallel
```

```
##
## Attaching package: 'BiocGenerics'
```

```
## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB
```

```
## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs
```

```
## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which, which.max, which.min
```

```
## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.
```

```
library(limma)
```

```
## Warning: package 'limma' was built under R version 3.6.2
```

```
##
## Attaching package: 'limma'
```

```
## The following object is masked from 'package:BiocGenerics':
##
##     plotMA
```

```
library(ggplot2)
library(XML)
```

```
## Warning: package 'XML' was built under R version 3.6.2
```

```
library(RBDmap)
# download via "http://www.hentze.embl.de/public/RBDmap/"
library(hwriter)
library(RColorBrewer)
library(Biostrings)
```

```
## Loading required package: S4Vectors
```

```
## Warning: package 'S4Vectors' was built under R version 3.6.2
```

```
## Loading required package: stats4
```

```
##
## Attaching package: 'S4Vectors'
```

```
## The following object is masked from 'package:base':
##
##     expand.grid
```

```
## Loading required package: IRanges
```

```
## Warning: package 'IRanges' was built under R version 3.6.2
```

```
##
## Attaching package: 'IRanges'
```

```
## The following object is masked from 'package:grDevices':
##
##     windows
```

```
## Loading required package: XVector
```

```
##
## Attaching package: 'Biostrings'
```

```
## The following object is masked from 'package:base':
##
##     strsplit
```

```
library(gridSVG)
```

```
##
## Attaching package: 'gridSVG'
```

```
## The following objects are masked from 'package:Biostrings':
##
##     mask, pattern
```

```
## The following object is masked from 'package:grDevices':
##
##     dev.off
```

```
library(pheatmap)
library(hwriter)
```

```
# Files are part of RBDmap, see link above
load(file.path('inputdata','enigmRBP.rda'))
load(file.path('inputdata','ProtFeatures.rda'))
summary(ProtFeatures)
```

```
##                Length Class       Mode
## names          20997  -none-      character
## ProtSeq        20997  AAStringSet S4
## GeneName       20997  -none-      character
## Symbol         20997  -none-      character
## Caution        20997  -none-      logical
## fracLowComplex 20997  -none-      numeric
## pI             20997  -none-      numeric
## h              20997  -none-      numeric
## fracDisorder   20997  -none-      numeric
```

```
load(file.path('inputdata','Index.rda'))
load(file.path('inputdata','ENSG2category.rda'))
load(file.path('inputdata','ENSGannotation.rda'))
source(file.path('R','doGSEA.R'))
source(file.path('R','hwriteSidebar.R'))
source(file.path('R','gridSvgLinePlot.R'))

#Use the following file to define any additional proteins to be added to the mapping e.g. viral protein
#Formatting should be protein sequences in fasta format.

SV_seq=readAAStringSet(file.path('inputdata','SV_proteins.txt'))
head(SV_seq)
```

```
##   A AAStringSet instance of length 6
##     width seq                                               names
## [1]   540 MEKPVVNVDVDPQSPFVVQLQKS...DKGIEAAAEVVCEVEGLQADIGA SV_wt_nsP1
## [2]   807 ALVETPRGHVRIIPQANDRMIGQ...FTPHHLNCVISSVYEGTRDGVGA SV_wt_nsP2
## [3]   549 APSYRTKRENIADCQEEAVVNAA...SRSAVSFPLRKQRRRRRSRRTEY SV_wt_nsP3
## [4]   613 LTGVGGYIFSTDTGPGHLQKKSV...LRTFAQSKRAFQAIRGEIKHLYG SV_wt_UGAnsP4
## [5]   264 MNRGFFNMLGRRPFPAPTAMWRP...LSVVTWNSKGKTIKTTPEGTEEW SV_wt_Capsid
## [6]    64 SAAPLVTAMCLLGNVSFPCDRPP...NHEAYDTLLNAILRCGSSGRSKR SV_wt_E3
```

Read peptide sequences and intensities from RIC and WCL

This loads the peptide output files from Maxquant for both the RIC ("_ric“) and Input (”_who")

```
input_raw_who= read.table(
  file.path("inputdata",'peptides_who.txt'),
  sep="\t", comment.char="",quote="",
  header=TRUE, stringsAsFactors=FALSE)

input_raw_ric= read.table(file.path("inputdata",'peptides_ric.txt'), sep="\t", comment.char="",quote=""
  header=TRUE, stringsAsFactors=FALSE)
```

Remove contaminations

```
input_raw_who=input_raw_who[input_raw_who$Potential.contaminant==''
  &input_raw_who$Reverse=='',]
input_raw_ric=input_raw_ric[input_raw_ric$Potential.contaminant=='' &input_raw_ric$Reverse=='',]
```

Specify sample names and remove unused columns from peptide table.

```
sample_names=c('hour18','hour4','mock')

input_raw_who =
  input_raw_who[,c(grep("Sequence",colnames(input_raw_who)),
                   grep("Intensity.",colnames(input_raw_who)))]

input_raw_who=input_raw_who[,c(1,6:8, 10:12, 14:16)]
input_raw_who[,c(2:10)][input_raw_who[,c(2:10)]<1]=NA
input_raw_who[,c(2:10)]=log2(input_raw_who[,c(2:10)])
input_raw_who=input_raw_who[,c(1,2,4,3,6,5,7,10,9,8)]
# Critical: At this point ensure that the sample order as deposited in input_raw_who from the maxquant
#Use View(input_raw_who) before and after the next command to ensure this is correctly translated
names(input_raw_who)=
  c('sequences',paste(sample_names,rep(1:3,each=3),sep='_'))



sample_names=c('hour18','hour4','mock')
input_raw_ric =
  input_raw_ric[,c(grep("Sequence",colnames(input_raw_ric)),
                   grep("Intensity.",colnames(input_raw_ric)))]
input_raw_ric=input_raw_ric[,c(1,6:8, 10:12, 14:16)]
input_raw_ric[,c(2:10)][input_raw_ric[,c(2:10)]<1]=NA
input_raw_ric[,c(2:10)]=log2(input_raw_ric[,c(2:10)])
input_raw_ric=input_raw_ric[,c(1,4,3,2,5,7,6,9,8,10)]
# Critical: At this point ensure that the sample order as deposited in input_raw_who from the maxquant
#Use View(input_raw_who) before and after the next command to ensure this is correctly translated
names(input_raw_ric)=
  c('sequences',paste(sample_names,rep(1:3,each=3),sep='_'))
```

Map the sequences to the protein sequence database

This provides a function to map protein IDs to Ensembl gene IDs

```
MapProt2Ensg=function(protein_list){
  ###MapProt2Ensg maps the peptide-mapping proteins to genes
  ###and return a list of mapped genes
  lapply(protein_list,function(x)
    { y=unname(ProtFeatures$GeneName[x])
  unique(y[!is.na(y)])})
}
```

Extract single matched peptides

(Only considering peptides than can be mapped to a single gene).

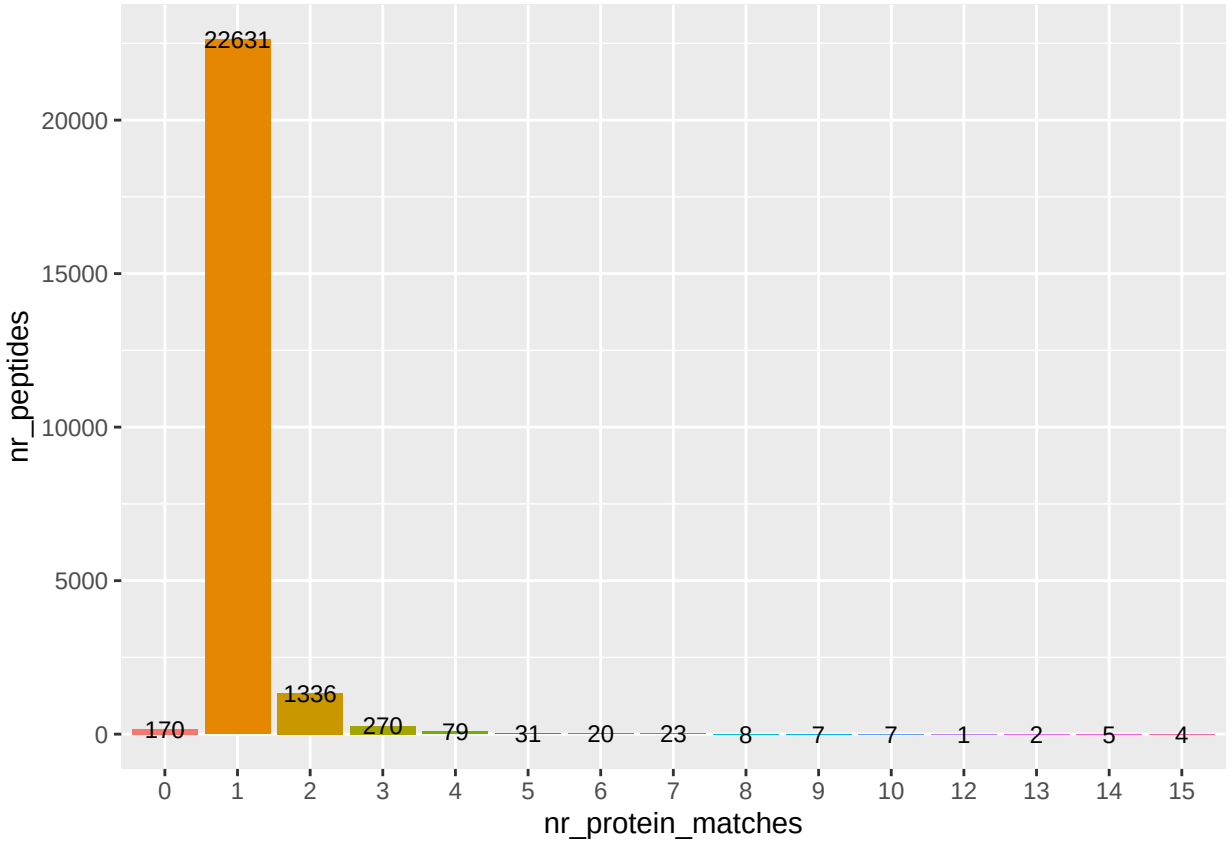
The name of additionally added proteins e.g. viral proteins is kept in the conversion from protein id to ENSEMBL gene id.

