RiboTag Neural Development

Bernd Fischer

b.fischer@dkfz.de

February 11, 2017

Contents

Libraries are loaded.

```
library(limma)
library(DESeq2)
library(hwriter)
library(RColorBrewer)
library(genefilter)
library(grid)
library(gridSVG)
source(file.path("R","hwriteSidebar.R"))
```

The dataset is loaded and genes with a count less than 1 over all samples are removed.

```
load(file.path("data","rawCounts.rda"))
load(file.path("data","expectedCounts.rda"))
load(file.path("data","Anno.rda"))

# pre-filter low count rows
Anno <- Anno[ rowSums(rawCounts) > 1, ]
expectedCounts <- expectedCounts[ rowSums(rawCounts) > 1, ]
rawCounts <- rawCounts[ rowSums(rawCounts) > 1, ]
Anno$transript_type[is.na(Anno$transript_type)] = "other"
```

The samples are annotated.

```
load("data/SampleList.rda")

SampleList$Population = factor(SampleList$Population,
                               levels=c("NSC","NSC/ENB","ENB","LNB","LNB/NEURON","NEURON"))
SampleList$Type = factor(SampleList$Type,
                         levels=c("RIBO_IPctr","RNAseq","RIBO_IPseq"))
SampleList$Replicate = factor(SampleList$Replicate,
                              levels=c("RIBO_IPctr","RNAseq","RIBO_IPseq"))
```

1 Differential gene expression of Total RNA

Sample indices for five different pairs of total RNA datasets are extracted.

```
SamplesRNAseq = list(
  "ENB-NSC" = which(SampleList$Population %in% c("NSC","ENB") &
                    SampleList$Type == "RNAseq"),
  "LNB-ENB" = which(SampleList$Population %in% c("ENB","LNB") &
                    SampleList$Type == "RNAseq"),
  "NEURON-LNB" = which(SampleList$Population %in% c("LNB","NEURON") &
                    SampleList$Type == "RNAseq"))

DesignRNAseq = list(
  "ENB-NSC" = as.formula("~Population"),
  "LNB-ENB" = as.formula("~Population"),
  "NEURON-LNB" = as.formula("~Population"))

for (n in names(DesignRNAseq)) {
  SamplesRNAseq[[n]] = SamplesRNAseq[[n]][
    order(SampleList[SamplesRNAseq[[n]],as.character(DesignRNAseq[[n]])[2]],
          SampleList[SamplesRNAseq[[n]],"Replicate"])]
}
```

Differential gene expression analysis is performed for each of these comparisons.

1.1 Differential gene expression in ENB-NSC

Create an output directory.

```
n = "ENB-NSC"
type="RNAseq"
Samples = SamplesRNAseq
filter = "none"
d = file.path("result",type, n)
dir.create(d, recursive = TRUE, showWarnings = FALSE)
mainmenu = file.path("..","..","..","mainmenu.html")

load(file.path("data","ENSG2category.rda"))
source(file.path("R","doGSEA.R"))
```

The count matrix is copied for convenience.

```
X = rawCounts[,Samples[[n]]]
Design = DesignRNAseq
```

A DESeq dataset is created that contains the respective samples.

```
dds <- DESeqDataSetFromMatrix(countData = X,
                              colData = SampleList[Samples[[n]],],
                              design = Design[[n]])

## converting counts to integer mode

## factor levels were dropped which had no samples

colData(dds)
```

```
## DataFrame with 6 rows and 6 columns
##                       Name Population     Type Replicate     Comment         OriginalName
##                <character>   <factor> <factor>  <factor> <character>          <character>
## NSC_RNAseq_1 NSC_RNAseq_1        NSC   RNAseq        NA                  01_Ticry5547YFP
## NSC_RNAseq_2 NSC_RNAseq_2        NSC   RNAseq        NA              14_Ticry8165_8166YFP
## NSC_RNAseq_3 NSC_RNAseq_3        NSC   RNAseq        NA                   02_Ticry6261YFP
## ENB_RNAseq_1 ENB_RNAseq_1        ENB   RNAseq        NA              15_DicryS8356_8495YF
## ENB_RNAseq_2 ENB_RNAseq_2        ENB   RNAseq        NA                  08_DicryS6365YFP
## ENB_RNAseq_3 ENB_RNAseq_3        ENB   RNAseq        NA              16_DicryS8355_8492YF
```

Log counts are computed and plotted as a pairs plot.

```
C = counts(dds)
C = log2(C+1)
lim = range(as.vector(C))
pairs(C,pch=20,cex=0.5,xlim=lim, ylim=lim,col="#00000022")
```

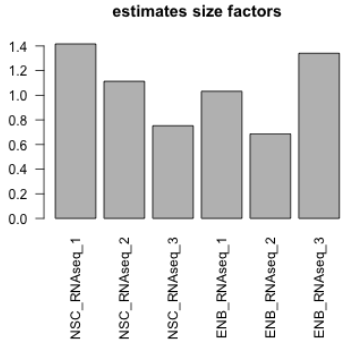
The standard DESeq2 workflow is used to for differential gene expression analysis. size factors are estimated using all genes. Subsequently, dispersion is estimated and p-values are computed by a Wald test.

```
dds = DESeq(dds)

## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship
```

```
## final dispersion estimates

## fitting model and testing

res = results(dds)
```

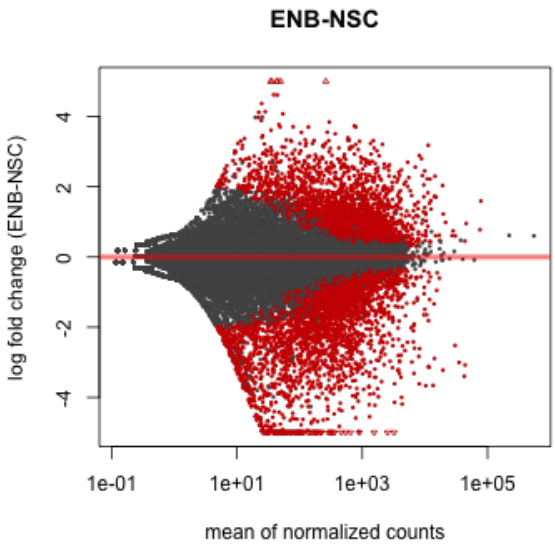
The size factors are plotted.

```
p = par(mar=c(10,4,4,2)+0.1)
barplot(sizeFactors(dds), main="estimates size factors",las=2)
par(p)
```



An MA plot shows in red the genes that are enriched by the RiboTag pull-down. For comparison, the second plot shows the genes that were used to estimate the size factors.

```
plotMA(res, ylim=c(-5,5), ylab=sprintf("log fold change (%s)",n), main=n)
```




```
load(file.path("data","ENSGannotation.rda"))
res$symbol = ENSGannotation[row.names(res),"name"]
res$symbol[is.na(res$symbol)] = row.names(res)[is.na(res$symbol)]

res$significant = ""
res$significant[which(res$padj <= 0.1)] = "*"
res$significant[which(res$padj <= 0.01)] = "**"
```

The data is saved.

```
save(dds, file=file.path(d, "dds.rda"))
save(res, file=file.path(d, "res.rda"))
```

```
col=rep("#00000022", nrow(res))
col[which(res$padj <= 0.1 & res$log2FoldChange > 0)] = brewer.pal(4,"RdBu")[2]
col[which(res$padj <= 0.1 & res$log2FoldChange < 0)] = brewer.pal(4,"RdBu")[3]
col[which(res$padj <= 0.01 & res$log2FoldChange > 0)] = brewer.pal(4,"RdBu")[1]
col[which(res$padj <= 0.01 & res$log2FoldChange < 0)] = brewer.pal(4,"RdBu")[4]

I = order(factor(col, levels=c("#00000022",brewer.pal(4,"RdBu")[c(2,3,1,4)])))
lim = range(as.vector(C))
pairs(C[I,],pch=20,cex=0.5,xlim=lim, ylim=lim,col=col[I])
```


```
fgup = row.names(res)[which(res$significant != "" & res$log2FoldChange > 0)]
fgdown = row.names(res)[which(res$significant != "" & res$log2FoldChange < 0)]
bg = row.names(res)[which(!is.na(res$padj))]
GSEAup = doGSEA(fgup,bg,ENSG2category, k=3)
GSEAdown = doGSEA(fgdown,bg,ENSG2category, k=3)
```

```
write.table(res, file=file.path(d, "resulttable.txt"), sep="\t", quote=FALSE)
page = openPage(file.path(d,"resulttable.html"), link.css="hwriter.css")
hwriteSidebar(page=page, link=mainmenu)
hwrite(sprintf("Differential gene expression (%s;%s); result table",type,n),
       heading=1, page=page)
hwrite("** = 1% FDR; * = 10% FDR",page=page,br=TRUE)
col = rep(NA, nrow(res))
col[which(res$significant != "" & res$log2FoldChange > 0)] = "#fbb4ae"
col[which(res$significant != "" & res$log2FoldChange < 0)] = "#b3cde3"
hwrite(as.data.frame(res), row.bgcolor=col, page=page)
closePage(page, splash=FALSE)
```

1.2 Differential gene expression in LNB-ENB

Create an output directory.

```
n = "LNB-ENB"
type="RNAseq"
Samples = SamplesRNAseq
filter = "none"
d = file.path("result",type, n)
dir.create(d, recursive = TRUE, showWarnings = FALSE)
mainmenu = file.path("..","..","..","mainmenu.html")

load(file.path("data","ENSG2category.rda"))
source(file.path("R","doGSEA.R"))
```

The count matrix is copied for convenience.

```
X = rawCounts[,Samples[[n]]]
Design = DesignRNAseq
```

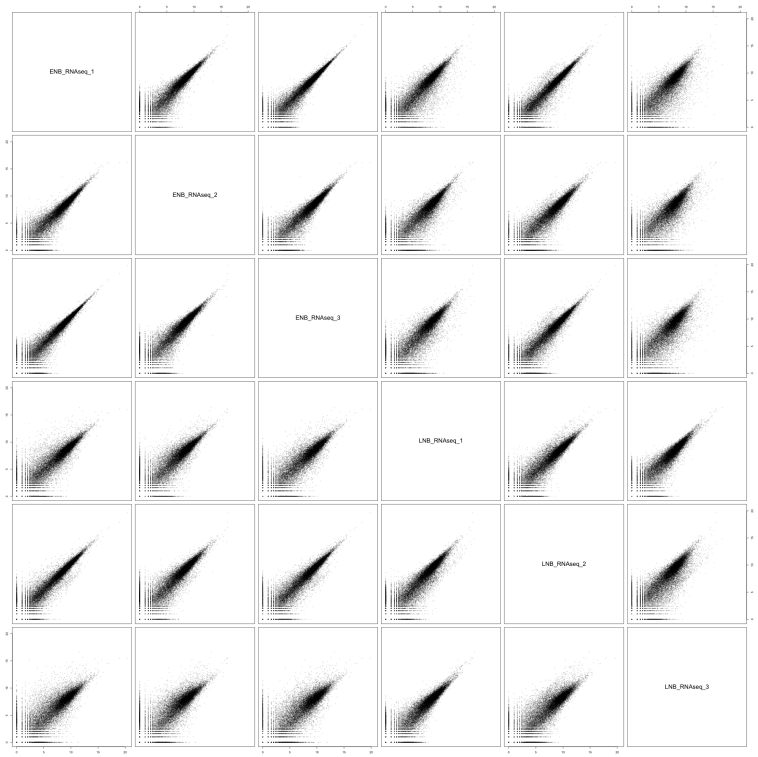
A DESeq dataset is created that contains the respective samples.

```
dds <- DESeqDataSetFromMatrix(countData = X,
                              colData = SampleList[Samples[[n]],],
                              design = Design[[n]])

## converting counts to integer mode

## factor levels were dropped which had no samples

colData(dds)
```

```
## DataFrame with 6 rows and 6 columns
##                       Name Population     Type Replicate     Comment         OriginalName
##                <character>   <factor> <factor>  <factor> <character>          <character>
## ENB_RNAseq_1 ENB_RNAseq_1        ENB   RNAseq        NA              15_DicryS8356_8495YF
## ENB_RNAseq_2 ENB_RNAseq_2        ENB   RNAseq        NA                  08_DicryS6365YFP
## ENB_RNAseq_3 ENB_RNAseq_3        ENB   RNAseq        NA              16_DicryS8355_8492YF
## LNB_RNAseq_1 LNB_RNAseq_1        LNB   RNAseq        NA                  10_DicryO6364YFP
## LNB_RNAseq_2 LNB_RNAseq_2        LNB   RNAseq        NA              18_DicryO8355_8492YF
## LNB_RNAseq_3 LNB_RNAseq_3        LNB   RNAseq        NA                  11_DicryO6365YFP
```

Log counts are computed and plotted as a pairs plot.

```
C = counts(dds)
C = log2(C+1)
lim = range(as.vector(C))
pairs(C,pch=20,cex=0.5,xlim=lim, ylim=lim,col="#00000022")
```



The standard DESeq2 workflow is used to for differential gene expression analysis. size factors are estimated using all genes. Subsequently, dispersion is estimated and p-values are computed by a Wald test.

```
dds = DESeq(dds)
```

```
## estimating size factors
```

```
## estimating dispersions
```

```
## gene-wise dispersion estimates
```

```
## mean-dispersion relationship
```