```
ProtIDs_who=mapPeptides(PeptideSet=input_raw_who$sequences,
                    c(ProtFeatures$ProtSeq,SV_seq),
                         verbose=FALSE)
ENSGid_who=MapProt2Ensg(ProtIDs_who)
ENSGid_who[grep('SV_', ProtIDs_who)]=
    ProtIDs_who[grep('SV_', ProtIDs_who)]
ENSGid_who[grep('SV_', ProtIDs_who)]=lapply(ENSGid_who[grep('SV_', ProtIDs_who)], function(x) x[1])

ProtIDs_ric=mapPeptides(PeptideSet=input_raw_ric$sequences,
                    c(ProtFeatures$ProtSeq,SV_seq),
                         verbose=FALSE)
ENSGid_ric=MapProt2Ensg(ProtIDs_ric)
ENSGid_ric[grep('SV_', ProtIDs_ric)]=ProtIDs_ric[grep('SV_', ProtIDs_ric)]
ENSGid_ric[grep('SV_', ProtIDs_ric)]=lapply(ENSGid_ric[grep('SV_', ProtIDs_ric)], function(x) x[1])
```

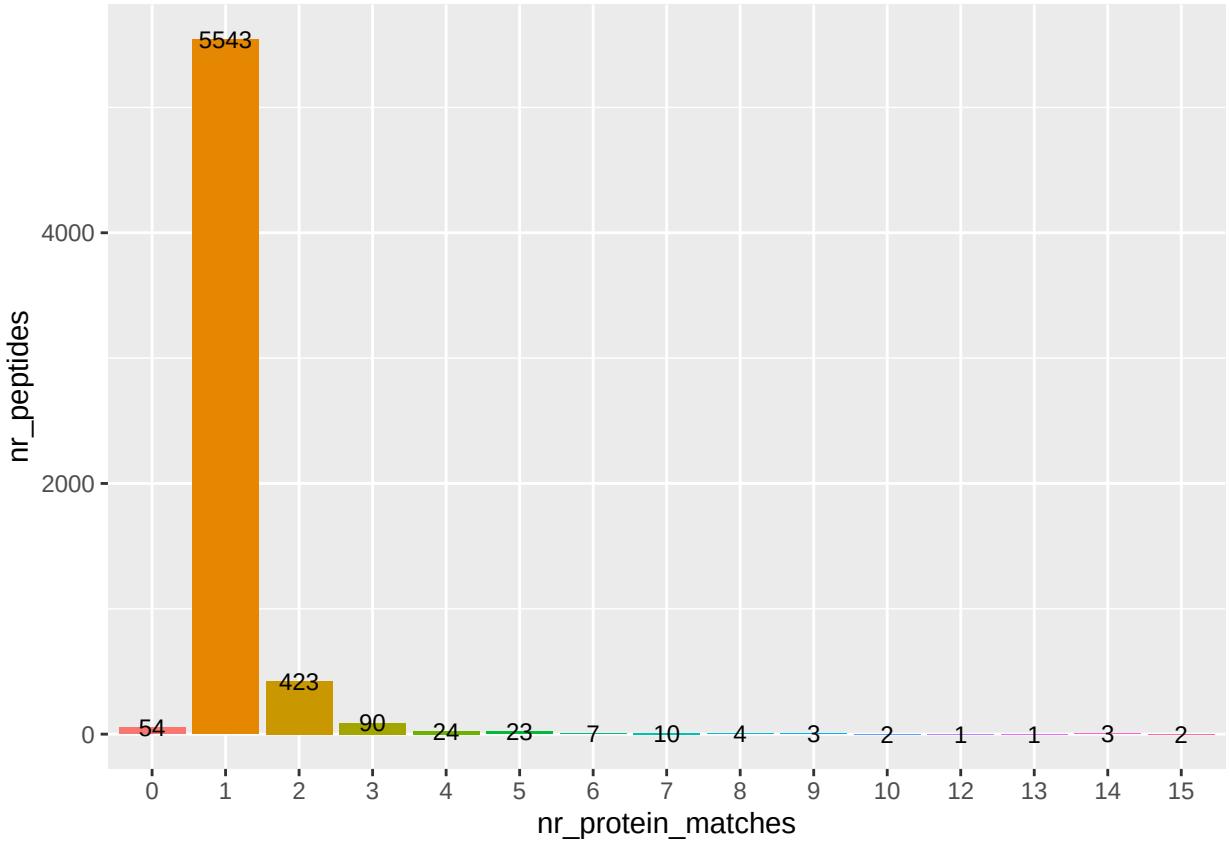
Mapping summary

The number of protein(s) to which peptides map to is represented on the x-axis. The number of peptides for this grouping is presented on the y-axis. Any peptide not uniquely mapping is discarded from the analysis.

```
df=as.data.frame(table(listLen(ENSGid_who)))
colnames(df)=c('nr_protein_matches','nr_peptides')
p = ggplot(df, aes(x=nr_protein_matches,y=nr_peptides,
                   fill=factor(nr_protein_matches)))
p = p + geom_bar(stat='identity')+theme(legend.position = "none") +
  geom_text(aes(label = nr_peptides), size = 3)

print(p)
```




```
df=as.data.frame(table(listLen(ENSGid_ric)))
colnames(df)=c('nr_protein_matches','nr_peptides')
p = ggplot(df, aes(x=nr_protein_matches,y=nr_peptides,
                   fill=factor(nr_protein_matches)))
p = p + geom_bar(stat='identity')+theme(legend.position = "none") +
  geom_text(aes(label = nr_peptides), size = 3)

print(p)
```



Aggregate mean intensity values for each ENSEMBL gene ID

The mean intensity values of peptides mapped to the same gene are calculated to represent the intensity of that protein.

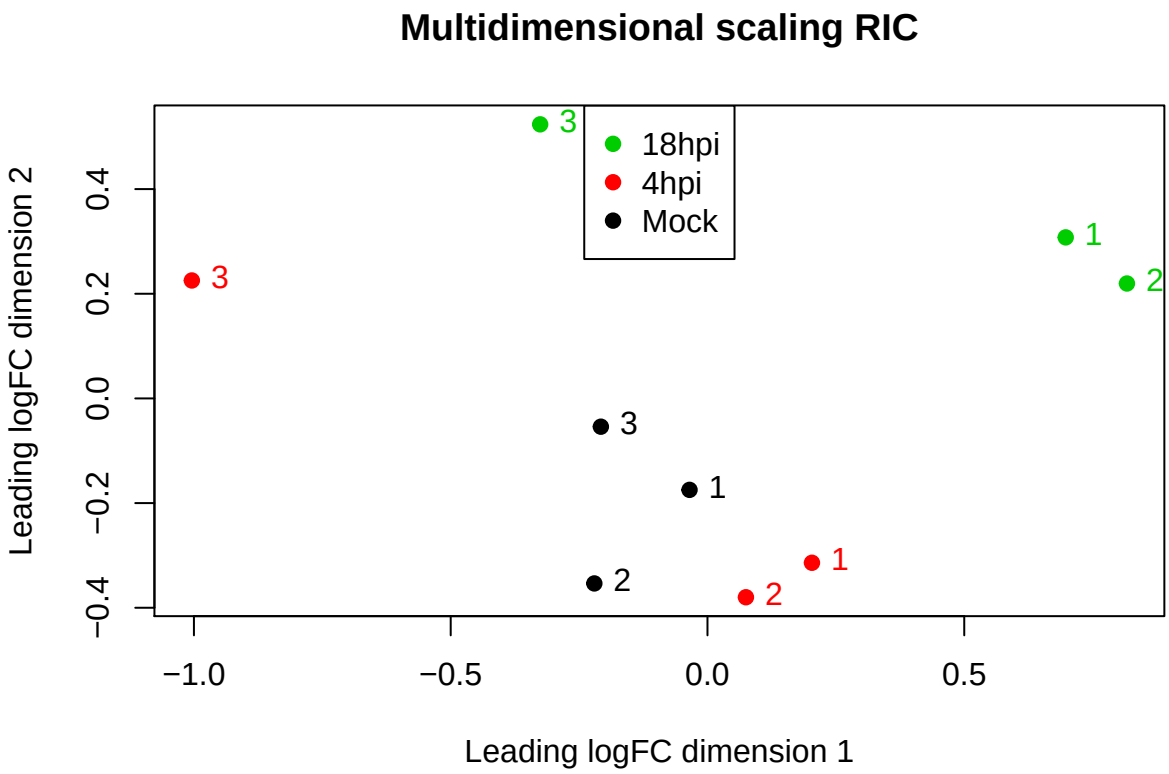
```
###### Input

ENSGidUnique_who=ENSGid_who
ENSGidUnique_who[listLen(ENSGidUnique_who)!= 1]=NA
ENSGidUnique_who=unlist(ENSGidUnique_who)
input_raw_who=cbind(input_raw_who,ENSGid=ENSGidUnique_who, stringsAsFactors=FALSE)


proteins_who=aggregate(input_raw_who[,2:10],
   by=list(input_raw_who$ENSGid), mean, na.rm=TRUE)
names(proteins_who)[1]='ENSGid'


proteins_who$symbol=unname(unlist(
  sapply(proteins_who$ENSGid,
         function(x) {if
           (any(which(ProtFeatures$GeneName == x)))
           ProtFeatures$Symbol[which(ProtFeatures$GeneName == x)]
             else NA})))

proteins_who$symbol[grep('SV_',proteins_who$ENSGid)]=
```

```
  as.character(proteins_who$ENSGid[grep('SV_',proteins_who$ENSGid)])

proteins_who$Know_RBP=ifelse(
  proteins_who$ENSGid %in% enigmRBP$Ensembl.gene.ID,
  'known_RBP', 'no')
proteins_who=proteins_who[,c(1,11,12,2:10)]
proteins_who=data.frame(proteins_who,stringsAsFactors = FALSE )
#row.names(proteins_who)=NULL

###### RIC


ENSGidUnique_ric=ENSGid_ric
ENSGidUnique_ric[listLen(ENSGidUnique_ric)!= 1]=NA
ENSGidUnique_ric=unlist(ENSGidUnique_ric)
input_raw_ric=cbind(input_raw_ric,ENSGid=ENSGidUnique_ric, stringsAsFactors=FALSE)


proteins_ric=aggregate(input_raw_ric[,2:10],
   by=list(input_raw_ric$ENSGid), mean, na.rm=TRUE)
names(proteins_ric)[1]='ENSGid'


proteins_ric$symbol=unname(unlist(
  sapply(proteins_ric$ENSGid,
         function(x) {if
           (any(which(ProtFeatures$GeneName == x)))
           ProtFeatures$Symbol[which(ProtFeatures$GeneName == x)]
             else NA})))

proteins_ric$symbol[grep('SV_',proteins_ric$ENSGid)]=
  as.character(proteins_ric$ENSGid[grep('SV_',proteins_ric$ENSGid)])

proteins_ric$Know_RBP=ifelse(
  proteins_ric$ENSGid %in% enigmRBP$Ensembl.gene.ID,
  'known_RBP', 'no')
proteins_ric=proteins_ric[,c(1,11,12,2:10)]
proteins_ric=data.frame(proteins_ric,stringsAsFactors = FALSE )
#row.names(proteins_who)=NULL
```

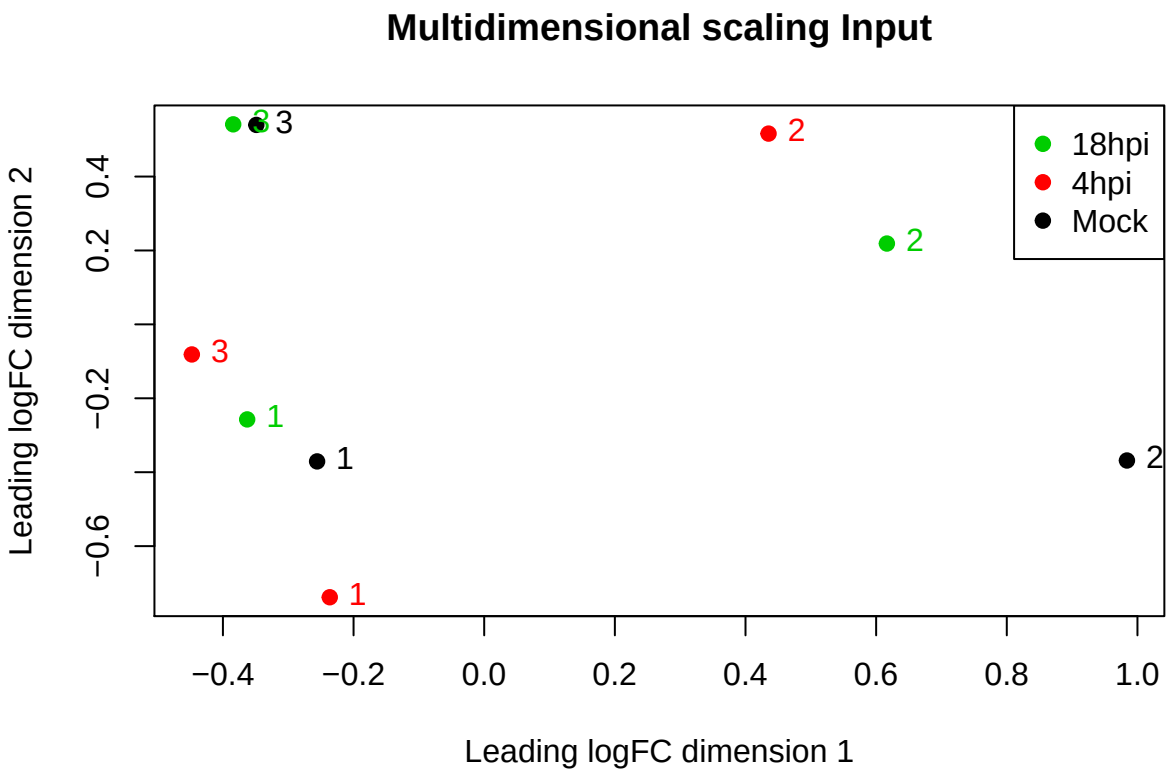
Multidimensional scaling to identify any potential problems

This approach visualises in an unbiased manner, how similar the samples in the dataset are to each other. Clustering of samples is expected to happen according to the experimental conditions. If samples cluster according to other variables e.g. biological replicate, this can highlight potential biases in the dataset for example differences in MS depth between replicates.