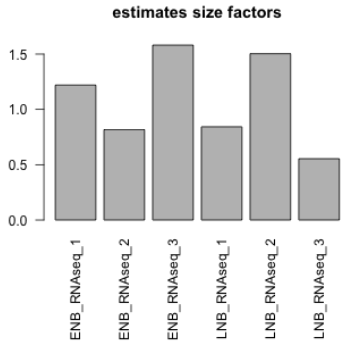
```
## final dispersion estimates
```

```
## fitting model and testing
```

```
res = results(dds)
```

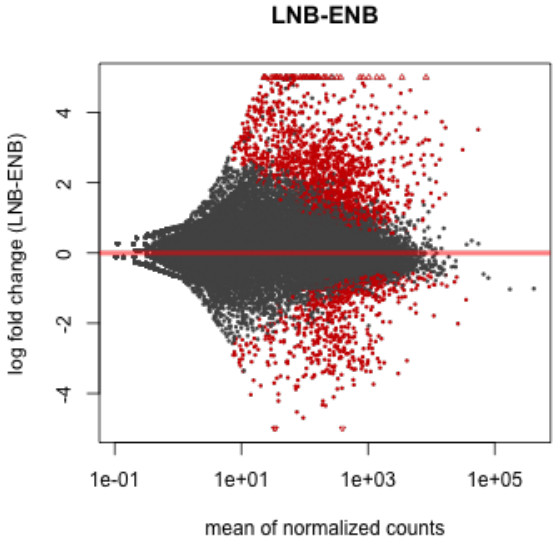
The size factors are plotted.

```
p = par(mar=c(10,4,4,2)+0.1)
barplot(sizeFactors(dds), main="estimates size factors",las=2)
par(p)
```



An MA plot shows in red the genes that are enriched by the RiboTag pull-down. For comparison, the second plot shows the genes that were used to estimate the size factors.

```
plotMA(res, ylim=c(-5,5), ylab=sprintf("log fold change (%s)",n), main=n)
```




```
load(file.path("data","ENSGannotation.rda"))
res$symbol = ENSGannotation[row.names(res),"name"]
res$symbol[is.na(res$symbol)] = row.names(res)[is.na(res$symbol)]

res$significant = ""
res$significant[which(res$padj <= 0.1)] = "*"
res$significant[which(res$padj <= 0.01)] = "**"
```

The data is saved.

```
save(dds, file=file.path(d, "dds.rda"))
save(res, file=file.path(d, "res.rda"))
```

```
col=rep("#00000022", nrow(res))
col[which(res$padj <= 0.1 & res$log2FoldChange > 0)] = brewer.pal(4,"RdBu")[2]
col[which(res$padj <= 0.1 & res$log2FoldChange < 0)] = brewer.pal(4,"RdBu")[3]
col[which(res$padj <= 0.01 & res$log2FoldChange > 0)] = brewer.pal(4,"RdBu")[1]
col[which(res$padj <= 0.01 & res$log2FoldChange < 0)] = brewer.pal(4,"RdBu")[4]

I = order(factor(col, levels=c("#00000022",brewer.pal(4,"RdBu")[c(2,3,1,4)])))
lim = range(as.vector(C))
pairs(C[I,],pch=20,cex=0.5,xlim=lim, ylim=lim,col=col[I])
```


```
fgup = row.names(res)[which(res$significant != "" & res$log2FoldChange > 0)]
fgdown = row.names(res)[which(res$significant != "" & res$log2FoldChange < 0)]
bg = row.names(res)[which(!is.na(res$padj))]
GSEAup = doGSEA(fgup,bg,ENSG2category, k=3)
GSEAdown = doGSEA(fgdown,bg,ENSG2category, k=3)
```

```
write.table(res, file=file.path(d, "resulttable.txt"), sep="\t", quote=FALSE)
page = openPage(file.path(d,"resulttable.html"), link.css="hwriter.css")
hwriteSidebar(page=page, link=mainmenu)
hwrite(sprintf("Differential gene expression (%s;%s); result table",type,n),
       heading=1, page=page)
hwrite("** = 1% FDR; * = 10% FDR",page=page,br=TRUE)
col = rep(NA, nrow(res))
col[which(res$significant != "" & res$log2FoldChange > 0)] = "#fbb4ae"
col[which(res$significant != "" & res$log2FoldChange < 0)] = "#b3cde3"
hwrite(as.data.frame(res), row.bgcolor=col, page=page)
closePage(page, splash=FALSE)
```

1.3 Differential gene expression in NEURON-LNB

Create an output directory.

```
n = "NEURON-LNB"
type="RNAseq"
Samples = SamplesRNAseq
filter = "none"
d = file.path("result",type, n)
dir.create(d, recursive = TRUE, showWarnings = FALSE)
mainmenu = file.path("..","..","..","mainmenu.html")

load(file.path("data","ENSG2category.rda"))
source(file.path("R","doGSEA.R"))
```

The count matrix is copied for convenience.

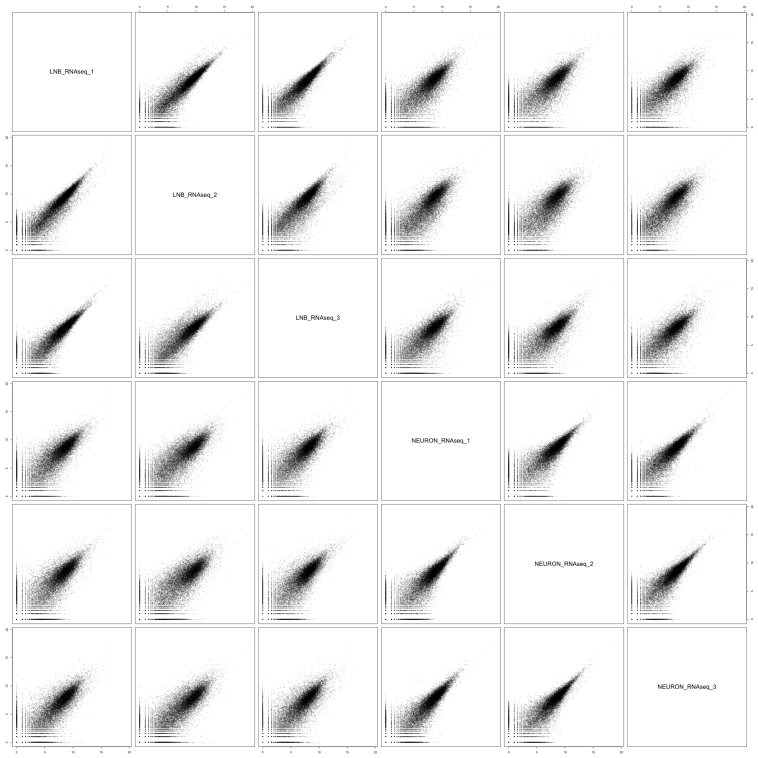
```
X = rawCounts[,Samples[[n]]]
Design = DesignRNAseq
```

A DESeq dataset is created that contains the respective samples.

```
dds <- DESeqDataSetFromMatrix(countData = X,
                              colData = SampleList[Samples[[n]],],
                              design = Design[[n]])

## converting counts to integer mode

## factor levels were dropped which had no samples

colData(dds)

## DataFrame with 6 rows and 6 columns
##                              Name Population     Type Replicate     Comment
##                       <character>   <factor> <factor>  <factor> <character>
## LNB_RNAseq_1       LNB_RNAseq_1        LNB   RNAseq        NA
## LNB_RNAseq_2       LNB_RNAseq_2        LNB   RNAseq        NA
## LNB_RNAseq_3       LNB_RNAseq_3        LNB   RNAseq        NA
## NEURON_RNAseq_1 NEURON_RNAseq_1     NEURON   RNAseq        NA
## NEURON_RNAseq_2 NEURON_RNAseq_2     NEURON   RNAseq        NA
## NEURON_RNAseq_3 NEURON_RNAseq_3     NEURON   RNAseq        NA
##                          OriginalName
##                           <character>
## LNB_RNAseq_1        10_DicryO6364YFP
## LNB_RNAseq_2    18_DicryO8355_8492YF
## LNB_RNAseq_3        11_DicryO6365YFP
## NEURON_RNAseq_1 09_Dxc4wpi_7490total
## NEURON_RNAseq_2 10_Dxc4wpi_7652total
## NEURON_RNAseq_3 11_Dxc4wpi_8047total
```

Log counts are computed and plotted as a pairs plot.

```
C = counts(dds)
C = log2(C+1)
lim = range(as.vector(C))
pairs(C,pch=20,cex=0.5,xlim=lim, ylim=lim,col="#00000022")
```



The standard DESeq2 workflow is used to for differential gene expression analysis. size factors are estimated using all genes. Subsequently, dispersion is estimated and p-values are computed by a Wald test.

```
dds = DESeq(dds)
```

```
## estimating size factors
```

```
## estimating dispersions
```

```
## gene-wise dispersion estimates
```

```
## mean-dispersion relationship
```

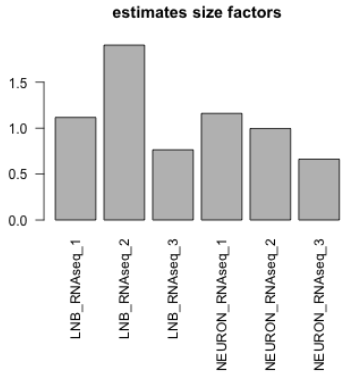
```
## final dispersion estimates
```

```
## fitting model and testing
```

```
res = results(dds)
```

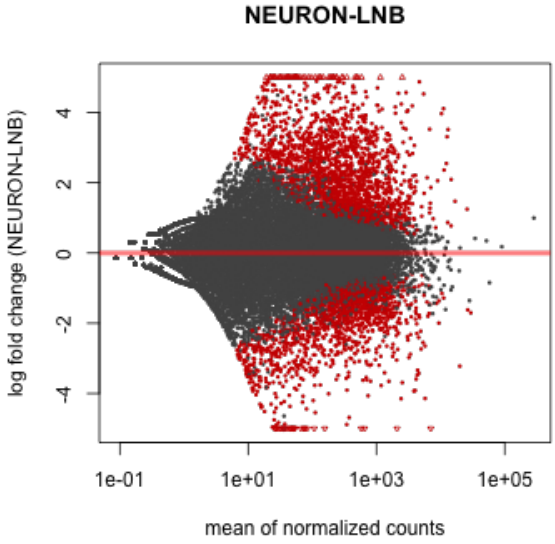
The size factors are plotted.

```
p = par(mar=c(10,4,4,2)+0.1)
barplot(sizeFactors(dds), main="estimates size factors",las=2)
par(p)
```



An MA plot shows in red the genes that are enriched by the RiboTag pull-down. For comparison, the second plot shows the genes that were used to estimate the size factors.

```
plotMA(res, ylim=c(-5,5), ylab=sprintf("log fold change (%s)",n), main=n)
```




```
load(file.path("data","ENSGannotation.rda"))
res$symbol = ENSGannotation[row.names(res),"name"]
res$symbol[is.na(res$symbol)] = row.names(res)[is.na(res$symbol)]

res$significant = ""
res$significant[which(res$padj <= 0.1)] = "*"
res$significant[which(res$padj <= 0.01)] = "**"
```

The data is saved.

```
save(dds, file=file.path(d, "dds.rda"))
save(res, file=file.path(d, "res.rda"))
```

```
col=rep("#00000022", nrow(res))
col[which(res$padj <= 0.1 & res$log2FoldChange > 0)] = brewer.pal(4,"RdBu")[2]
col[which(res$padj <= 0.1 & res$log2FoldChange < 0)] = brewer.pal(4,"RdBu")[3]
col[which(res$padj <= 0.01 & res$log2FoldChange > 0)] = brewer.pal(4,"RdBu")[1]
col[which(res$padj <= 0.01 & res$log2FoldChange < 0)] = brewer.pal(4,"RdBu")[4]

I = order(factor(col, levels=c("#00000022",brewer.pal(4,"RdBu")[c(2,3,1,4)])))
lim = range(as.vector(C))
pairs(C[I,],pch=20,cex=0.5,xlim=lim, ylim=lim,col=col[I])
```


```
fgup = row.names(res)[which(res$significant != "" & res$log2FoldChange > 0)]
fgdown = row.names(res)[which(res$significant != "" & res$log2FoldChange < 0)]
bg = row.names(res)[which(!is.na(res$padj))]
GSEAup = doGSEA(fgup,bg,ENSG2category, k=3)
GSEAdown = doGSEA(fgdown,bg,ENSG2category, k=3)
```

```
write.table(res, file=file.path(d, "resulttable.txt"), sep="\t", quote=FALSE)
page = openPage(file.path(d,"resulttable.html"), link.css="hwriter.css")
hwriteSidebar(page=page, link=mainmenu)
hwrite(sprintf("Differential gene expression (%s;%s); result table",type,n),
       heading=1, page=page)
hwrite("** = 1% FDR; * = 10% FDR",page=page,br=TRUE)
col = rep(NA, nrow(res))
col[which(res$significant != "" & res$log2FoldChange > 0)] = "#fbb4ae"
col[which(res$significant != "" & res$log2FoldChange < 0)] = "#b3cde3"
hwrite(as.data.frame(res), row.bgcolor=col, page=page)
closePage(page, splash=FALSE)
```

2 Differential ribosome signal

Sample indices for five different pairs of ribosome tag datasets are extracted.

```
SamplesRIBO_IPseq = list(
  "ENB-NSC" = which(SampleList$Population %in% c("NSC","ENB") &
                    SampleList$Type == "RIBO_IPseq"),
  "LNB-ENB" = which(SampleList$Population %in% c("ENB","LNB") &
                    SampleList$Type == "RIBO_IPseq"),
  "NEURON-LNB" = which(SampleList$Population %in% c("LNB","NEURON") &
                    SampleList$Type == "RIBO_IPseq"))

DesignRIBOseq = list(
  "ENB-NSC" = as.formula("~Population"),
```

```
  "LNB-ENB" = as.formula("~Population"),
  "NEURON-LNB" = as.formula("~Population"))

for (n in names(DesignRIBOseq)) {
  SamplesRIBO_IPseq[[n]] = SamplesRIBO_IPseq[[n]][
    order(SampleList[SamplesRIBO_IPseq[[n]],as.character(DesignRIBOseq[[n]])[2]],
          SampleList[SamplesRIBO_IPseq[[n]],"Replicate"])]
}
```

Differential gene expression analysis is performed for each of these comparisons.