```
proteins_prebatch_ric=proteins_ric

MDS_ric=plotMDS(proteins_prebatch_ric[c(4:12)], labels = NULL, pch= c(19), col= c(3:1), cex = 1,   gene
text(x=MDS_ric$x, y = MDS_ric$y, labels=c("1","1","1","2","2","2","3","3","3"), pos= 4, col=c(3:1))
legend("top", legend=c("18hpi","4hpi","Mock"), col=3:1, pch=19)
```




```
proteins_prebatch_who=proteins_who

MDS_who=plotMDS(proteins_prebatch_who[c(4:12)], labels = NULL, pch= c(19), col= c(3:1), cex = 1,   gene
text(x=MDS_who$x, y = MDS_who$y, labels=c("1","1","1","2","2","2","3","3","3"), pos= 4, col=c(3:1))
legend("topright", legend=c("18hpi","4hpi","Mock"), col=3:1, pch=19)
```



Perform batch correction (only when necessary).

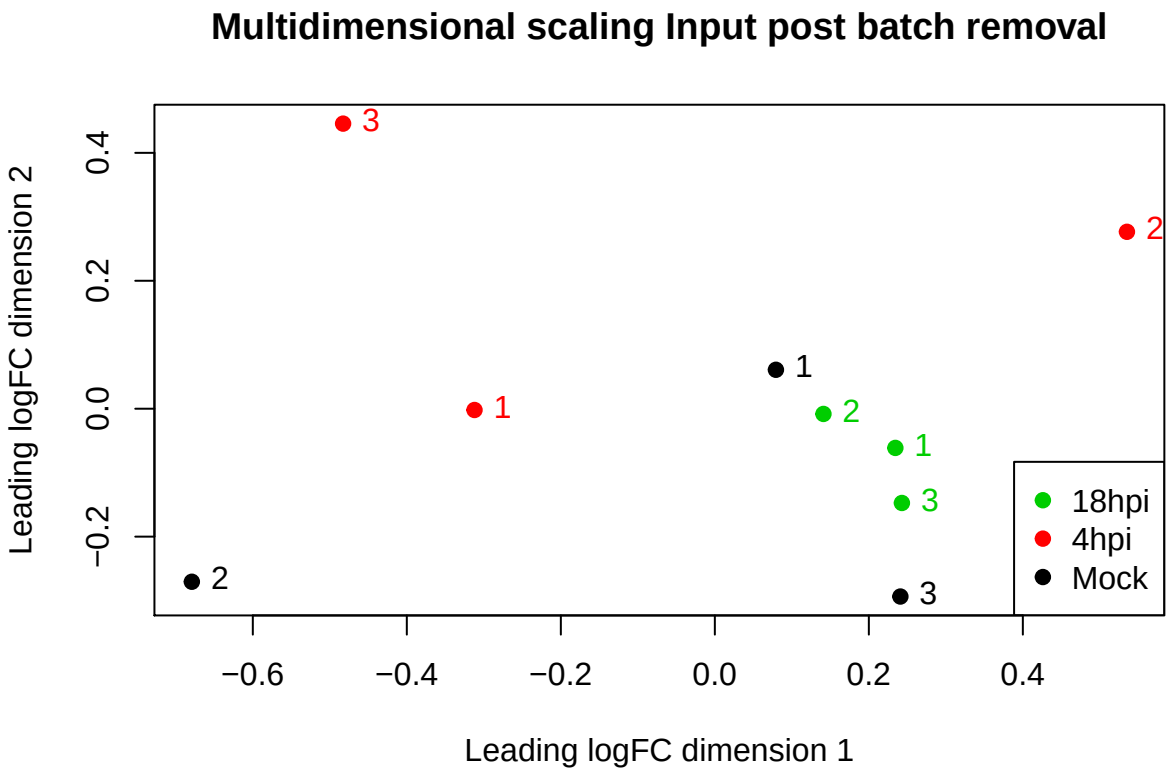
For statistical analysis, batch correction should be implemented in an appropriate design matrix.

For plotting of intensity values for visualisation, batch effects can be removed using removeBatchEffect.

```
## Potential batch effect due to SILAC labelling - Light, Medium and Heavy labels are represented with

batch <- c("C","B","A","B","A","C","A","C","B")

## Potential batch effect due to experimental repeat/MS run

batch2 <- c("A","A","A","B","B","B","C","C","C")

# Multidimensional scaling of RIC shows clustering based on treatment but not based in experimental bat
# Hence, batch correction is here not required.


# Multidimensional scaling of Input shows clustering based on experimental batch, hence batch correctio
# However, we do not observe clustering of treatments after correction, suggesting that samples are ver
# Lack of changes between treatments is later confirmed by scatter plot of fold changes and statistical

proteins_who_batch<-proteins_prebatch_who
proteins_who_batch[,4:12]<-removeBatchEffect(proteins_prebatch_who[,4:12], batch2)
```

```
## Warning: Partial NA coefficients for 1040 probe(s)
```

```
MDS_who=plotMDS(proteins_who_batch[c(4:12)], labels = NULL, pch= c(19), col= c(3:1), cex = 1,   gene.sel
text(x=MDS_who$x, y = MDS_who$y, labels=c("1","1","1","2","2","2","3","3","3"), pos= 4, col=c(3:1))
legend("bottomright", legend=c("18hpi","4hpi","Mock"), col=3:1, pch=19)
```



Estimate relative RNA binding activity

Substract WCL from RIC values (already log2 transformed) to estimate how strong the change in RIC is driven by changes in abundance (RIC/WCL)

```
proteins_int=merge(proteins_who_batch, proteins_ric,
                   by=c('ENSGid', 'symbol','Know_RBP'))
proteins_int[,4:12]=proteins_int[,13:21]-proteins_int[,4:12]
proteins_int=proteins_int[,1:12]
```

Save the proteins lists

```
saveRDS(proteins_ric, 'Proteins_ric.rds')
saveRDS(proteins_who, 'Proteins_who.rds')
saveRDS(proteins_who_batch, 'Proteins_who_batch.rds')
saveRDS(proteins_int, 'Proteins_int.rds')
```

Differential t-test at protein level

Moderated t-test for set enrichment

The aggregated mean intensity values for each protein is tested for enrichment between differnt states of infection. This is done using a paired moderated t-test as implemented in the Bioconductor-package limma.

For more complex scenarios, use an appropriate design matrix.

p-values are corrected for multiple testing using the Benjamini-Hochberg approach.

```
dirTtest='t_test'
dir.create(file.path(dirTtest), showWarnings = FALSE)

Intensities = as.matrix(proteins_who[,grepl("hour",colnames(proteins_who)) | grepl("mock",colnames(prote
cond = sapply(strsplit(colnames(Intensities), split="_"),
              function(x) { x[1] })

sample_combi=combn(sample_names,2,simplify=TRUE)
colnames(sample_combi) =
  apply(sample_combi,2,
        function(x) {
          paste('diff',paste(x,collapse='_'),sep='_')
          })

diff_table_who = diff_intensities_who = list()
for (s in colnames(sample_combi)) {
  sample1 = Intensities[,cond == sample_combi[1,s]]
  sample2 = Intensities[,cond == sample_combi[2,s]]
  X =sample1-sample2

  # This segment performs a median correction
    for (j in 1:length(sample_names)) {
    X[,j] = X[,j] - median(X[,j],na.rm = TRUE)}

  diff_intensities_who[[s]] = X
  fit=eBayes(lmFit(X))
  fit$p.adj=p.adjust(fit$p.value,method="BH")

    fit$p.adj[is.na(fit$p.adj)]=1

  dt = data.frame(ENSGid = proteins_who$ENSGid,
                  symbol = proteins_who$symbol,
                  Know_RBP = proteins_who$Know_RBP,
                  log2FC = fit$coefficients[,1],
                  p.value = fit$p.value[,1],
                  p.adj = fit$p.adj,
                  stringsAsFactors=FALSE)
  dt$sig=''
  dt$sig[dt$p.adj<0.1]='*'
  dt$sig[dt$p.adj<0.01]='**'
  dt=dt[,c(1,2,3,4,5,6,7)]
  diff_table_who[[s]] = dt
}
```

```
Intensities = as.matrix(proteins_ric[,grepl("hour",colnames(proteins_ric)) |
                                     grepl("mock",colnames(proteins_ric))])
cond = sapply(strsplit(colnames(Intensities), split="_"),
              function(x) { x[1] })

sample_combi=combn(sample_names,2,simplify=TRUE)
colnames(sample_combi) =
  apply(sample_combi,2,
        function(x) {
          paste('diff',paste(x,collapse='_'),sep='_')
          })

diff_table_ric = diff_intensities_ric = list()
for (s in colnames(sample_combi)) {
  sample1 = Intensities[,cond == sample_combi[1,s]]
  sample2 = Intensities[,cond == sample_combi[2,s]]
  X =sample1-sample2

  # This segment performs median correction.
    for (j in 1:length(sample_names)) {
    X[,j] = X[,j] - median(X[,j],na.rm = TRUE)}

  diff_intensities_ric[[s]] = X
  fit=eBayes(lmFit(X))
  fit$p.adj=p.adjust(fit$p.value,method="BH")

    fit$p.adj[is.na(fit$p.adj)]=1

  dt = data.frame(ENSGid = proteins_ric$ENSGid,
                  symbol = proteins_ric$symbol,
                  Know_RBP = proteins_ric$Know_RBP,
                  log2FC = fit$coefficients[,1],
                  p.value = fit$p.value[,1],
                  p.adj = fit$p.adj,
                  stringsAsFactors=FALSE)
  dt$sig=''
  dt$sig[dt$p.adj<0.1]='*'
  dt$sig[dt$p.adj<0.01]='**'
  dt=dt[,c(1,2,3,4,5,6,7)]
  diff_table_ric[[s]] = dt
}

save(diff_table_ric, file=file.path("t_test/diff_table_ric.rda"))
save(diff_table_who, file=file.path("t_test/diff_table_who.rda"))
```

Saving the diff tables as text files

```
for (s in names(diff_table_ric)) {
  write.table(diff_table_ric[[s]][order(diff_table_ric[[s]]$p.value,
                                        diff_table_ric[[s]]$log2FC),],
              file=file.path("t_test",paste0("RIC_",s,'.txt')),
              sep="\t",row.names =FALSE, quote=FALSE)
```

```
  page = openPage(file.path("t_test",paste0("RIC_",s,".html")),link.css="hwriter.css")
  hwrite(paste0("Result list ",s),heading=1,page=page)
  table=diff_table_ric[[s]][order(diff_table_ric[[s]]$p.adj, diff_table_ric[[s]]$log2FC),]
  row.names(table)=NULL
  hwrite(format(table,digits=4),
              page=page)
}

for (s in names(diff_table_who)) {
  write.table(diff_table_who[[s]][order(diff_table_who[[s]]$p.value,
                                  diff_table_who[[s]]$log2FC),],
              file=file.path("t_test",paste0("Input_",s,'.txt')),
              sep="\t",row.names =FALSE, quote=FALSE)

  page = openPage(file.path("t_test",paste0("Input_",s,".html")),link.css="hwriter.css")
  hwrite(paste0("Result list ",s),heading=1,page=page)
  table=diff_table_who[[s]][order(diff_table_who[[s]]$p.adj, diff_table_who[[s]]$log2FC),]
  row.names(table)=NULL
  hwrite(format(table,digits=4),
              page=page)
}
```

Performing semi-quantitative analysis

This segment performs a semi-quantitative analysis, comparing the different experimental conditions, to identify strong on vs off and off vs on situations which escape the above statistical analysis due to missing values.

The analysis generates tables which contain proteins, which have detectable intensity values in one condition (in 3 or 2 replicates), but are detected only once or absent in a second condition.

```
semi_dir=file.path('semi_quant')
dir.create(semi_dir, showWarnings = FALSE)

add_summary=function(proteinset){
  hour18_index=grep('18',colnames(proteinset))
  hour4_index=grep('4',colnames(proteinset))
  mock_index=grep('mock',colnames(proteinset))
  proteinset=cbind(proteinset,
                   hour18_total=apply(proteinset,1,
                                      function(x) {
                                        sum(!is.na(x[hour18_index]))
                                      }))
  proteinset=cbind(proteinset,
                   hour4_total=apply(proteinset,1,
                                     function(x) {
                                       sum(!is.na(x[hour4_index]))
                                     }))
  proteinset=cbind(proteinset,
                   mock_total=apply(proteinset,1,
                                    function(x) {
                                      sum(!is.na(x[mock_index]))
                                    }))
```

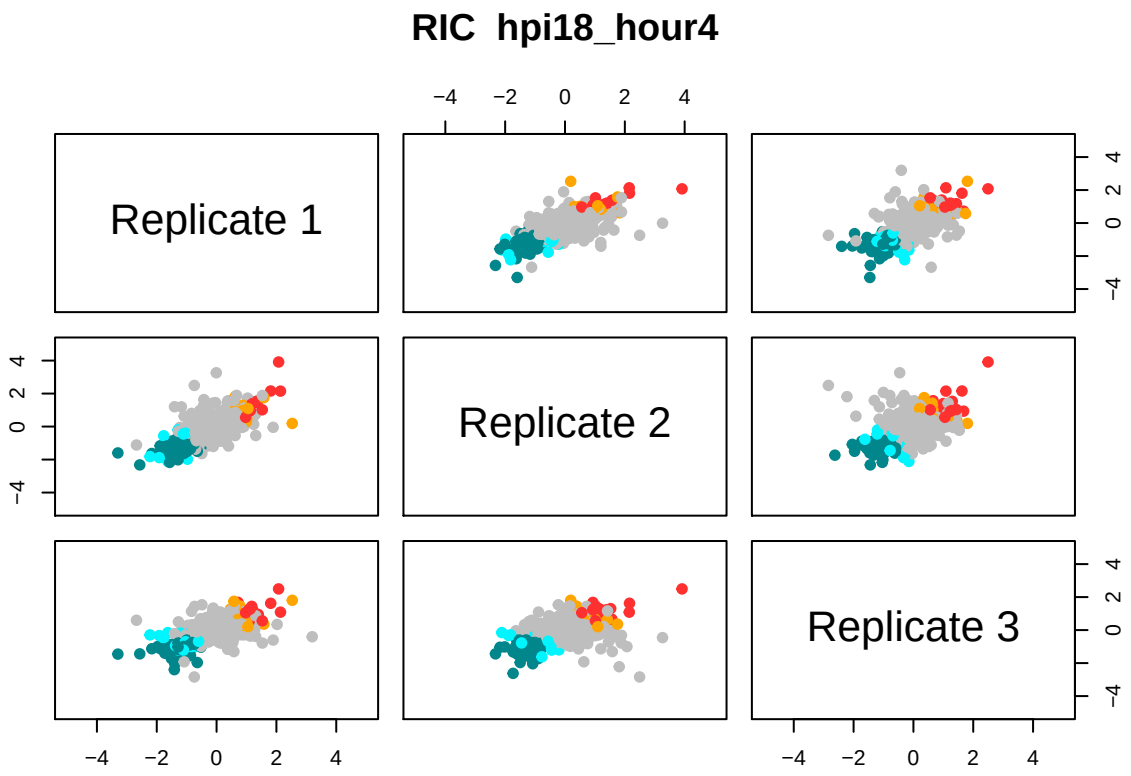
```
  proteinset
}

semi_hour4_mock=apply(is.na(Intensities[,cond == "mock"]) -
                        is.na(Intensities[,cond == "hour4"]), 1,
  function(x) { sum(x) } )
semi_hour4_mock_up = proteins_ric[semi_hour4_mock>=2,]
semi_hour4_mock_up =
  semi_hour4_mock_up[,c(1,2,3,6,9,12,5,8,11,4,7,10)]
semi_hour4_mock_up=add_summary(semi_hour4_mock_up)
write.table(semi_hour4_mock_up, file=
              file.path(semi_dir,'semi_hour4_mock_up.txt'),
              row.names=FALSE,quote=FALSE)

semi_hour4_mock_down=proteins_ric[semi_hour4_mock<=-2,]
semi_hour4_mock_down=
  semi_hour4_mock_down[,c(1,2,3,6,9,12,5,8,11,4,7,10)]
semi_hour4_mock_down=add_summary(semi_hour4_mock_down)
write.table(semi_hour4_mock_down, file=
              file.path(semi_dir,'semi_hour4_mock_down.txt'),
              row.names=FALSE,quote=FALSE)

semi_hour18_mock=apply(is.na(Intensities[,cond == "mock"]) -
                        is.na(Intensities[,cond == "hour18"]), 1,
  function(x) { sum(x) } )
semi_hour18_mock_up=proteins_ric[semi_hour18_mock>=2,]
semi_hour18_mock_up=
  semi_hour18_mock_up[,c(1,2,3,6,9,12,5,8,11,4,7,10)]
semi_hour18_mock_up=add_summary(semi_hour18_mock_up )
write.table(semi_hour18_mock_up, file=
              file.path(semi_dir,'semi_hour18_mock_up.txt'),
              row.names=FALSE,quote=FALSE)

semi_hour18_mock_down=proteins_ric[semi_hour18_mock<=-2,]
semi_hour18_mock_down=
  semi_hour18_mock_down[,c(1,2,3,6,9,12,5,8,11,4,7,10)]
semi_hour18_mock_down=add_summary(semi_hour18_mock_down)
write.table(semi_hour18_mock_down,
            file=file.path(semi_dir,'semi_hour18_mock_down.txt'),
            row.names=FALSE,quote=FALSE)

semi_hour18_hour4=apply(is.na(Intensities[,cond == "hour4"]) -
                        is.na(Intensities[,cond == "hour18"]), 1,
  function(x) { sum(x) } )
semi_hour18_hour4_up=proteins_ric[semi_hour18_hour4>=2,]
semi_hour18_hour4_up=
  semi_hour18_hour4_up[,c(1,2,3,6,9,12,5,8,11,4,7,10)]
semi_hour18_hour4_up=add_summary(semi_hour18_hour4_up)
write.table(semi_hour18_hour4_up,
            file=file.path(semi_dir,'semi_hour18_hour4_up.txt'),
            row.names=FALSE,quote=FALSE)