2.1 Differential ribosome tag signal in ENB-NSC

Create an output directory.

```
n = "ENB-NSC"
type="RIBO_IPseq"
Samples = SamplesRIBO_IPseq
filter = "none"
d = file.path("result",type, n)
dir.create(d, recursive = TRUE, showWarnings = FALSE)
mainmenu = file.path("..","..","..","mainmenu.html")

load(file.path("data","ENSG2category.rda"))
source(file.path("R","doGSEA.R"))
```

The count matrix is copied for convenience.

```
Design = DesignRIBOseq
X = rawCounts[,Samples[[n]]]
```

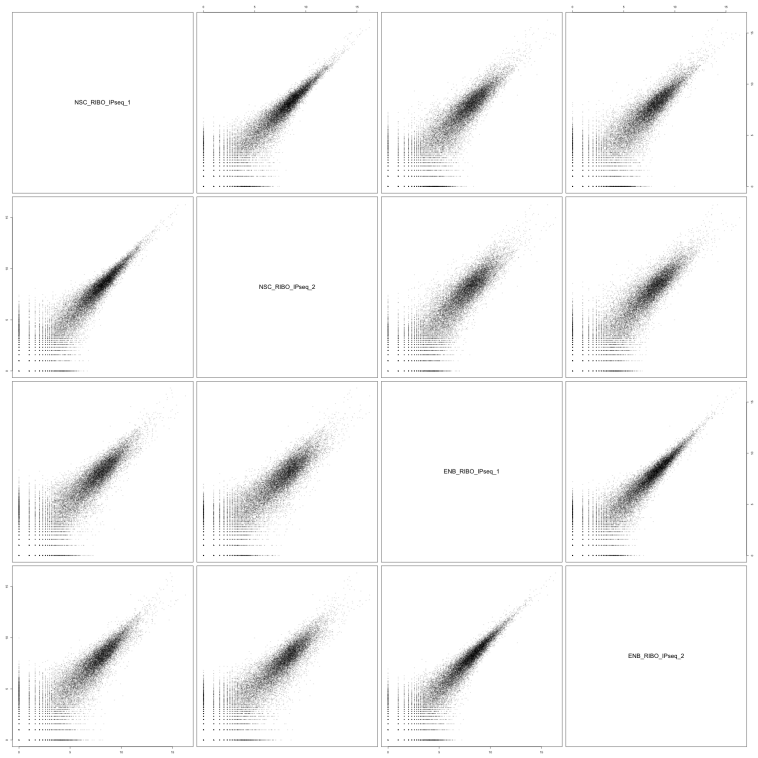
A DESeq dataset is created that contains the respective samples.

```
dds <- DESeqDataSetFromMatrix(countData = X,
                              colData = SampleList[Samples[[n]],],
                              design = Design[[n]])

## converting counts to integer mode

## factor levels were dropped which had no samples

colData(dds)

## DataFrame with 4 rows and 6 columns
##                               Name Population       Type Replicate     Comment
##                        <character>   <factor>   <factor>  <factor> <character>
## NSC_RIBO_IPseq_1 NSC_RIBO_IPseq_1        NSC RIBO_IPseq        NA
## NSC_RIBO_IPseq_2 NSC_RIBO_IPseq_2        NSC RIBO_IPseq        NA
## ENB_RIBO_IPseq_1 ENB_RIBO_IPseq_1        ENB RIBO_IPseq        NA
## ENB_RIBO_IPseq_2 ENB_RIBO_IPseq_2        ENB RIBO_IPseq        NA
##                              OriginalName
##                               <character>
## NSC_RIBO_IPseq_1 10_SVZ_TICRY4d_4632_plus_
## NSC_RIBO_IPseq_2 12_SVZ_TICRY4d_4943_plus_
## ENB_RIBO_IPseq_1 6_SVZ_DICRY3dp_5516_plus_
## ENB_RIBO_IPseq_2 9_SVZ_DICRY3dp_5519_plus_
```

Log counts are computed and plotted as a pairs plot.

```
C = counts(dds)
C = log2(C+1)
lim = range(as.vector(C))
```

```
pairs(C,pch=20,cex=0.5,xlim=lim, ylim=lim,col="#00000022")
```



The standard DESeq2 workflow is used to for analysis differential ribosome tag signal. Size factors are estimated using all genes. Subsequently, dispersion is estimated and p-values are computed by a Wald test.

```
dds = DESeq(dds)
```

```
## estimating size factors
```

```
## estimating dispersions
```

```
## gene-wise dispersion estimates
```

```
## mean-dispersion relationship
```

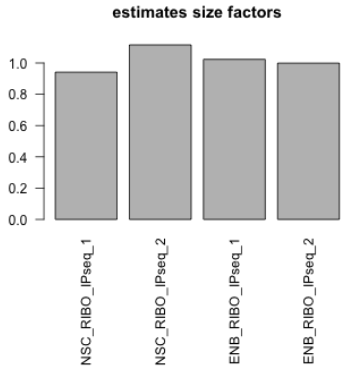
```
## final dispersion estimates
```

```
## fitting model and testing
```

```
res = results(dds)
```

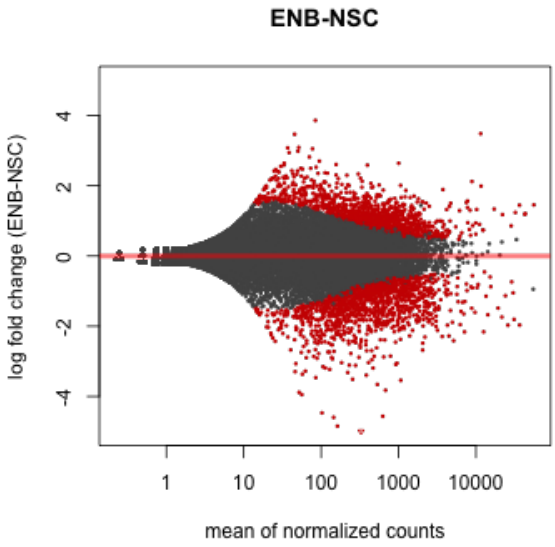
The size factors are plotted.

```
p = par(mar=c(10,4,4,2)+0.1)
barplot(sizeFactors(dds), main="estimates size factors",las=2)
par(p)
```



An MA plot shows in red the genes that are enriched by the RiboTag pull-down. For comparison, the second plot shows the genes that were used to estimate the size factors.

```
plotMA(res, ylim=c(-5,5), ylab=sprintf("log fold change (%s)",n), main=n)
```




```
load(file.path("data","ENSGannotation.rda"))
res$symbol = ENSGannotation[row.names(res),"name"]
res$symbol[is.na(res$symbol)] = row.names(res)[is.na(res$symbol)]

res$significant = ""
res$significant[which(res$padj <= 0.1)] = "*"
res$significant[which(res$padj <= 0.01)] = "**"
```

The data is saved.

```
save(dds, file=file.path(d, "dds.rda"))
save(res, file=file.path(d, "res.rda"))
```

```
col=rep("#00000022", nrow(res))
col[which(res$padj <= 0.1 & res$log2FoldChange > 0)] = brewer.pal(4,"RdBu")[2]
col[which(res$padj <= 0.1 & res$log2FoldChange < 0)] = brewer.pal(4,"RdBu")[3]
col[which(res$padj <= 0.01 & res$log2FoldChange > 0)] = brewer.pal(4,"RdBu")[1]
col[which(res$padj <= 0.01 & res$log2FoldChange < 0)] = brewer.pal(4,"RdBu")[4]

I = order(factor(col, levels=c("#00000022",brewer.pal(4,"RdBu")[c(2,3,1,4)])))
lim = range(as.vector(C))
pairs(C[I,],pch=20,cex=0.5,xlim=lim, ylim=lim,col=col[I])
```


```
fgup = row.names(res)[which(res$significant != "" & res$log2FoldChange > 0)]
fgdown = row.names(res)[which(res$significant != "" & res$log2FoldChange < 0)]
bg = row.names(res)[which(!is.na(res$padj))]
GSEAup = doGSEA(fgup,bg,ENSG2category, k=3)
GSEAdown = doGSEA(fgdown,bg,ENSG2category, k=3)
```

```
write.table(res, file=file.path(d, "resulttable.txt"), sep="\t", quote=FALSE)
page = openPage(file.path(d,"resulttable.html"), link.css="hwriter.css")
hwriteSidebar(page=page, link=mainmenu)
hwrite(sprintf("Differential ribosome signal (%s;%s); result table",type,n),
       heading=1, page=page)
hwrite("** = 1% FDR; * = 10% FDR",page=page,br=TRUE)
col = rep(NA, nrow(res))
col[which(res$significant != "" & res$log2FoldChange > 0)] = "#fbb4ae"
col[which(res$significant != "" & res$log2FoldChange < 0)] = "#b3cde3"
hwrite(as.data.frame(res), row.bgcolor=col, page=page)
closePage(page, splash=FALSE)
```

2.2 Differential ribosome tag signal in LNB-ENB

Create an output directory.

```
n = "LNB-ENB"
type="RIBO_IPseq"
Samples = SamplesRIBO_IPseq
filter = "none"
d = file.path("result",type, n)
dir.create(d, recursive = TRUE, showWarnings = FALSE)
mainmenu = file.path("..","..","..","mainmenu.html")

load(file.path("data","ENSG2category.rda"))
source(file.path("R","doGSEA.R"))
```

The count matrix is copied for convenience.

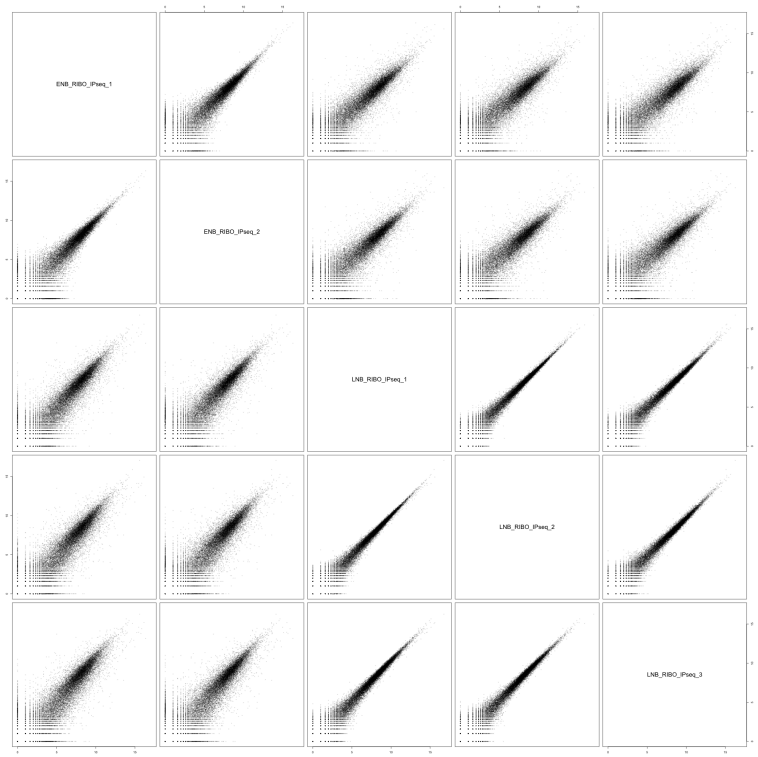
```
Design = DesignRIBOseq
X = rawCounts[,Samples[[n]]]
```

A DESeq dataset is created that contains the respective samples.

```
dds <- DESeqDataSetFromMatrix(countData = X,
                              colData = SampleList[Samples[[n]],],
                              design = Design[[n]])

## converting counts to integer mode

## factor levels were dropped which had no samples

colData(dds)

## DataFrame with 5 rows and 6 columns
##                              Name Population       Type Replicate     Comment
##                        <character>   <factor>   <factor>  <factor> <character>
## ENB_RIBO_IPseq_1 ENB_RIBO_IPseq_1        ENB RIBO_IPseq        NA
## ENB_RIBO_IPseq_2 ENB_RIBO_IPseq_2        ENB RIBO_IPseq        NA
## LNB_RIBO_IPseq_1 LNB_RIBO_IPseq_1        LNB RIBO_IPseq        NA
## LNB_RIBO_IPseq_2 LNB_RIBO_IPseq_2        LNB RIBO_IPseq        NA
## LNB_RIBO_IPseq_3 LNB_RIBO_IPseq_3        LNB RIBO_IPseq        NA
##                               OriginalName
##                                <character>
## ENB_RIBO_IPseq_1 6_SVZ_DICRY3dp_5516_plus_
## ENB_RIBO_IPseq_2 9_SVZ_DICRY3dp_5519_plus_
## LNB_RIBO_IPseq_1 1_OB_DICRY3dpi_5516_plus_
## LNB_RIBO_IPseq_2 4_OB_DICRY3dpi_5519_plus_
## LNB_RIBO_IPseq_3 5_OB_DICRY3dpi_5524_plus_
```