semi_hour18_hour4_down=proteins_ric[semi_hour18_hour4<=-2,]
```

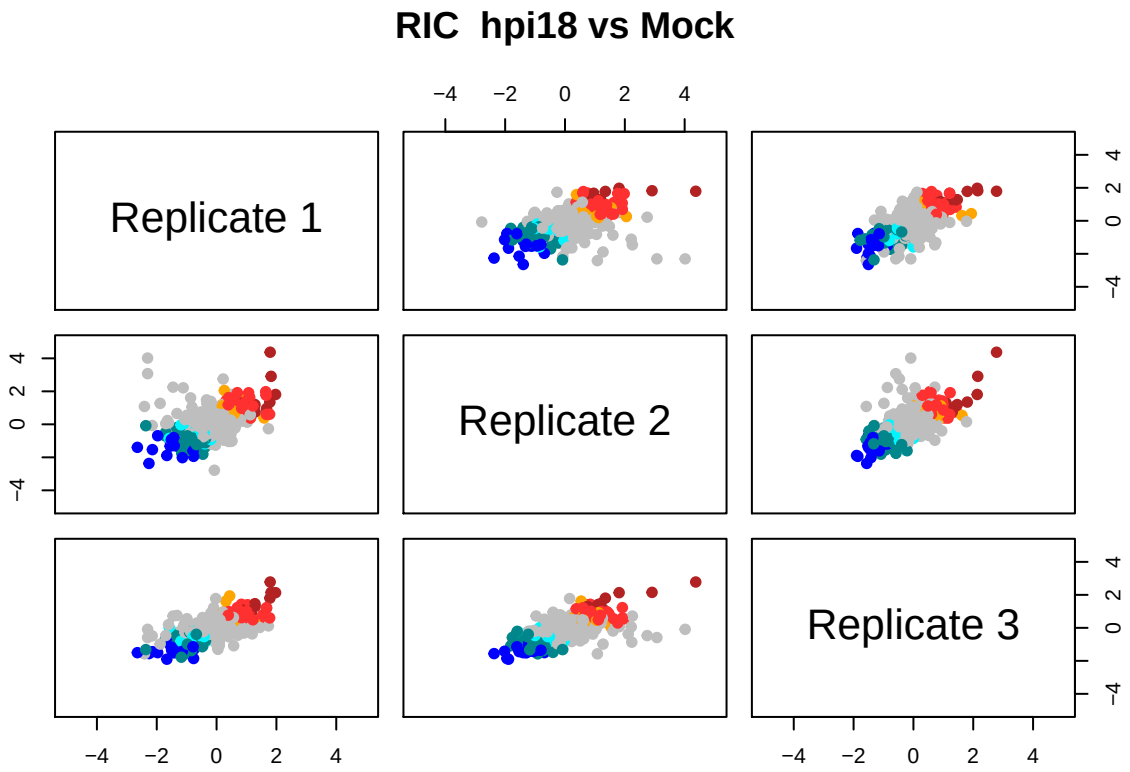
```
semi_hour18_hour4_down=
  semi_hour18_hour4_down[,c(1,2,3,6,9,12,5,8,11,4,7,10)]
semi_hour18_hour4_down=add_summary(semi_hour18_hour4_down)
write.table(semi_hour18_hour4_down,
            file=file.path(semi_dir,'semi_hour18_hour4_down.txt'),
            row.names=FALSE,quote=FALSE)
```

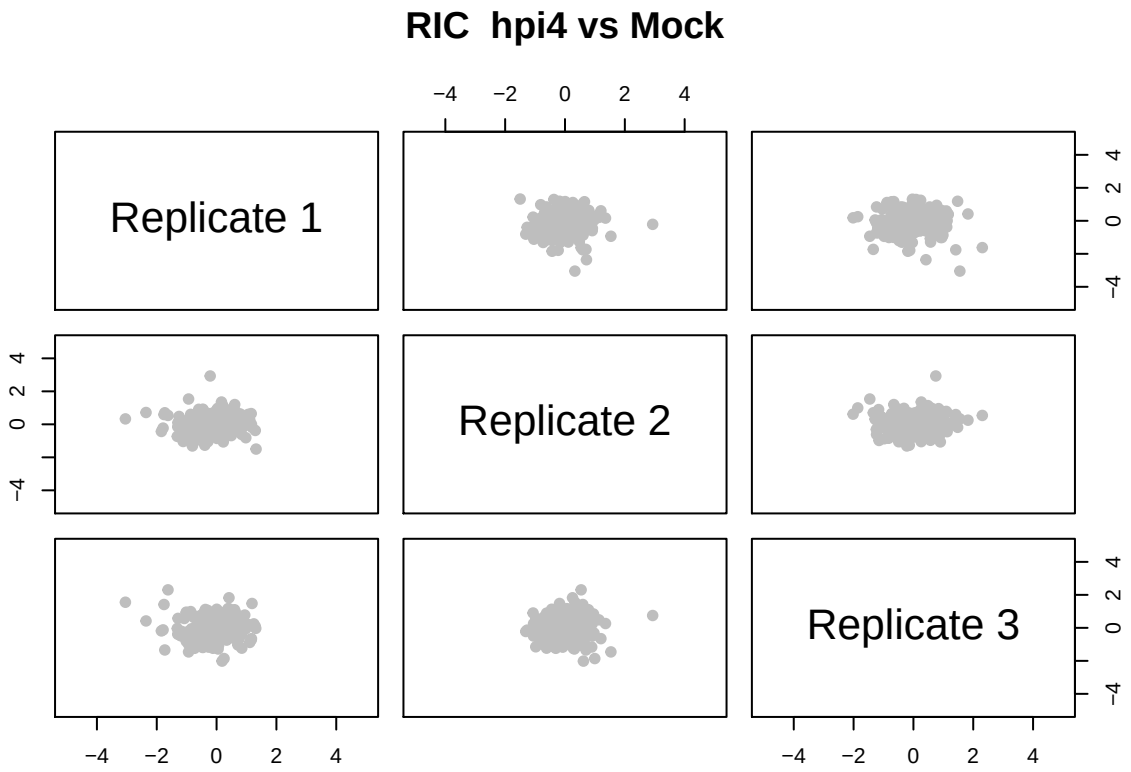
Scatterplots

This segment generates scatterplots, which display the log2 fold change comparing the different replicates, highlighting significantely changed proteins in colour.

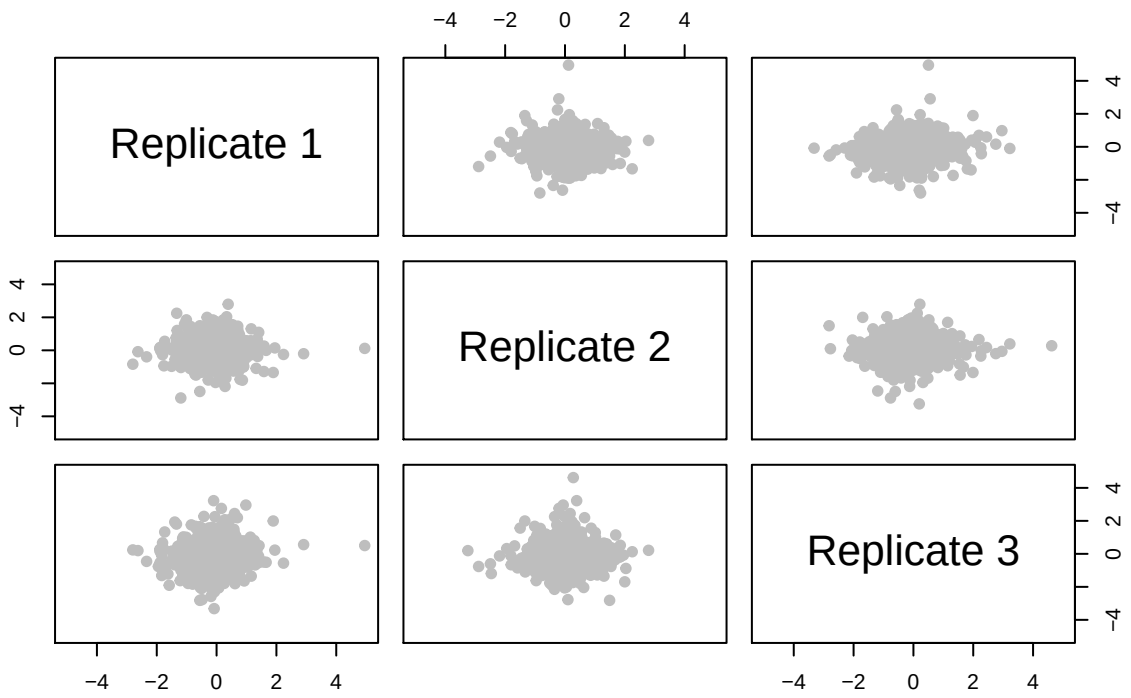
```
for (s in colnames(sample_combi)) {
  col = ifelse(diff_table_ric[[s]]$p.adj <= 0.1, "orange", "gray")
  col[which(diff_table_ric[[s]]$p.adj <= 0.05)] = "firebrick1"
  col[which(diff_table_ric[[s]]$p.adj <= 0.01)] = "firebrick"
  col[which(diff_table_ric[[s]]$p.adj <= 0.1 &
              diff_table_ric[[s]]$log2FC < 0)] = "turquoise1"
    col[which(diff_table_ric[[s]]$p.adj <= 0.05 &
              diff_table_ric[[s]]$log2FC < 0)] = "turquoise4"
  col[which(diff_table_ric[[s]]$p.adj <= 0.01 &
              diff_table_ric[[s]]$log2FC < 0)] = "blue"

  pairs(diff_intensities_ric[[s]],col=col, pch=19,labels = c("Replicate 1", "Replicate 2", "Replicate 3
```

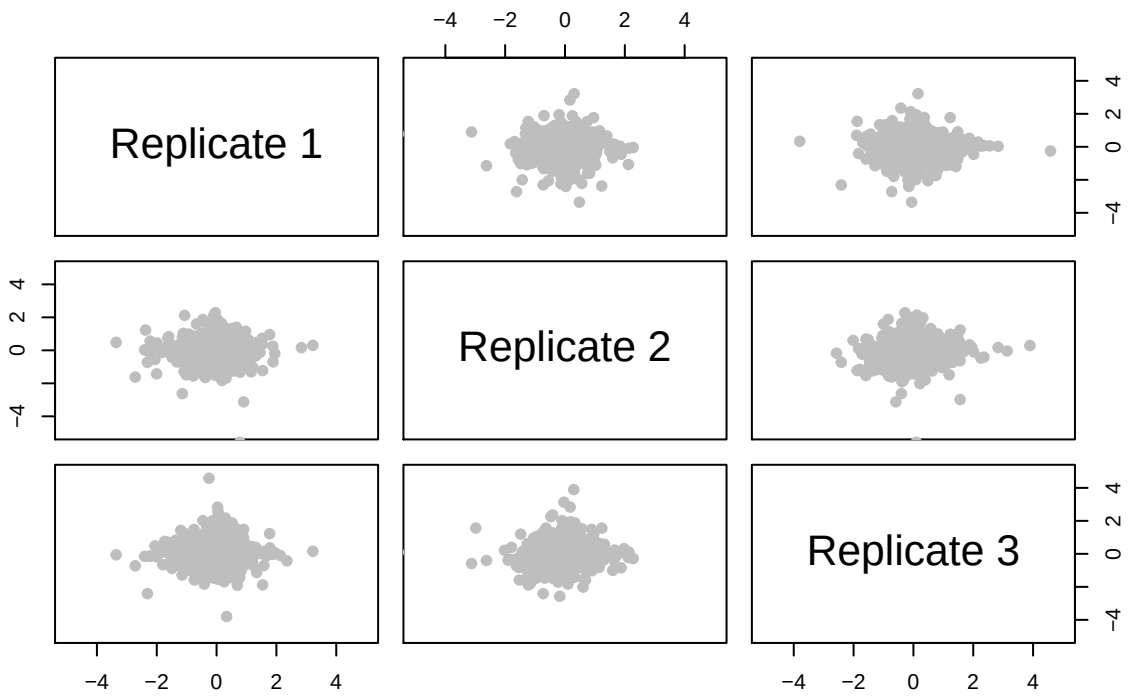


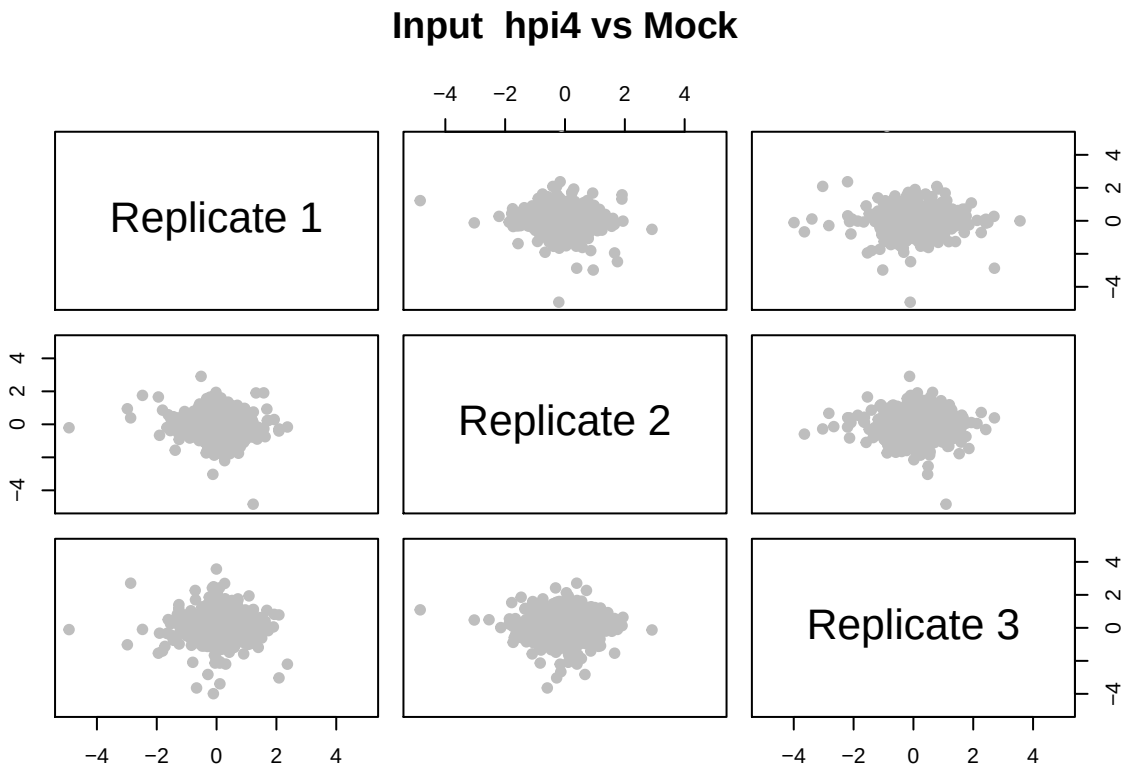

RIC hpi18 vs Mock
Replicate 1
Replicate 2
Replicate 3
−4
−2
0
2
4




```
for (s in colnames(sample_combi)) {
  col = ifelse(diff_table_who[[s]]$p.adj <= 0.1, "orange", "gray")
  col[which(diff_table_who[[s]]$p.adj <= 0.05)] = "firebrick1"
  col[which(diff_table_who[[s]]$p.adj <= 0.01)] = "firebrick"
  col[which(diff_table_who[[s]]$p.adj <= 0.1 &
              diff_table_who[[s]]$log2FC < 0)] = "turquoise1"
    col[which(diff_table_who[[s]]$p.adj <= 0.05 &
              diff_table_who[[s]]$log2FC < 0)] = "turquoise4"
  col[which(diff_table_who[[s]]$p.adj <= 0.01 &
              diff_table_who[[s]]$log2FC < 0)] = "blue"

pairs(diff_intensities_who[[s]],col=col, pch=19,labels = c("Replicate 1", "Replicate 2", "Replicate 3")
```


Input hpi18_hour4
Replicate 1
Replicate 2
Replicate 3
−4
−2
0
2
4


Input hpi18 vs Mock
Replicate 1
Replicate 2
Replicate 3
−4
−2
0
2
4



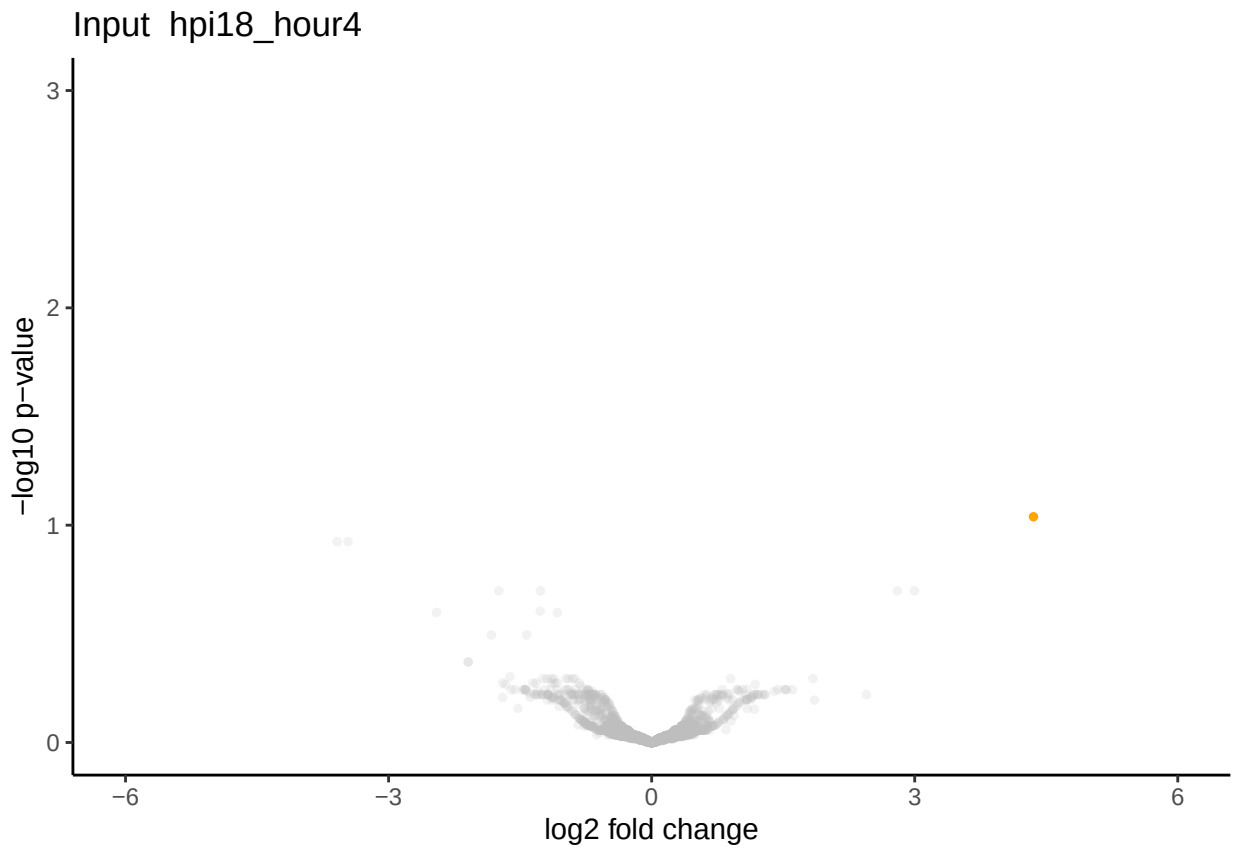
Volcanoplot

Here volcanoplots displaying the log2 fold change vs significance over the 3 biological replicates are generated, with different significance levels highlighted in colour

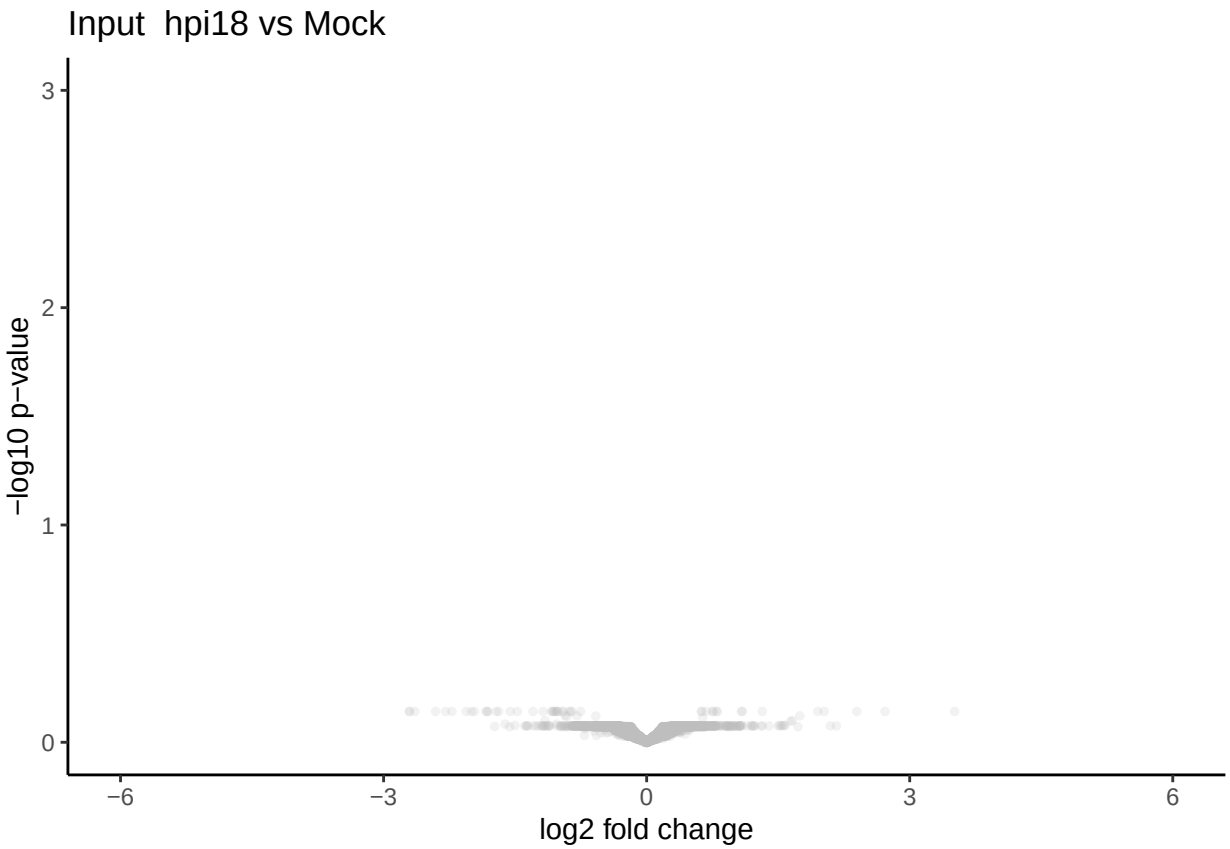
```
for (s in colnames(sample_combi)) {
  volData = diff_table_who[[s]]
  volData = volData[!is.na(volData$p.adj),]
 col = ifelse(volData$p.adj <= 0.1, "orange", "gray")
  col[which(volData$p.adj <= 0.05)] = "firebrick1"
  col[which(volData$p.adj <= 0.01)] = "firebrick"
  col[which(volData$p.adj <= 0.1 &  volData$log2FC < 0)] = "turquoise1"
  col[which(volData$p.adj <= 0.05 &  volData$log2FC < 0)] = "turquoise4"
  col[which(volData$p.adj <= 0.01 &  volData$log2FC < 0)] = "blue"
  col[col=='gray']=adjustcolor('gray', alpha=0.2)

   g = ggplot(volData, aes(log2FC,-log10(p.adj)))+
      geom_point(size=1, color=col)+
      theme(panel.grid.major = element_blank(),
            panel.grid.minor = element_blank(),
            panel.background = element_blank(),
            axis.line = element_line(colour = "black"))+

    xlim(c(-6, 6)) + ylim(c(0, 3)) +
    xlab("log2 fold change") + ylab("-log10 p-value") +
```

```
    labs(title = paste("Input ", gsub("_mock"," vs Mock",gsub("diff_hour","hpi",s))))

  print(g)
}
```

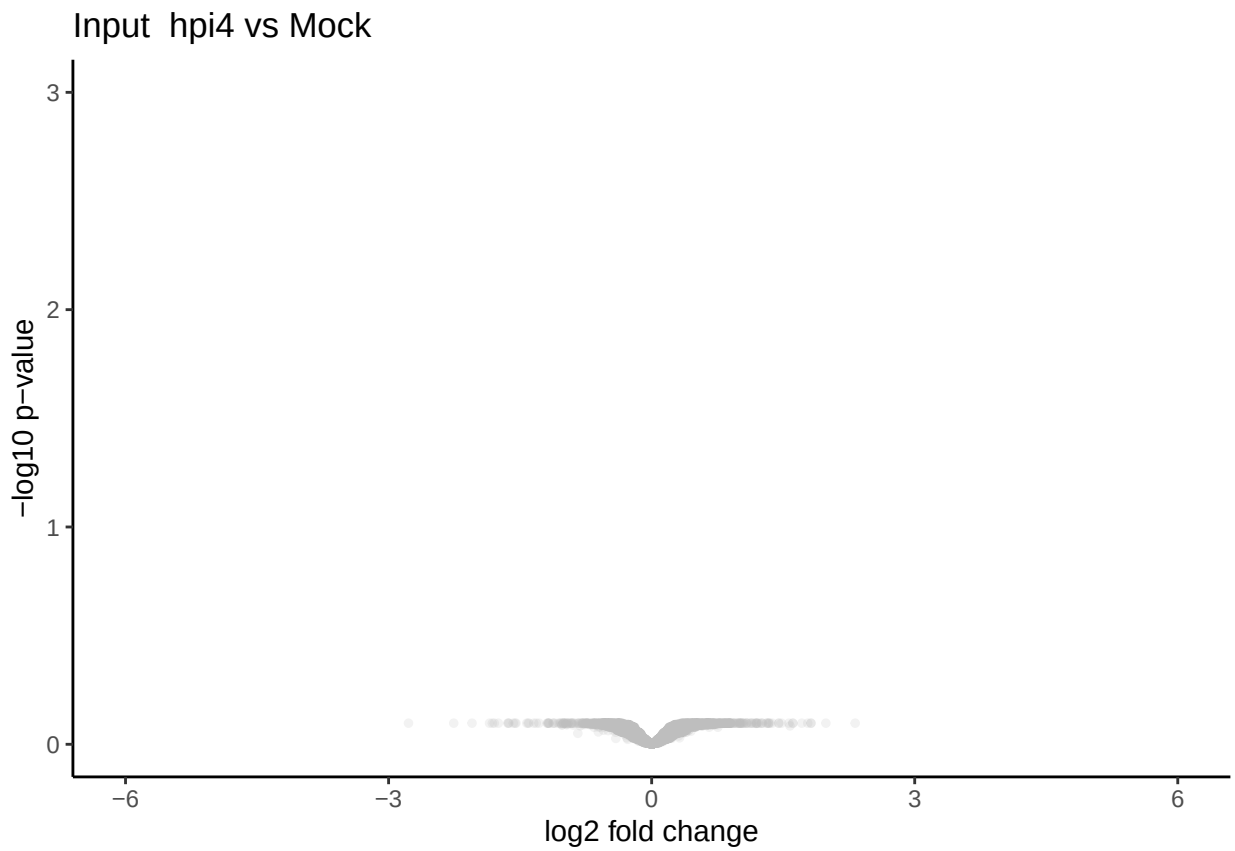
```
## Warning: Removed 639 rows containing missing values (geom_point).
```