Log counts are computed and plotted as a pairs plot.

```
C = counts(dds)
C = log2(C+1)
lim = range(as.vector(C))
pairs(C,pch=20,cex=0.5,xlim=lim, ylim=lim,col="#00000022")
```



The standard DESeq2 workflow is used to for analysis differential ribosome tag signal. Size factors are estimated using all genes. Subsequently, dispersion is estimated and p-values are computed by a Wald test.

```
dds = DESeq(dds)
```

```
## estimating size factors
```

```
## estimating dispersions
```

```
## gene-wise dispersion estimates
```

```
## mean-dispersion relationship
```

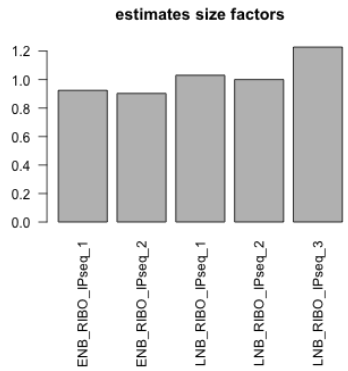
```
## final dispersion estimates
```

```
## fitting model and testing
```

```
res = results(dds)
```

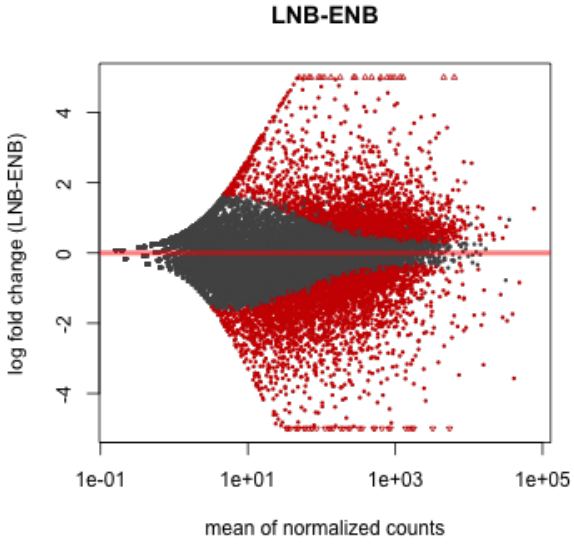
The size factors are plotted.

```
p = par(mar=c(10,4,4,2)+0.1)
barplot(sizeFactors(dds), main="estimates size factors",las=2)
par(p)
```



An MA plot shows in red the genes that are enriched by the RiboTag pull-down. For comparison, the second plot shows the genes that were used to estimate the size factors.

```
plotMA(res, ylim=c(-5,5), ylab=sprintf("log fold change (%s)",n), main=n)
```

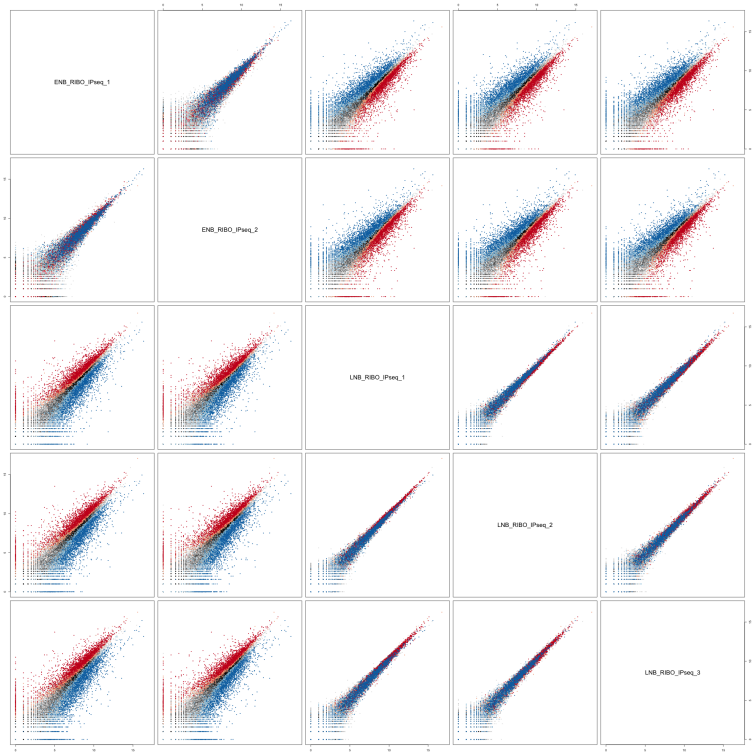



```
load(file.path("data","ENSGannotation.rda"))
res$symbol = ENSGannotation[row.names(res),"name"]
res$symbol[is.na(res$symbol)] = row.names(res)[is.na(res$symbol)]

res$significant = ""
res$significant[which(res$padj <= 0.1)] = "*"
res$significant[which(res$padj <= 0.01)] = "**"
```

The data is saved.

```
save(dds, file=file.path(d, "dds.rda"))
save(res, file=file.path(d, "res.rda"))
```

```
col=rep("#00000022", nrow(res))
col[which(res$padj <= 0.1 & res$log2FoldChange > 0)] = brewer.pal(4,"RdBu")[2]
col[which(res$padj <= 0.1 & res$log2FoldChange < 0)] = brewer.pal(4,"RdBu")[3]
col[which(res$padj <= 0.01 & res$log2FoldChange > 0)] = brewer.pal(4,"RdBu")[1]
col[which(res$padj <= 0.01 & res$log2FoldChange < 0)] = brewer.pal(4,"RdBu")[4]

I = order(factor(col, levels=c("#00000022",brewer.pal(4,"RdBu")[c(2,3,1,4)])))
lim = range(as.vector(C))
pairs(C[I,],pch=20,cex=0.5,xlim=lim, ylim=lim,col=col[I])
```




```
fgup = row.names(res)[which(res$significant != "" & res$log2FoldChange > 0)]
fgdown = row.names(res)[which(res$significant != "" & res$log2FoldChange < 0)]
bg = row.names(res)[which(!is.na(res$padj))]
GSEAup = doGSEA(fgup,bg,ENSG2category, k=3)
GSEAdown = doGSEA(fgdown,bg,ENSG2category, k=3)
```

```
write.table(res, file=file.path(d, "resulttable.txt"), sep="\t", quote=FALSE)
page = openPage(file.path(d,"resulttable.html"), link.css="hwriter.css")
hwriteSidebar(page=page, link=mainmenu)
hwrite(sprintf("Differential ribosome signal (%s;%s); result table",type,n),
       heading=1, page=page)
hwrite("** = 1% FDR; * = 10% FDR",page=page,br=TRUE)
col = rep(NA, nrow(res))
col[which(res$significant != "" & res$log2FoldChange > 0)] = "#fbb4ae"
col[which(res$significant != "" & res$log2FoldChange < 0)] = "#b3cde3"
hwrite(as.data.frame(res), row.bgcolor=col, page=page)
closePage(page, splash=FALSE)
```

2.3 Differential ribosome tag signal in NEURON-LNB

Create an output directory.

```
n = "NEURON-LNB"
type="RIBO_IPseq"
Samples = SamplesRIBO_IPseq
filter = "none"
d = file.path("result",type, n)
dir.create(d, recursive = TRUE, showWarnings = FALSE)
mainmenu = file.path("..","..","..","mainmenu.html")

load(file.path("data","ENSG2category.rda"))
source(file.path("R","doGSEA.R"))
```

The count matrix is copied for convenience.

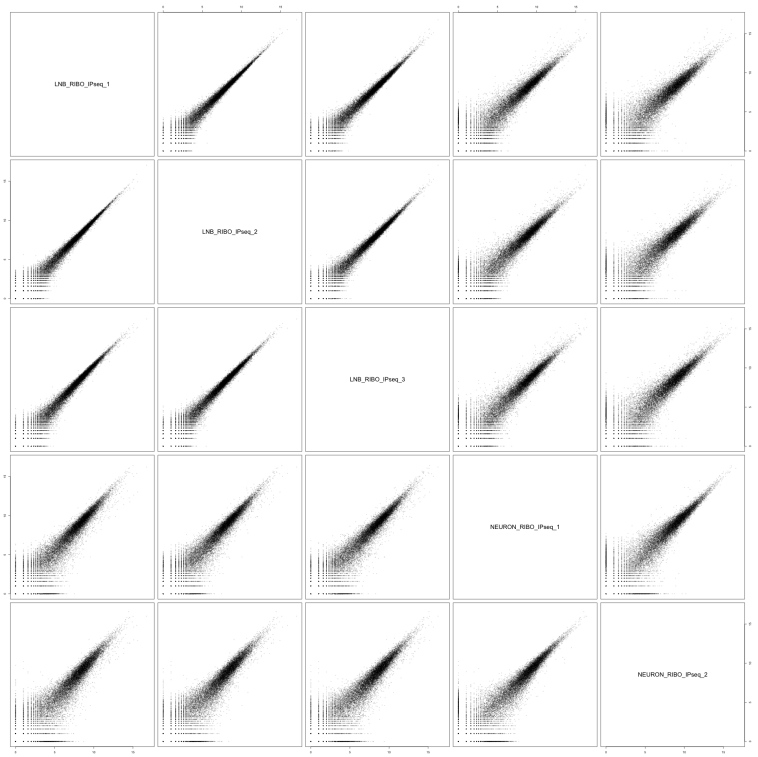
```
Design = DesignRIBOseq
X = rawCounts[,Samples[[n]]]
```

A DESeq dataset is created that contains the respective samples.

```
dds <- DESeqDataSetFromMatrix(countData = X,
                              colData = SampleList[Samples[[n]],],
                              design = Design[[n]])

## converting counts to integer mode

## factor levels were dropped which had no samples

colData(dds)

## DataFrame with 5 rows and 6 columns
##                                   Name Population       Type Replicate       Comment
##                               <character>   <factor>   <factor>  <factor>   <character>
## LNB_RIBO_IPseq_1       LNB_RIBO_IPseq_1        LNB RIBO_IPseq        NA
## LNB_RIBO_IPseq_2       LNB_RIBO_IPseq_2        LNB RIBO_IPseq        NA
## LNB_RIBO_IPseq_3       LNB_RIBO_IPseq_3        LNB RIBO_IPseq        NA
## NEURON_RIBO_IPseq_1 NEURON_RIBO_IPseq_1     NEURON RIBO_IPseq        NA
## NEURON_RIBO_IPseq_2 NEURON_RIBO_IPseq_2     NEURON RIBO_IPseq        NA higher 3 bias
##                                  OriginalName
##                                   <character>
## LNB_RIBO_IPseq_1    1_OB_DICRY3dpi_5516_plus_
## LNB_RIBO_IPseq_2    4_OB_DICRY3dpi_5519_plus_
## LNB_RIBO_IPseq_3    5_OB_DICRY3dpi_5524_plus_
## NEURON_RIBO_IPseq_1 05_Dxc4wpi_7155_plus_Ribo
## NEURON_RIBO_IPseq_2 07_Dxc4wpi_7154_plus_Ribo
```

Log counts are computed and plotted as a pairs plot.

```
C = counts(dds)
C = log2(C+1)
lim = range(as.vector(C))
pairs(C,pch=20,cex=0.5,xlim=lim, ylim=lim,col="#00000022")
```



The standard DESeq2 workflow is used to for analysis differential ribosome tag signal. Size factors are estimated using all genes. Subsequently, dispersion is estimated and p-values are computed by a Wald test.

```
dds = DESeq(dds)
```

```
## estimating size factors
```

```
## estimating dispersions
```

```
## gene-wise dispersion estimates
```

```
## mean-dispersion relationship
```

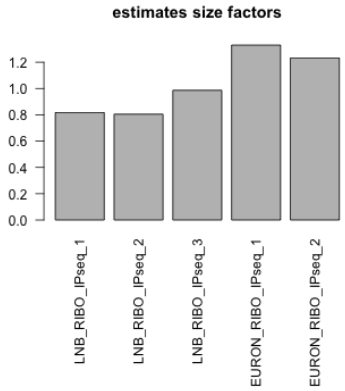
```
## final dispersion estimates
```

```
## fitting model and testing
```

```
res = results(dds)
```

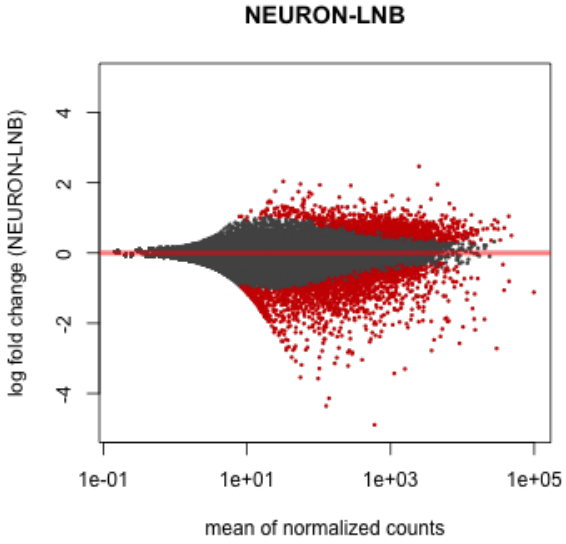
The size factors are plotted.

```
p = par(mar=c(10,4,4,2)+0.1)
barplot(sizeFactors(dds), main="estimates size factors",las=2)
par(p)
```