```
## Warning: Removed 638 rows containing missing values (geom_point).
```

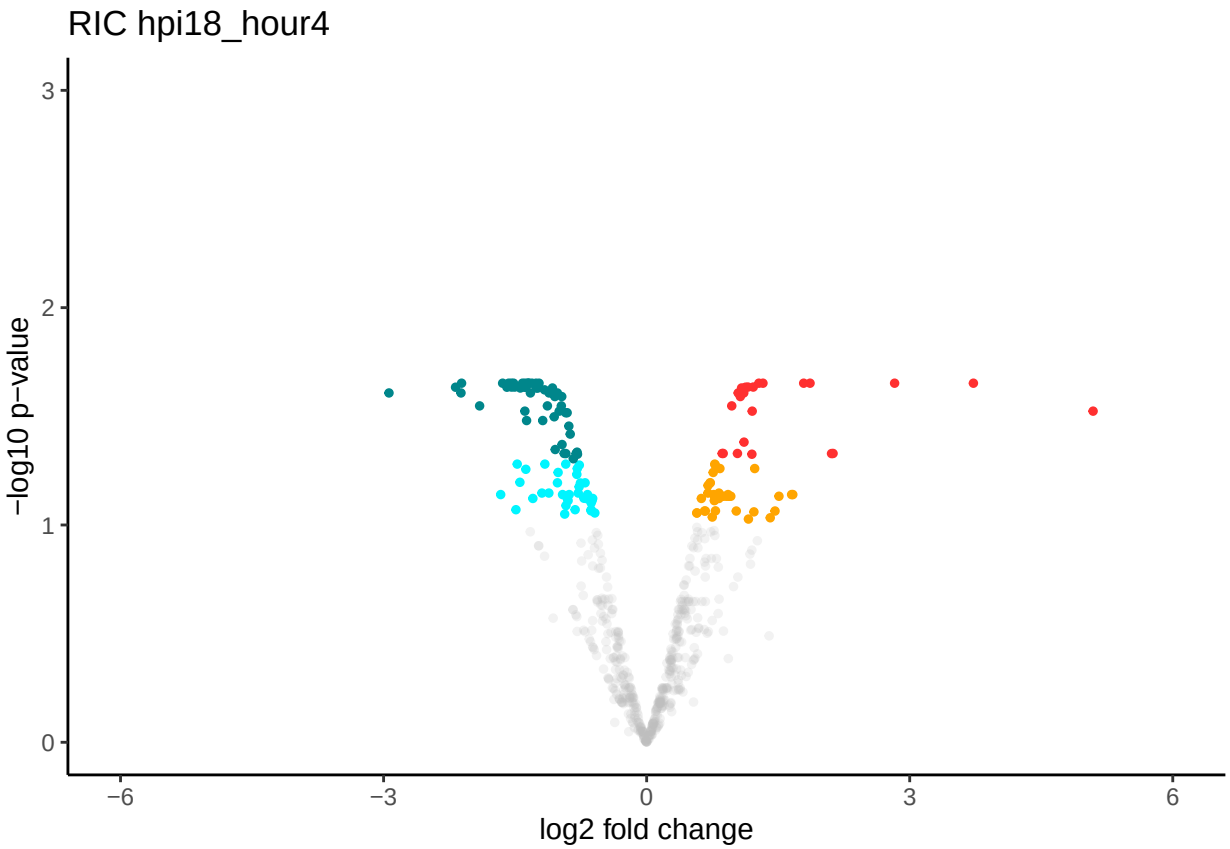



```
## Warning: Removed 676 rows containing missing values (geom_point).
```

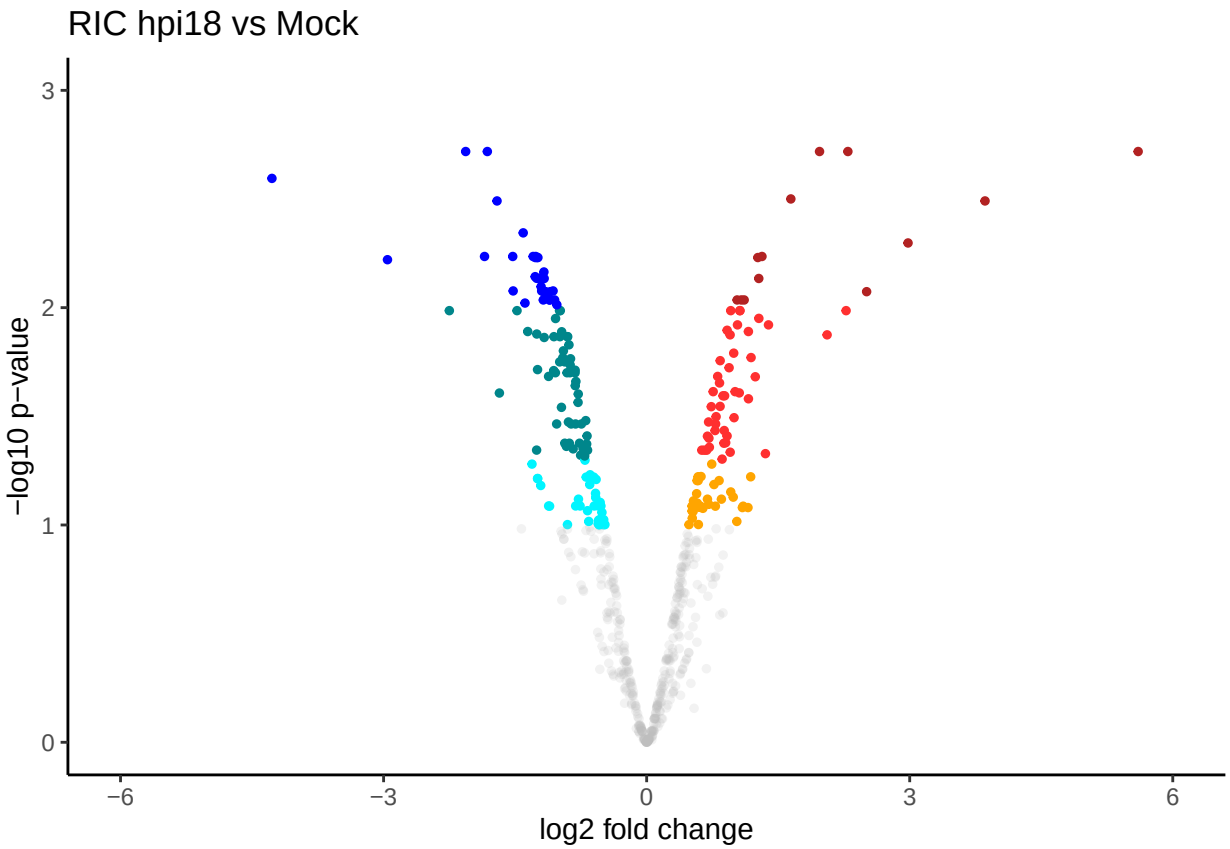



```
for (s in colnames(sample_combi)) {
  volData = diff_table_ric[[s]]
  volData = volData[!is.na(volData$p.adj),]
 col = ifelse(volData$p.adj <= 0.1, "orange", "gray")
  col[which(volData$p.adj <= 0.05)] = "firebrick1"
  col[which(volData$p.adj <= 0.01)] = "firebrick"
  col[which(volData$p.adj <= 0.1 &  volData$log2FC < 0)] = "turquoise1"
  col[which(volData$p.adj <= 0.05 &  volData$log2FC < 0)] = "turquoise4"
  col[which(volData$p.adj <= 0.01 &  volData$log2FC < 0)] = "blue"
  col[col=='gray']=adjustcolor('gray', alpha=0.2)

   g = ggplot(volData, aes(log2FC,-log10(p.adj)))+
      geom_point(size=1, color=col)+
      theme(panel.grid.major = element_blank(),
            panel.grid.minor = element_blank(),
            panel.background = element_blank(),
            axis.line = element_line(colour = "black"))+

    xlim(c(-6, 6)) + ylim(c(0, 3)) +
    xlab("log2 fold change") + ylab("-log10 p-value") +
    labs(title = paste("RIC", gsub("_mock"," vs Mock",gsub("diff_hour","hpi",s))))

  print(g)
}
```

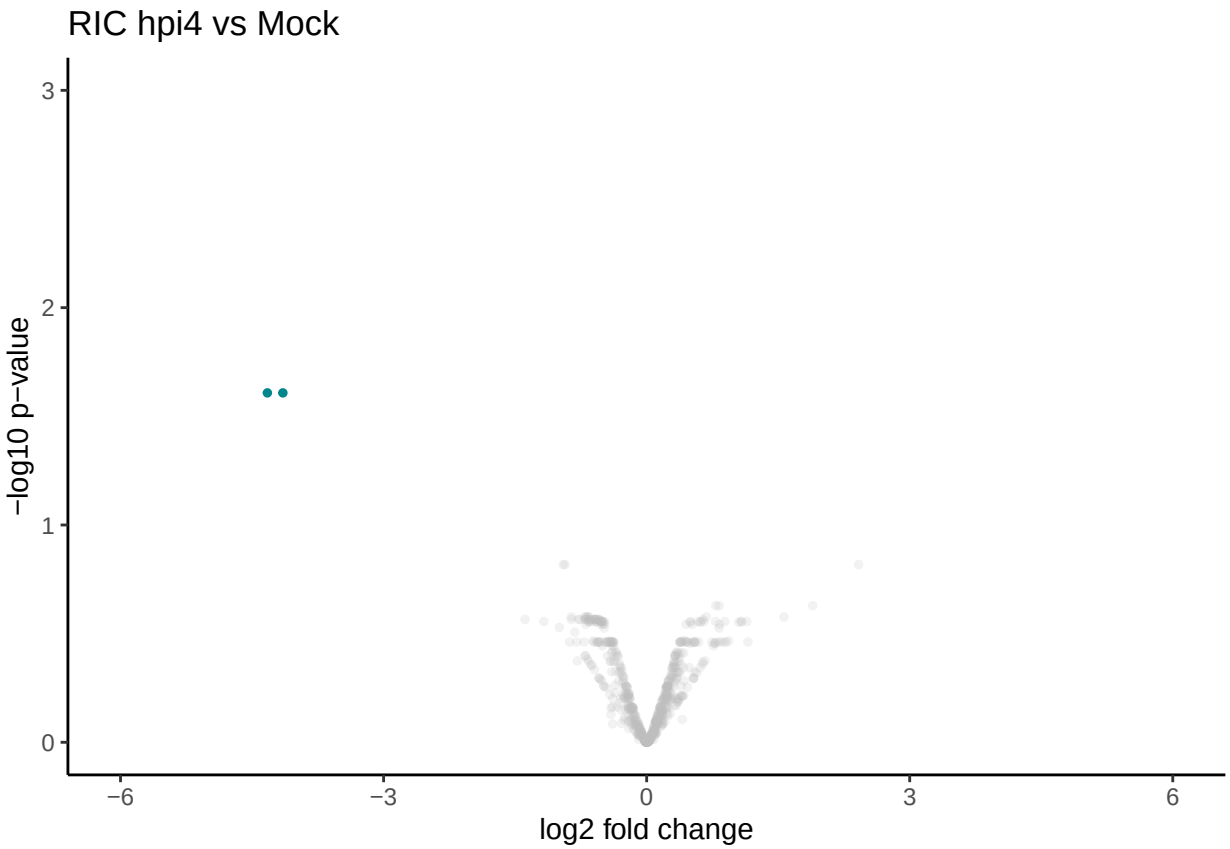
```
## Warning: Removed 176 rows containing missing values (geom_point).
```




```
## Warning: Removed 178 rows containing missing values (geom_point).
```




```
## Warning: Removed 173 rows containing missing values (geom_point).
```

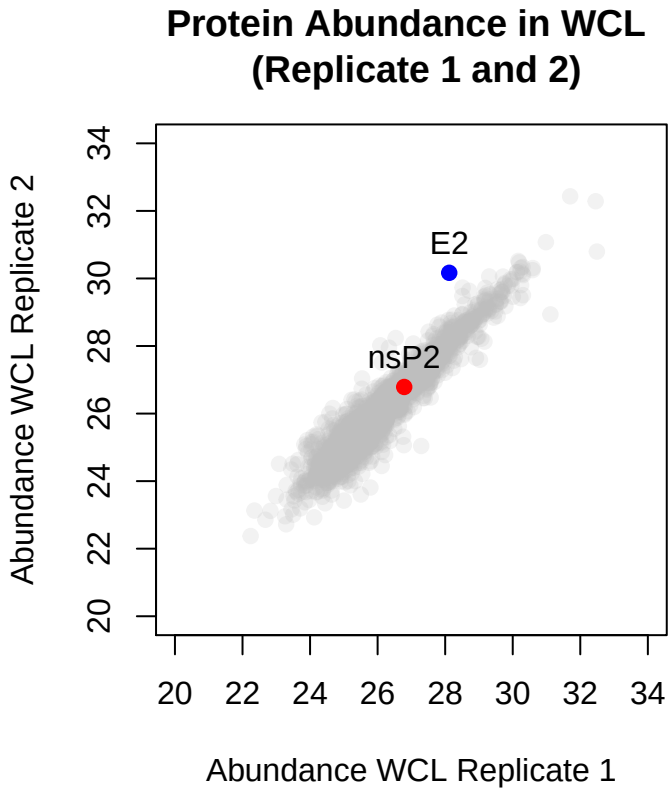


Intensity between two replicates

Plotting the Intensity of proteins between two replicates in the Input

Selected proteins are highlighted in colour.

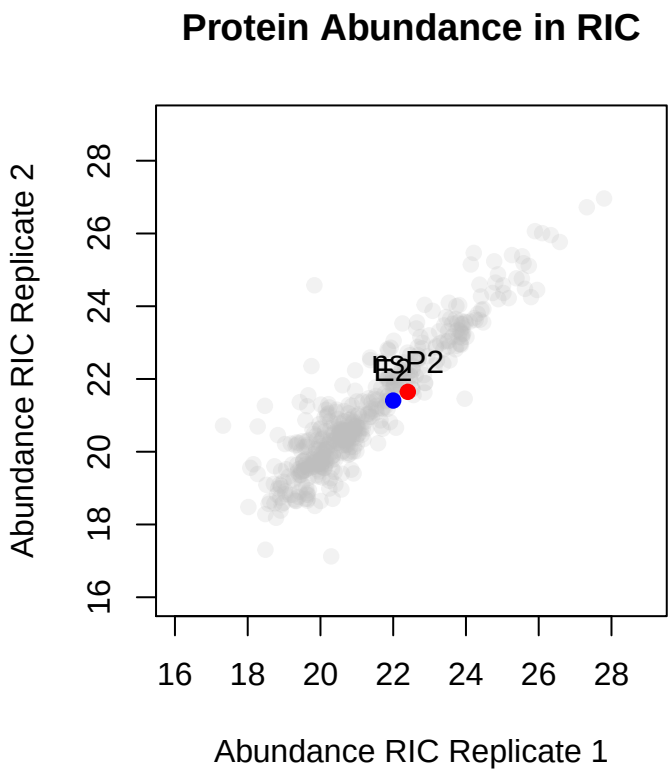
```
grepl('SV_wt_nsP2',proteins_who_batch$ENSGid)->A
grepl('SV_wt_E2',proteins_who_batch$ENSGid)->B

par(pty="s")

plot(proteins_who_batch[,4], proteins_who_batch[,7], col=alpha('gray',0.2),
     pch=19,xlim=c(20,34), ylim=c(20,34),
       xlab='Abundance WCL Replicate 1', ylab='Abundance WCL Replicate 2')
title(main="Protein Abundance in WCL \n (Replicate 1 and 2)")

points(proteins_who_batch[A,4], proteins_who_batch[A,7], col='red', pch=19)
text(proteins_who_batch[A,4], proteins_who_batch[A,7],
     labels="nsP2", cex= 1, pos=3)

points(proteins_who_batch[B,4], proteins_who_batch[B,7], col='blue', pch=19)
text(proteins_who_batch[B,4], proteins_who_batch[B,7],
     labels="E2", cex= 1, pos=3)
```



Plotting the Intensity of proteins between two replicates in the RIC.

Selected proteins are highlighted in colour.

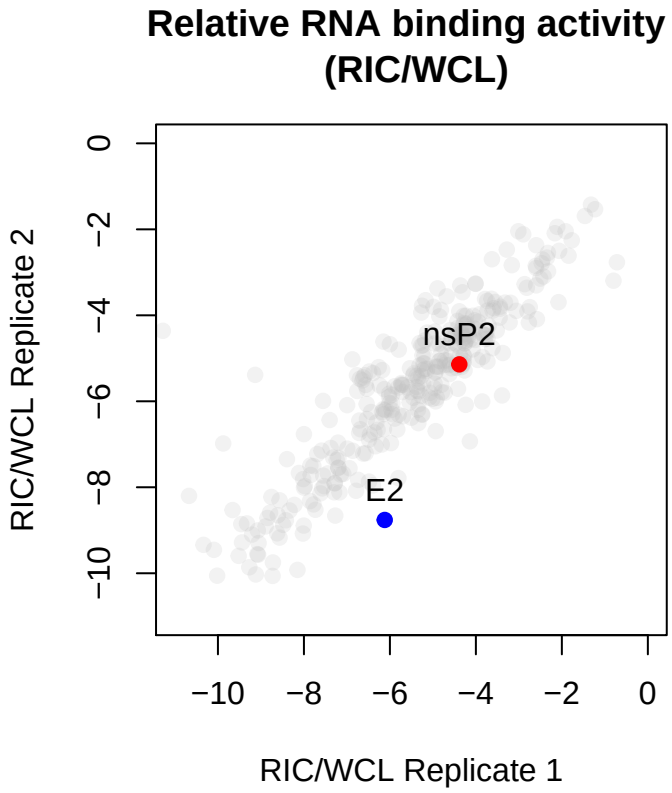
```
grepl('SV_wt_nsP2',proteins_ric$ENSGid)->AI
grepl('SV_wt_E2',proteins_ric$ENSGid)->BI

par(pty="s")

plot(proteins_ric[,4], proteins_ric[,7], col=alpha('gray',0.2),
     pch=19,xlim=c(16,29), ylim=c(16,29),
       xlab='Abundance RIC Replicate 1', ylab='Abundance RIC Replicate 2')
title(main="Protein Abundance in RIC")

points(proteins_ric[AI,4], proteins_ric[AI,7], col='red', pch=19)
text(proteins_ric[AI,4], proteins_ric[AI,7],
     labels="nsP2", cex= 1, pos=3)

points(proteins_ric[BI,4], proteins_ric[BI,7], col='blue', pch=19)
text(proteins_ric[BI,4], proteins_ric[BI,7],
     labels="E2", cex= 1, pos=3)
```



Plotting the Intensity of proteins between two replicates in the RIC/WCL comparision to estimate RNA binding affinity

Selected proteins are highlighted in colour.

```
grepl('SV_wt_nsP2',proteins_int$ENSGid)->AI
grepl('SV_wt_E2',proteins_int$ENSGid)->BI

par(pty="s")

plot(proteins_int[,4], proteins_int[,7], col=alpha('gray',0.2),
     pch=19,xlim=c(-11,0), ylim=c(-11,0),
       xlab='RIC/WCL Replicate 1', ylab='RIC/WCL Replicate 2')
title(main="Relative RNA binding activity \n (RIC/WCL)")

points(proteins_int[AI,4], proteins_int[AI,7], col='red', pch=19)
text(proteins_int[AI,4], proteins_int[AI,7],
     labels="nsP2", cex= 1, pos=3)

points(proteins_int[BI,4], proteins_int[BI,7], col='blue', pch=19)
text(proteins_int[BI,4], proteins_int[BI,7],
     labels="E2", cex= 1, pos=3)
```



Sessioninfo

```
sessionInfo()
```

```
## R version 3.6.1 (2019-07-05)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 18362)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United Kingdom.1252
## [2] LC_CTYPE=English_United Kingdom.1252
## [3] LC_MONETARY=English_United Kingdom.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United Kingdom.1252
##
## attached base packages:
##  [1] stats4    parallel  grid      stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] pheatmap_1.0.12     gridSVG_1.7-1       Biostrings_2.54.0
##  [4] XVector_0.26.0      IRanges_2.20.2      S4Vectors_0.24.3
##  [7] RColorBrewer_1.1-2  hwriter_1.3.2       RBDmap_0.0.17
```

```
## [10] XML_3.99-0.3        limma_3.42.2        Biobase_2.46.0
## [13] BiocGenerics_0.32.0 VennDiagram_1.6.20  futile.logger_1.4.3
## [16] ggrepel_0.8.2       ggplot2_3.3.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.4           prettyunits_1.1.1    assertthat_0.2.1
##  [4] digest_0.6.25        BiocFileCache_1.10.2 R6_2.4.1
##  [7] futile.options_1.0.1 RSQLite_2.2.0        evaluate_0.14
## [10] httr_1.4.1           pillar_1.4.3         zlibbioc_1.32.0
## [13] rlang_0.4.5          progress_1.2.2       curl_4.3
## [16] blob_1.2.1           rmarkdown_2.1        labeling_0.3
## [19] stringr_1.4.0        bit_1.1-15.2         biomaRt_2.42.0
## [22] munsell_0.5.0        compiler_3.6.1       xfun_0.12
## [25] pkgconfig_2.0.3      askpass_1.1          htmltools_0.4.0
## [28] openssl_1.4.1        tidyselect_1.0.0     tibble_2.1.3
## [31] crayon_1.3.4         dplyr_0.8.5          dbplyr_1.4.2
## [34] withr_2.1.2          rappdirs_0.3.1       jsonlite_1.6.1
## [37] gtable_0.3.0         lifecycle_0.2.0      DBI_1.1.0
## [40] magrittr_1.5         formatR_1.7          scales_1.1.0
## [43] stringi_1.4.6        farver_2.0.3         vctrs_0.2.4
## [46] lambda.r_1.2.4       tools_3.6.1          bit64_0.9-7
## [49] glue_1.3.2           purrr_0.3.3          hms_0.5.3
## [52] yaml_2.2.1           AnnotationDbi_1.48.0 colorspace_1.4-1
## [55] memoise_1.1.0        knitr_1.28
```