An MA plot shows in red the genes that are enriched by the RiboTag pull-down. For comparison, the second plot shows the genes that were used to estimate the size factors.

```
plotMA(res, ylim=c(-5,5), ylab=sprintf("log fold change (%s)",n), main=n)
```




```
load(file.path("data","ENSGannotation.rda"))
res$symbol = ENSGannotation[row.names(res),"name"]
res$symbol[is.na(res$symbol)] = row.names(res)[is.na(res$symbol)]

res$significant = ""
res$significant[which(res$padj <= 0.1)] = "*"
res$significant[which(res$padj <= 0.01)] = "**"
```

The data is saved.

```
save(dds, file=file.path(d, "dds.rda"))
save(res, file=file.path(d, "res.rda"))
```

```
col=rep("#00000022", nrow(res))
col[which(res$padj <= 0.1 & res$log2FoldChange > 0)] = brewer.pal(4,"RdBu")[2]
col[which(res$padj <= 0.1 & res$log2FoldChange < 0)] = brewer.pal(4,"RdBu")[3]
col[which(res$padj <= 0.01 & res$log2FoldChange > 0)] = brewer.pal(4,"RdBu")[1]
col[which(res$padj <= 0.01 & res$log2FoldChange < 0)] = brewer.pal(4,"RdBu")[4]

I = order(factor(col, levels=c("#00000022",brewer.pal(4,"RdBu")[c(2,3,1,4)])))
lim = range(as.vector(C))
pairs(C[I,],pch=20,cex=0.5,xlim=lim, ylim=lim,col=col[I])
```


```
fgup = row.names(res)[which(res$significant != "" & res$log2FoldChange > 0)]
fgdown = row.names(res)[which(res$significant != "" & res$log2FoldChange < 0)]
bg = row.names(res)[which(!is.na(res$padj))]
GSEAup = doGSEA(fgup,bg,ENSG2category, k=3)
GSEAdown = doGSEA(fgdown,bg,ENSG2category, k=3)
```

```
write.table(res, file=file.path(d, "resulttable.txt"), sep="\t", quote=FALSE)
page = openPage(file.path(d,"resulttable.html"), link.css="hwriter.css")
hwriteSidebar(page=page, link=mainmenu)
hwrite(sprintf("Differential ribosome signal (%s;%s); result table",type,n),
       heading=1, page=page)
hwrite("** = 1% FDR; * = 10% FDR",page=page,br=TRUE)
col = rep(NA, nrow(res))
col[which(res$significant != "" & res$log2FoldChange > 0)] = "#fbb4ae"
col[which(res$significant != "" & res$log2FoldChange < 0)] = "#b3cde3"
hwrite(as.data.frame(res), row.bgcolor=col, page=page)
closePage(page, splash=FALSE)
```

3 Translation Efficiency

```
SamplesTranslationEfficiency = list(
  "NSC" = which(SampleList$Population %in% c("NSC", "NSC/ENB")),
  "ENB" = which(SampleList$Population %in% c("ENB", "NSC/ENB")),
  "LNB" = which(SampleList$Population %in% c("LNB","LNB/NEURON")),
  "NEURON" = which(SampleList$Population %in% c("NEURON","LNB/NEURON")) )

DesignTranslationEfficiency = as.formula("~Type")

for (n in names(SamplesTranslationEfficiency)) {
  SamplesTranslationEfficiency[[n]] = SamplesTranslationEfficiency[[n]][
    order(SampleList[SamplesTranslationEfficiency[[n]],"Type"],
          SampleList[SamplesTranslationEfficiency[[n]],"Replicate"])]
```

```
}
```

Translation efficiency is estimated for high abundand genes.

3.1 Filter unspecific binders in NSC

Create an output directory.

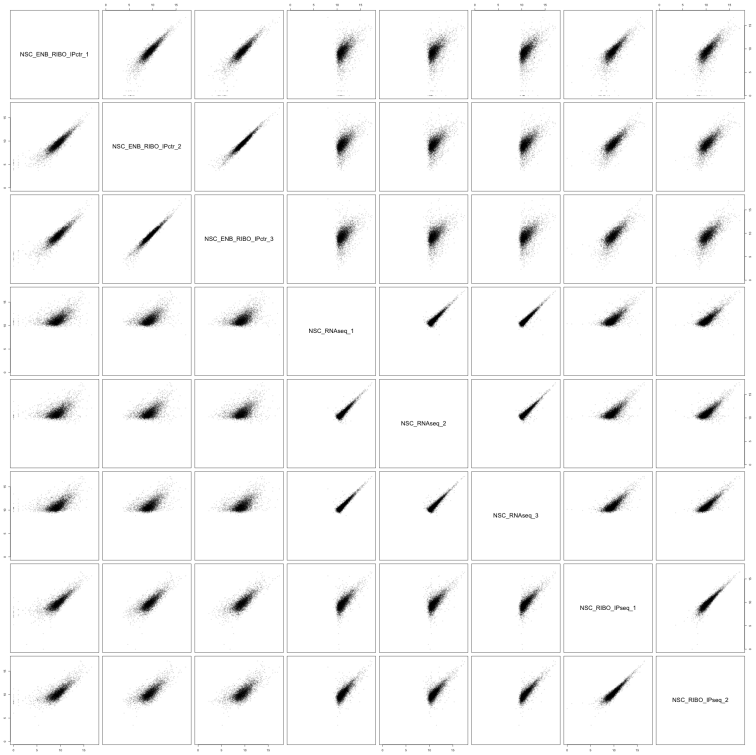
```
n = "NSC"
d = file.path("result","TranslationEfficiency", n)
dir.create(d, recursive = TRUE, showWarnings = FALSE)
mainmenu = file.path("..","..","..","mainmenu.html")
source(file.path("R","doGSEA.R"))
```

A DESeq dataset is created that contains the respective samples.

```
Design = DesignTranslationEfficiency
EC_all <- expectedCounts[,SamplesTranslationEfficiency[[n]]]
SL <- SampleList[SamplesTranslationEfficiency[[n]],]
# dds_all <- DESeqDataSetFromMatrix(
#     countData = expectedCounts[,SamplesTranslationEfficiency[[n]]],
#     colData = SampleList[SamplesTranslationEfficiency[[n]],],
#     design = ~ Type)
# mcols(dds_all) = Anno
# colData(dds_all)
```

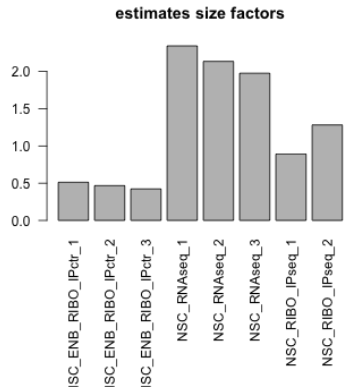
Log counts are computed and plotted as a pairs plot.

```
I = which(rowMeans(log2(EC_all[,
            SL$Type == "RNAseq"]+1)) >= 10 &
            Anno$transript_type == "protein_coding")
EC = EC_all[I,]
# I = which(rowMeans(log2(counts(dds_all)[,
#             colData(dds_all)£Type == "RNAseq"]+1)) >= 10 &
#             mcols(dds_all)£transript_type == "protein_coding")
# dds = dds_all[I,]

C = log2(EC+1)
Call = log2(EC_all+1)
# C = log2(counts(dds)+1)
# Call = log2(counts(dds_all)+1)
lim = range(as.vector(C))
pairs(C,pch=20,cex=0.5,col="#00000022",xlim=lim, ylim=lim)
```



The genes in the cluster with lower read counts in the control sample are used to estimate the size factors.

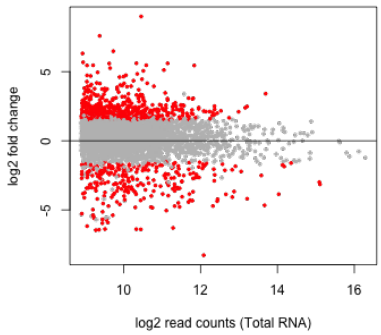
```
s = sizeFactors(estimateSizeFactors(DESeqDataSetFromMatrix(
    countData = round(EC), colData = SL, design = ~ Type),
    controlGenes=which(rowMeans(log2(EC[,SL$Type == "RNAseq"]+1)) >= 13)))

## converting counts to integer mode

#sizeFactors(dds) = s
p = par(mar=c(10,4,4,2)+0.1)
barplot(s, main="estimates size factors",las=2)
par(p)
```



The count data is normalized and unspecific binders are identified. Therefore, RiboControl samples are compared to total RNA samples. The genes that are significantly higher in RiboControl samples are potential unspecific binders of the tag. These genes are filtered as well.

```
D = C
m = log2(s)
for (i in 1:ncol(EC)) {
  D[,i] = D[,i] - m[i]
}
```

```
Dall = Call
for (i in 1:ncol(EC)) {
  Dall[,i] = Dall[,i] - m[i]
}

IR = which(SL$Type == "RIBO_IPseq")
IT = which(SL$Type == "RNAseq")
IC = which(SL$Type == "RIBO_IPctr")

fitc = eBayes(lmFit(cbind(D[,IC[1]]-D[,IT[1]],D[,IC[2]]-D[,IT[2]])))
padjc = p.adjust(fitc$p.value[,1],method="BH")
lfcc = fitc$coefficients[,1]

plot(rowMeans(D[,SL$Type == "RNAseq"]),lfcc,
     pch=20,cex=0.7,col=ifelse(padjc < 0.01, "red","gray"),
     xlab="log2 read counts (Total RNA)", ylab="log2 fold change")
abline(h=0)

UnspecBinders = which(padjc <= 0.01 & lfcc > 0)

fitc$padj = padjc
```



To estimate the translation efficiency, the RiboTag data is modeled as a function of RiboControl and total RNA.

```
fit1 = list()
for (i in 1:2) {
  D2 = data.frame("Ribo" = D[,IR[i]], "Total" = D[,IT[i]], "ctrl" = D[,IC[i]])
  fit1[[i]] = lm(Ribo ~ Total+ctrl, data=D2)
  fit1[[i]]$anova = anova(fit1[[i]])
  fit1[[i]]$expVar = fit1[[i]]$anova["Sum Sq",drop=FALSE]
  fit1[[i]]$expVar = fit1[[i]]$expVar/sum(fit1[[i]]$expVar)
  D2all = data.frame("Ribo" = Dall[,IR[i]], "Total" = Dall[,IT[i]], "ctrl" = Dall[,IC[i]], row.names =
  p = predict(fit1[[i]],newdata = D2all)
  fit1[[i]]$residualsAll = D2all$Ribo - p
}

R1 = fit1[[1]]$residuals
R2 = fit1[[2]]$residuals
R1all = fit1[[1]]$residualsAll
R2all = fit1[[2]]$residualsAll
R1all = R1all - shorth(R1,na.rm=TRUE)
R2all = R2all - shorth(R2,na.rm=TRUE)
R1 = R1 - shorth(R1,na.rm=TRUE)
R2 = R2 - shorth(R2,na.rm=TRUE)
```

```
D2 = data.frame("R1" = R1, "R2" = R2)
fit2 = lm(data=D2)
fit2$anova = anova(fit2)
fit2$expVar = fit2$anova[,"Sum Sq",drop=FALSE]
fit2$expVar = fit2$expVar/sum(fit2$expVar)

fita = eBayes(lmFit(cbind(R1,R2)))
padja = p.adjust(fita$p.value[,1],method="BH")
lfca = fita$coefficients[,1]
```

```
ExpVar = t(as.matrix(cbind(rep1=fit1[[1]]$expVar,rep2=fit1[[2]]$expVar)))
ExpVar = cbind(ExpVar[,1:2],ExpVar[,3] * fit2$expVar[1,1], ExpVar[,3] * fit2$expVar[2,1])
colnames(ExpVar) = c("total RNA", "background", "transl. eff.", "residuals")
row.names(ExpVar) = paste0("rep ",1:2)
# barplot(ExpVar,beside=TRUE,
#         legend=TRUE,ylab="explained variance")
# barplot(t(ExpVar),beside=FALSE,
#         legend=TRUE,ylab="explained variance",xlim=c(0,4))
```

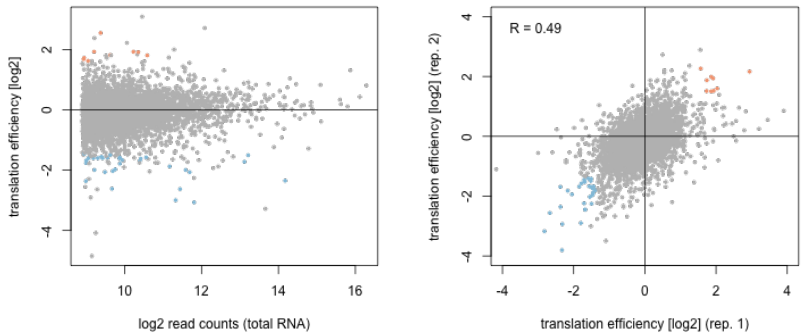
```
res = as.data.frame(Anno[row.names(EC),])
#res$symbol[is.na(res$symbol)] = row.names(res)[is.na(res$symbol)]

res$pval = fita$p.value[,1]
res$padj = padja
res$log2FoldChange = lfca
res$significant = ""
res$significant[which(padja <= 0.1)] = "*"
res$significant[which(padja <= 0.01)] = "**"
attr(res,"log2FC") = rowMeans(cbind(R1all,R2all))
attr(res,"log2total") = rowMeans(Dall[,IT])
attr(res,"log2FCsd") = rowSds(cbind(R1all,R2all))
attr(res,"log2TotalSD") = rowSds(Dall[,IT])
```

The R/Bioconductor package limma shrinks the standard deviation of each gene towards a global estimate of standard deviation along all genes. In the case of large outliers, this can result in spurious hits. To avoid these, genes with standard deviation larger than the absolute value of the log-fold change are not regarded as significant.

```
K = which(res$significant != "" & (fita$sigma > abs(fita$coefficients[,1])))

write.table(res[K,], sep="\t", file = file.path(d, "filteredGenes.txt"), quote=FALSE)

res$padj[K] = NA
res$significant[K] = ""
```

```
col=rep("gray", length(padja))
col[which(res$significant == "*" & lfca > 0)] = brewer.pal(4,"RdBu")[2]
col[which(res$significant == "*" & lfca < 0)] = brewer.pal(4,"RdBu")[3]
col[which(res$significant == "**" & lfca > 0)] = brewer.pal(4,"RdBu")[1]
col[which(res$significant == "**" & lfca < 0)] = brewer.pal(4,"RdBu")[4]
plot(rowMeans(D[,SL$Type == "RNAseq"]),lfca,
     pch=20,cex=0.7,col=col,
     xlab="log2 read counts (total RNA)", ylab="translation efficiency [log2]")
abline(h=0)

plot(R1,R2,
     pch=20,cex=0.7,col=col,
     xlab="translation efficiency [log2] (rep. 1)",
     ylab="translation efficiency [log2] (rep. 2)",
     xlim=c(-4,4),ylim=c(-4,4))
```

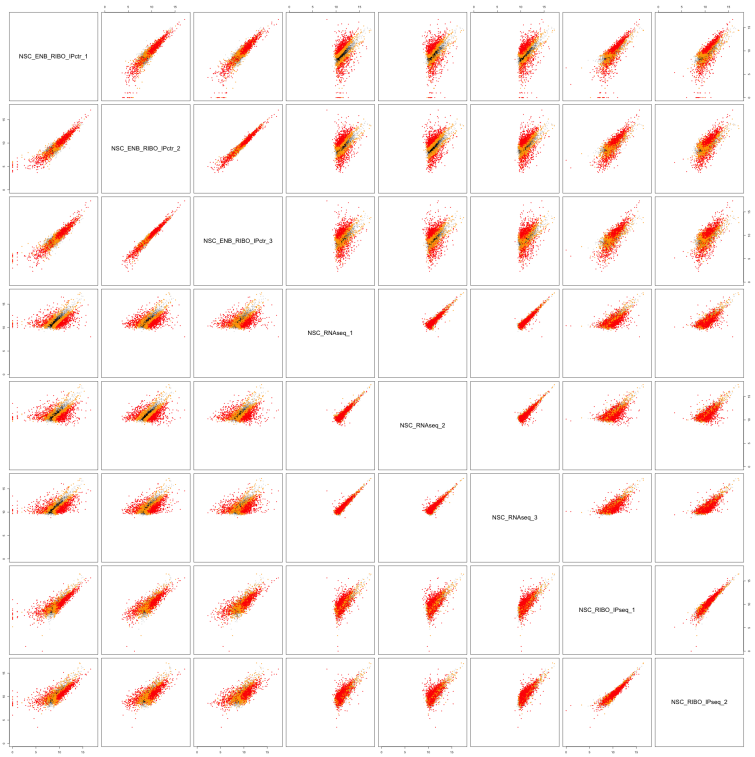
```
abline(h=0,v=0)
text(-3.8,3.8,paste0("R = ",format(cor(R1,R2,use="pairwise.complete"),digits=2)),adj = c(0,1))
```



The data is saved.

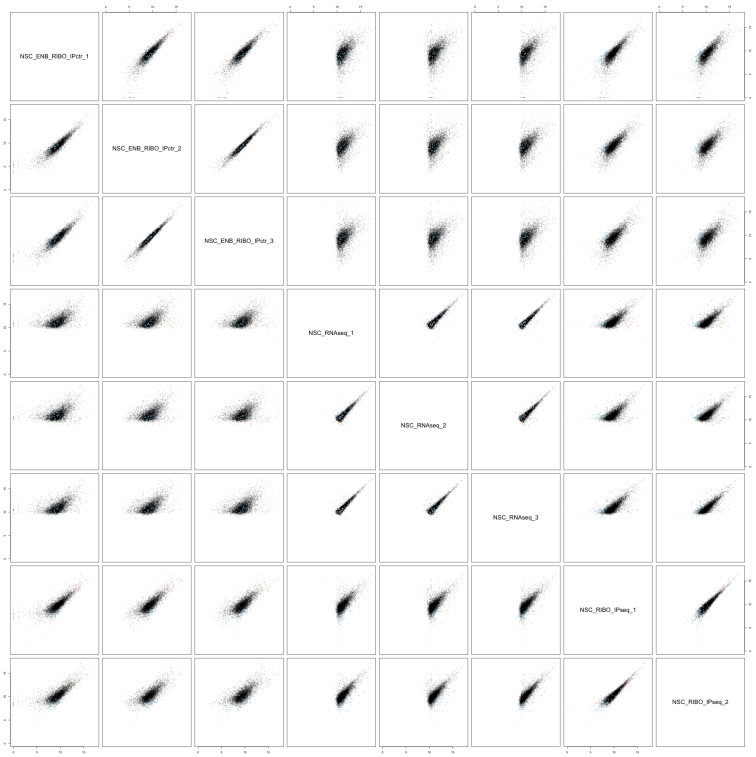
```
save(EC, file=file.path(d, "EC.rda"))
save(res, file=file.path(d, "res.rda"))
save(fitc, file=file.path(d, "fitc.rda"))
save(ExpVar, file=file.path(d, "ExpVar.rda"))
```

```
col = rep("#00000022", dim(C)[1])
col[which(fitc$padj <= 0.1)] = "orange"
col[which(fitc$padj <= 0.01)] = "red"
I = order(factor(col, levels=c("#00000022","orange","red")))
lim = range(as.vector(C))
pairs(C[I,],pch=20,cex=0.5,xlim=lim, ylim=lim,col=col[I])
```



```
col=rep("#00000022", nrow(res))
col[which(res$significant == "*" & res$log2FoldChange > 0)] = brewer.pal(4,"RdBu")[2]
col[which(res$significant == "*" & res$log2FoldChange < 0)] = brewer.pal(4,"RdBu")[3]
col[which(res$significant == "**" & res$log2FoldChange > 0)] = brewer.pal(4,"RdBu")[1]
```

```
col[which(res$significant == "**" & res$log2FoldChange < 0)] = brewer.pal(4,"RdBu")[4]

I = order(factor(col, levels=c("#00000022",brewer.pal(4,"RdBu")[c(2,3,1,4)])))
lim = range(as.vector(C))
pairs(C[I,],pch=20,cex=0.5,xlim=lim, ylim=lim,col=col[I])
```




```
fgup = row.names(res)[which(res$significant != "" & res$log2FoldChange > 0)]
fgdown = row.names(res)[which(res$significant != "" & res$log2FoldChange < 0)]
bg = row.names(res)[which(!is.na(res$padj))]
GSEAup = doGSEA(fgup,bg,ENSG2category, k=3)
GSEAdown = doGSEA(fgdown,bg,ENSG2category, k=3)
```

```
res2 = res[order(factor(res$significant, levels=c("**","*","")),
                 sign(-res$log2FoldChange),res$symbol),]
res2$UnspecBinders = ""
res2$UnspecBinders[row.names(res2) %in% names(UnspecBinders)] = "UnspecBinder"
write.table(res2, file=file.path(d, "resulttable.txt"), sep="\t", quote=FALSE)
page = openPage(file.path(d,"resulttable.html"), link.css="hwriter.css")
hwriteSidebar(page=page, link=mainmenu)
hwrite(sprintf("Translation efficiency (%s); result table",n),
       heading=1, page=page)
hwrite("** = 1% FDR; * = 10% FDR",page=page,br=TRUE)
col = rep("",nrow(res))
col[which(res2$significant %in% c("*","**") & res2$log2FoldChange > 0)] =
  brewer.pal(3,"Pastel1")[1]
col[which(res2$significant %in% c("*","**") & res2$log2FoldChange < 0)] =
  brewer.pal(3,"Pastel1")[2]
hwrite(as.data.frame(res2), row.bgcolor=col, page=page)
closePage(page, splash=FALSE)
```

3.2 Filter unspecific binders in ENB

Create an output directory.

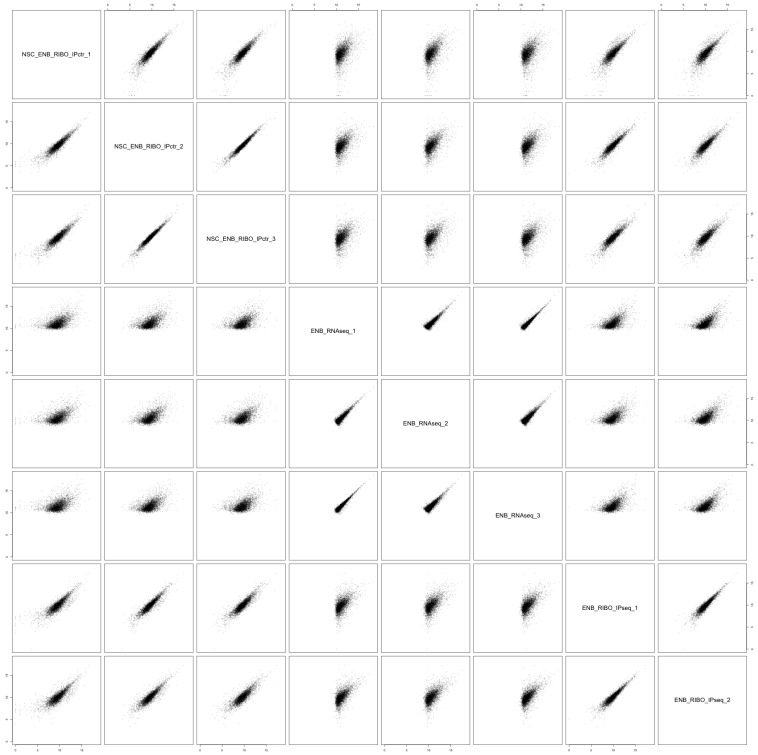
```
n = "ENB"
d = file.path("result","TranslationEfficiency", n)
dir.create(d, recursive = TRUE, showWarnings = FALSE)
mainmenu = file.path("..","..","..","mainmenu.html")
source(file.path("R","doGSEA.R"))
```

A DESeq dataset is created that contains the respective samples.

```
Design = DesignTranslationEfficiency
EC_all <- expectedCounts[,SamplesTranslationEfficiency[[n]]]
SL <- SampleList[SamplesTranslationEfficiency[[n]],]
# dds_all <- DESeqDataSetFromMatrix(
#     countData = expectedCounts[,SamplesTranslationEfficiency[[n]]],
#     colData = SampleList[SamplesTranslationEfficiency[[n]],],
#     design = ~ Type)
# mcols(dds_all) = Anno
# colData(dds_all)
```

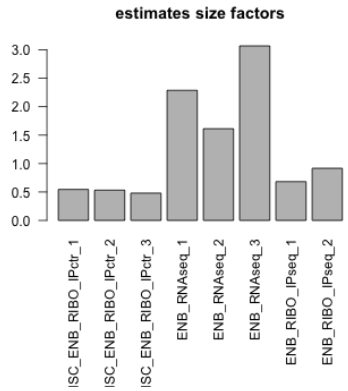
Log counts are computed and plotted as a pairs plot.

```
I = which(rowMeans(log2(EC_all[,
            SL$Type == "RNAseq"]+1)) >= 10 &
            Anno$transript_type == "protein_coding")
EC = EC_all[I,]
# I = which(rowMeans(log2(counts(dds_all)[,
#             colData(dds_all)£Type == "RNAseq"]+1)) >= 10 &
#             mcols(dds_all)£transript_type == "protein_coding")
# dds = dds_all[I,]

C = log2(EC+1)
Call = log2(EC_all+1)
# C = log2(counts(dds)+1)
# Call = log2(counts(dds_all)+1)
lim = range(as.vector(C))
pairs(C,pch=20,cex=0.5,col="#00000022",xlim=lim, ylim=lim)
```



The genes in the cluster with lower read counts in the control sample are used to estimate the size factors.

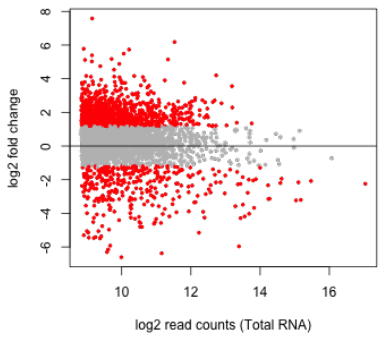
```
s = sizeFactors(estimateSizeFactors(DESeqDataSetFromMatrix(
    countData = round(EC), colData = SL, design = ~ Type),
    controlGenes=which(rowMeans(log2(EC[,SL$Type == "RNAseq"]+1)) >= 13)))

## converting counts to integer mode

#sizeFactors(dds) = s
p = par(mar=c(10,4,4,2)+0.1)
barplot(s, main="estimates size factors",las=2)
par(p)
```



The count data is normalized and unspecific binders are identified. Therefore, RiboControl samples are compared to total RNA samples. The genes that are significantly higher in RiboControl samples are potential unspecific binders of the tag. These genes are filtered as well.

```
D = C
m = log2(s)
for (i in 1:ncol(EC)) {
  D[,i] = D[,i] - m[i]
}
```

```
Dall = Call
for (i in 1:ncol(EC)) {
  Dall[,i] = Dall[,i] - m[i]
}

IR = which(SL$Type == "RIBO_IPseq")
IT = which(SL$Type == "RNAseq")
IC = which(SL$Type == "RIBO_IPctr")

fitc = eBayes(lmFit(cbind(D[,IC[1]]-D[,IT[1]],D[,IC[2]]-D[,IT[2]])))
padjc = p.adjust(fitc$p.value[,1],method="BH")
lfcc = fitc$coefficients[,1]

plot(rowMeans(D[,SL$Type == "RNAseq"]),lfcc,
     pch=20,cex=0.7,col=ifelse(padjc < 0.01, "red","gray"),
     xlab="log2 read counts (Total RNA)", ylab="log2 fold change")
abline(h=0)

UnspecBinders = which(padjc <= 0.01 & lfcc > 0)

fitc$padj = padjc
```



To estimate the translation efficiency, the RiboTag data is modeled as a function of RiboControl and total RNA.

```
fit1 = list()
for (i in 1:2) {
  D2 = data.frame("Ribo" = D[,IR[i]], "Total" = D[,IT[i]], "ctrl" = D[,IC[i]])
  fit1[[i]] = lm(Ribo ~ Total+ctrl, data=D2)
  fit1[[i]]$anova = anova(fit1[[i]])
  fit1[[i]]$expVar = fit1[[i]]$anova["Sum Sq",drop=FALSE]
  fit1[[i]]$expVar = fit1[[i]]$expVar/sum(fit1[[i]]$expVar)
  D2all = data.frame("Ribo" = Dall[,IR[i]], "Total" = Dall[,IT[i]], "ctrl" = Dall[,IC[i]], row.names =
  p = predict(fit1[[i]],newdata = D2all)
  fit1[[i]]$residualsAll = D2all$Ribo - p
}

R1 = fit1[[1]]$residuals
R2 = fit1[[2]]$residuals
R1all = fit1[[1]]$residualsAll
R2all = fit1[[2]]$residualsAll
R1all = R1all - shorth(R1,na.rm=TRUE)
R2all = R2all - shorth(R2,na.rm=TRUE)
R1 = R1 - shorth(R1,na.rm=TRUE)
R2 = R2 - shorth(R2,na.rm=TRUE)
```

```
D2 = data.frame("R1" = R1, "R2" = R2)
fit2 = lm(data=D2)
fit2$anova = anova(fit2)
fit2$expVar = fit2$anova[,"Sum Sq",drop=FALSE]
fit2$expVar = fit2$expVar/sum(fit2$expVar)

fita = eBayes(lmFit(cbind(R1,R2)))
padja = p.adjust(fita$p.value[,1],method="BH")
lfca = fita$coefficients[,1]
```

```
ExpVar = t(as.matrix(cbind(rep1=fit1[[1]]$expVar,rep2=fit1[[2]]$expVar)))
ExpVar = cbind(ExpVar[,1:2],ExpVar[,3] * fit2$expVar[1,1], ExpVar[,3] * fit2$expVar[2,1])
colnames(ExpVar) = c("total RNA", "background", "transl. eff.", "residuals")
row.names(ExpVar) = paste0("rep ",1:2)
# barplot(ExpVar,beside=TRUE,
#         legend=TRUE,ylab="explained variance")
# barplot(t(ExpVar),beside=FALSE,
#         legend=TRUE,ylab="explained variance",xlim=c(0,4))
```

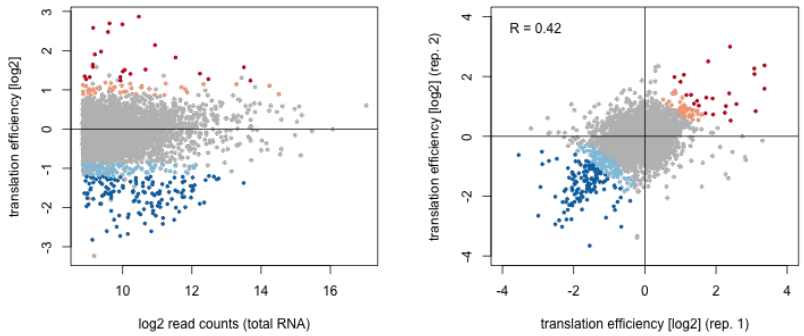
```
res = as.data.frame(Anno[row.names(EC),])
#res$symbol[is.na(res$symbol)] = row.names(res)[is.na(res$symbol)]

res$pval = fita$p.value[,1]
res$padj = padja
res$log2FoldChange = lfca
res$significant = ""
res$significant[which(padja <= 0.1)] = "*"
res$significant[which(padja <= 0.01)] = "**"
attr(res,"log2FC") = rowMeans(cbind(R1all,R2all))
attr(res,"log2total") = rowMeans(Dall[,IT])
attr(res,"log2FCsd") = rowSds(cbind(R1all,R2all))
attr(res,"log2TotalSD") = rowSds(Dall[,IT])
```

The R/Bioconductor package limma shrinks the standard deviation of each gene towards a global estimate of standard deviation along all genes. In the case of large outliers, this can result in spurious hits. To avoid these, genes with standard deviation larger than the absolute value of the log-fold change are not regarded as significant.

```
K = which(res$significant != "" & (fita$sigma > abs(fita$coefficients[,1])))

write.table(res[K,], sep="\t", file = file.path(d, "filteredGenes.txt"), quote=FALSE)

res$padj[K] = NA
res$significant[K] = ""
```

```
col=rep("gray", length(padja))
col[which(res$significant == "*" & lfca > 0)] = brewer.pal(4,"RdBu")[2]
col[which(res$significant == "*" & lfca < 0)] = brewer.pal(4,"RdBu")[3]
col[which(res$significant == "**" & lfca > 0)] = brewer.pal(4,"RdBu")[1]
col[which(res$significant == "**" & lfca < 0)] = brewer.pal(4,"RdBu")[4]
plot(rowMeans(D[,SL$Type == "RNAseq"]),lfca,
     pch=20,cex=0.7,col=col,
     xlab="log2 read counts (total RNA)", ylab="translation efficiency [log2]")
abline(h=0)

plot(R1,R2,
     pch=20,cex=0.7,col=col,
     xlab="translation efficiency [log2] (rep. 1)",
     ylab="translation efficiency [log2] (rep. 2)",
     xlim=c(-4,4),ylim=c(-4,4))
```

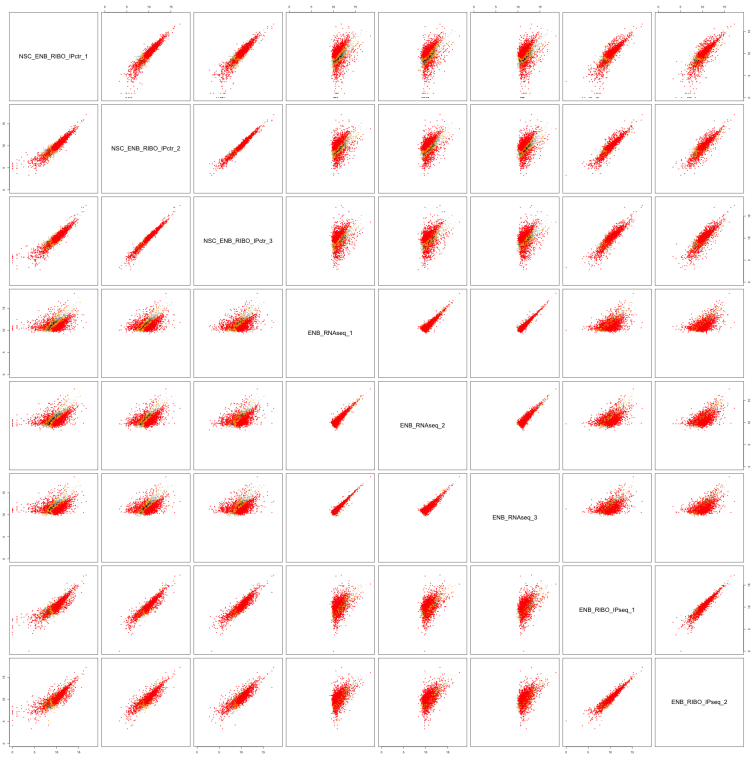
```
abline(h=0,v=0)
text(-3.8,3.8,paste0("R = ",format(cor(R1,R2,use="pairwise.complete"),digits=2)),adj = c(0,1))
```



The data is saved.

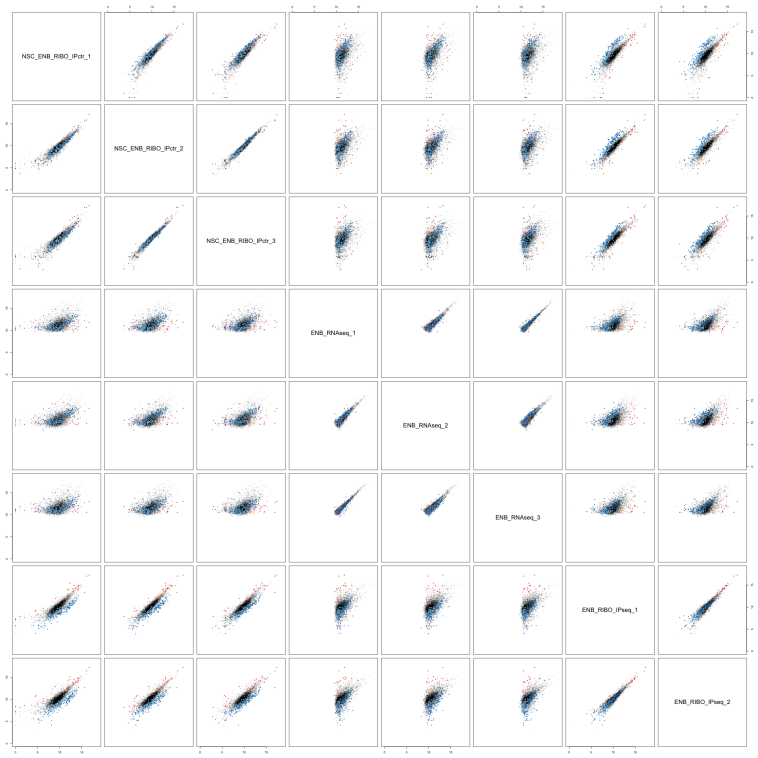
```
save(EC, file=file.path(d, "EC.rda"))
save(res, file=file.path(d, "res.rda"))
save(fitc, file=file.path(d, "fitc.rda"))
save(ExpVar, file=file.path(d, "ExpVar.rda"))
```

```
col = rep("#00000022", dim(C)[1])
col[which(fitc$padj <= 0.1)] = "orange"
col[which(fitc$padj <= 0.01)] = "red"
I = order(factor(col, levels=c("#00000022","orange","red")))
lim = range(as.vector(C))
pairs(C[I,],pch=20,cex=0.5,xlim=lim, ylim=lim,col=col[I])
```




```
col=rep("#00000022", nrow(res))
col[which(res$significant == "*" & res$log2FoldChange > 0)] = brewer.pal(4,"RdBu")[2]
col[which(res$significant == "*" & res$log2FoldChange < 0)] = brewer.pal(4,"RdBu")[3]
col[which(res$significant == "**" & res$log2FoldChange > 0)] = brewer.pal(4,"RdBu")[1]
```

```
col[which(res$significant == "**" & res$log2FoldChange < 0)] = brewer.pal(4,"RdBu")[4]

I = order(factor(col, levels=c("#00000022",brewer.pal(4,"RdBu")[c(2,3,1,4)])))
lim = range(as.vector(C))
pairs(C[I,],pch=20,cex=0.5,xlim=lim, ylim=lim,col=col[I])
```




```
fgup = row.names(res)[which(res$significant != "" & res$log2FoldChange > 0)]
fgdown = row.names(res)[which(res$significant != "" & res$log2FoldChange < 0)]
bg = row.names(res)[which(!is.na(res$padj))]
GSEAup = doGSEA(fgup,bg,ENSG2category, k=3)
GSEAdown = doGSEA(fgdown,bg,ENSG2category, k=3)
```

```
res2 = res[order(factor(res$significant, levels=c("**","*","")),
                 sign(-res$log2FoldChange),res$symbol),]
res2$UnspecBinders = ""
res2$UnspecBinders[row.names(res2) %in% names(UnspecBinders)] = "UnspecBinder"
write.table(res2, file=file.path(d, "resulttable.txt"), sep="\t", quote=FALSE)
page = openPage(file.path(d,"resulttable.html"), link.css="hwriter.css")
hwriteSidebar(page=page, link=mainmenu)
hwrite(sprintf("Translation efficiency (%s); result table",n),
       heading=1, page=page)
hwrite("** = 1% FDR; * = 10% FDR",page=page,br=TRUE)
col = rep("",nrow(res))
col[which(res2$significant %in% c("*","**") & res2$log2FoldChange > 0)] =
  brewer.pal(3,"Pastel1")[1]
col[which(res2$significant %in% c("*","**") & res2$log2FoldChange < 0)] =
  brewer.pal(3,"Pastel1")[2]
hwrite(as.data.frame(res2), row.bgcolor=col, page=page)
closePage(page, splash=FALSE)
```

3.3 Filter unspecific binders in LNB

Create an output directory.

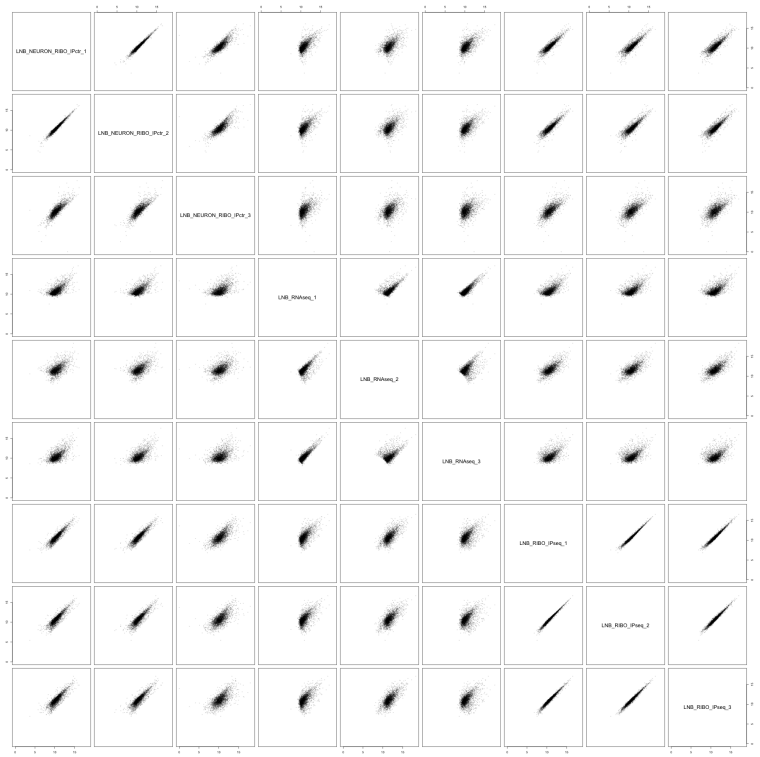
```
n = "LNB"
d = file.path("result","TranslationEfficiency", n)
dir.create(d, recursive = TRUE, showWarnings = FALSE)
mainmenu = file.path("..","..","..","mainmenu.html")
source(file.path("R","doGSEA.R"))
```

A DESeq dataset is created that contains the respective samples.

```
Design = DesignTranslationEfficiency
EC_all <- expectedCounts[,SamplesTranslationEfficiency[[n]]]
SL <- SampleList[SamplesTranslationEfficiency[[n]],]
# dds_all <- DESeqDataSetFromMatrix(
#     countData = expectedCounts[,SamplesTranslationEfficiency[[n]]],
#     colData = SampleList[SamplesTranslationEfficiency[[n]],],
#     design = ~ Type)
# mcols(dds_all) = Anno
# colData(dds_all)
```

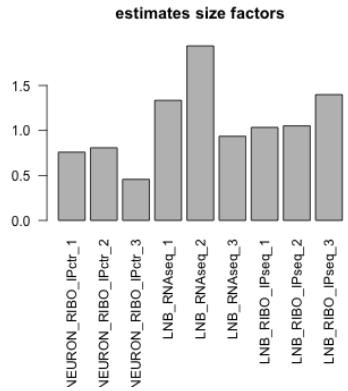
Log counts are computed and plotted as a pairs plot.

```
I = which(rowMeans(log2(EC_all[,
            SL$Type == "RNAseq"]+1)) >= 10 &
            Anno$transript_type == "protein_coding")
EC = EC_all[I,]
# I = which(rowMeans(log2(counts(dds_all)[,
#             colData(dds_all)£Type == "RNAseq"]+1)) >= 10 &
#             mcols(dds_all)£transript_type == "protein_coding")
# dds = dds_all[I,]

C = log2(EC+1)
Call = log2(EC_all+1)
# C = log2(counts(dds)+1)
# Call = log2(counts(dds_all)+1)
lim = range(as.vector(C))
pairs(C,pch=20,cex=0.5,col="#00000022",xlim=lim, ylim=lim)
```



The genes in the cluster with lower read counts in the control sample are used to estimate the size factors.

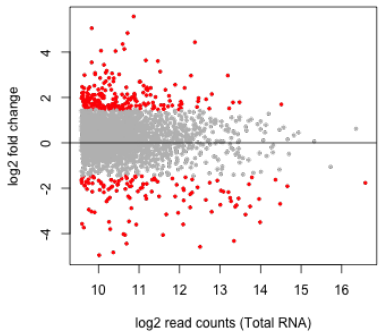
```
s = sizeFactors(estimateSizeFactors(DESeqDataSetFromMatrix(
    countData = round(EC), colData = SL, design = ~ Type),
    controlGenes=which(rowMeans(log2(EC[,SL$Type == "RNAseq"]+1)) >= 13)))

## converting counts to integer mode

#sizeFactors(dds) = s
p = par(mar=c(10,4,4,2)+0.1)
barplot(s, main="estimates size factors",las=2)
par(p)
```



The count data is normalized and unspecific binders are identified. Therefore, RiboControl samples are compared to total RNA samples. The genes that are significantly higher in RiboControl samples are potential unspecific binders of the tag. These genes are filtered as well.

```
D = C
m = log2(s)
for (i in 1:ncol(EC)) {
  D[,i] = D[,i] - m[i]
}
```

```
Dall = Call
for (i in 1:ncol(EC)) {
  Dall[,i] = Dall[,i] - m[i]
}

IR = which(SL$Type == "RIBO_IPseq")
IT = which(SL$Type == "RNAseq")
IC = which(SL$Type == "RIBO_IPctr")

fitc = eBayes(lmFit(cbind(D[,IC[1]]-D[,IT[1]],D[,IC[2]]-D[,IT[2]])))
padjc = p.adjust(fitc$p.value[,1],method="BH")
lfcc = fitc$coefficients[,1]

plot(rowMeans(D[,SL$Type == "RNAseq"]),lfcc,
     pch=20,cex=0.7,col=ifelse(padjc < 0.01, "red","gray"),
     xlab="log2 read counts (Total RNA)", ylab="log2 fold change")
abline(h=0)

UnspecBinders = which(padjc <= 0.01 & lfcc > 0)

fitc$padj = padjc
```



To estimate the translation efficiency, the RiboTag data is modeled as a function of RiboControl and total RNA.

```
fit1 = list()
for (i in 1:2) {
  D2 = data.frame("Ribo" = D[,IR[i]], "Total" = D[,IT[i]], "ctrl" = D[,IC[i]])
  fit1[[i]] = lm(Ribo ~ Total+ctrl, data=D2)
  fit1[[i]]$anova = anova(fit1[[i]])
  fit1[[i]]$expVar = fit1[[i]]$anova["Sum Sq",drop=FALSE]
  fit1[[i]]$expVar = fit1[[i]]$expVar/sum(fit1[[i]]$expVar)
  D2all = data.frame("Ribo" = Dall[,IR[i]], "Total" = Dall[,IT[i]], "ctrl" = Dall[,IC[i]], row.names =
  p = predict(fit1[[i]],newdata = D2all)
  fit1[[i]]$residualsAll = D2all$Ribo - p
}

R1 = fit1[[1]]$residuals
R2 = fit1[[2]]$residuals
R1all = fit1[[1]]$residualsAll
R2all = fit1[[2]]$residualsAll
R1all = R1all - shorth(R1,na.rm=TRUE)
R2all = R2all - shorth(R2,na.rm=TRUE)
R1 = R1 - shorth(R1,na.rm=TRUE)
R2 = R2 - shorth(R2,na.rm=TRUE)
```

```
D2 = data.frame("R1" = R1, "R2" = R2)
fit2 = lm(data=D2)
fit2$anova = anova(fit2)
fit2$expVar = fit2$anova[,"Sum Sq",drop=FALSE]
fit2$expVar = fit2$expVar/sum(fit2$expVar)

fita = eBayes(lmFit(cbind(R1,R2)))
padja = p.adjust(fita$p.value[,1],method="BH")
lfca = fita$coefficients[,1]
```

```
ExpVar = t(as.matrix(cbind(rep1=fit1[[1]]$expVar,rep2=fit1[[2]]$expVar)))
ExpVar = cbind(ExpVar[,1:2],ExpVar[,3] * fit2$expVar[1,1], ExpVar[,3] * fit2$expVar[2,1])
colnames(ExpVar) = c("total RNA", "background", "transl. eff.", "residuals")
row.names(ExpVar) = paste0("rep ",1:2)
# barplot(ExpVar,beside=TRUE,
#         legend=TRUE,ylab="explained variance")
# barplot(t(ExpVar),beside=FALSE,
#         legend=TRUE,ylab="explained variance",xlim=c(0,4))
```

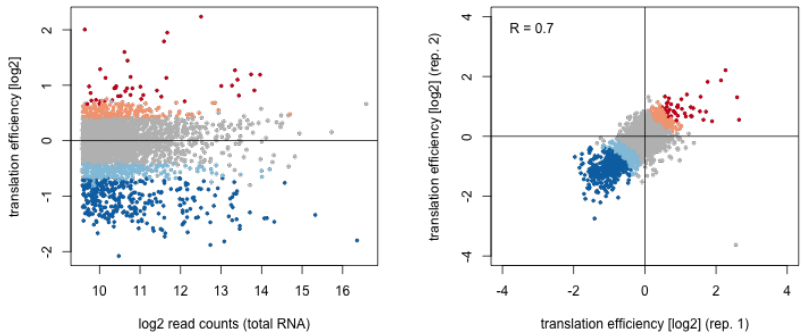
```
res = as.data.frame(Anno[row.names(EC),])
#res$symbol[is.na(res$symbol)] = row.names(res)[is.na(res$symbol)]

res$pval = fita$p.value[,1]
res$padj = padja
res$log2FoldChange = lfca
res$significant = ""
res$significant[which(padja <= 0.1)] = "*"
res$significant[which(padja <= 0.01)] = "**"
attr(res,"log2FC") = rowMeans(cbind(R1all,R2all))
attr(res,"log2total") = rowMeans(Dall[,IT])
attr(res,"log2FCsd") = rowSds(cbind(R1all,R2all))
attr(res,"log2TotalSD") = rowSds(Dall[,IT])
```

The R/Bioconductor package limma shrinks the standard deviation of each gene towards a global estimate of standard deviation along all genes. In the case of large outliers, this can result in spurious hits. To avoid these, genes with standard deviation larger than the absolute value of the log-fold change are not regarded as significant.

```
K = which(res$significant != "" & (fita$sigma > abs(fita$coefficients[,1])))

write.table(res[K,], sep="\t", file = file.path(d, "filteredGenes.txt"), quote=FALSE)

res$padj[K] = NA
res$significant[K] = ""
```

```
col=rep("gray", length(padja))
col[which(res$significant == "*" & lfca > 0)] = brewer.pal(4,"RdBu")[2]
col[which(res$significant == "*" & lfca < 0)] = brewer.pal(4,"RdBu")[3]
col[which(res$significant == "**" & lfca > 0)] = brewer.pal(4,"RdBu")[1]
col[which(res$significant == "**" & lfca < 0)] = brewer.pal(4,"RdBu")[4]
plot(rowMeans(D[,SL$Type == "RNAseq"]),lfca,
     pch=20,cex=0.7,col=col,
     xlab="log2 read counts (total RNA)", ylab="translation efficiency [log2]")
abline(h=0)

plot(R1,R2,
     pch=20,cex=0.7,col=col,
     xlab="translation efficiency [log2] (rep. 1)",
     ylab="translation efficiency [log2] (rep. 2)",
     xlim=c(-4,4),ylim=c(-4,4))
```

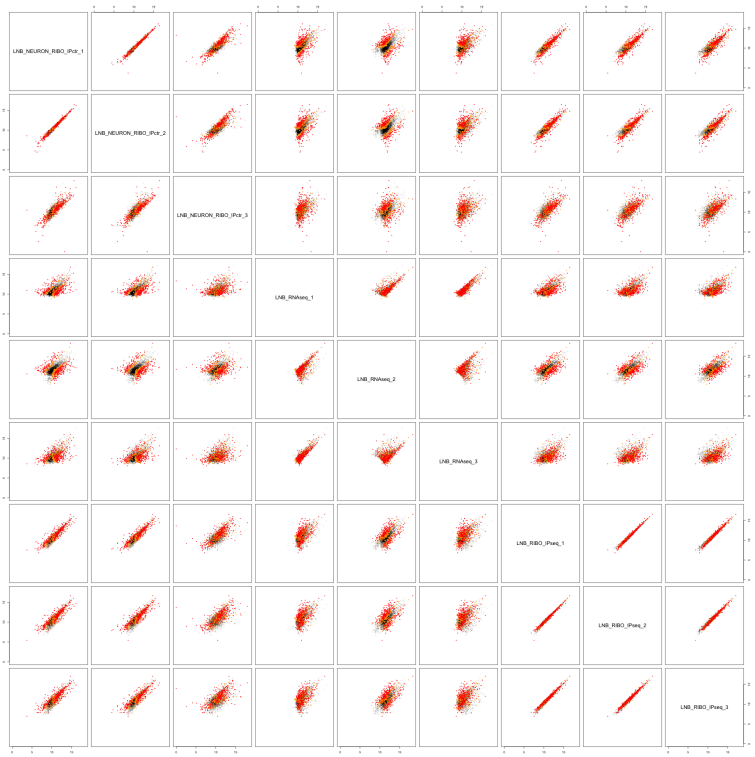
```
abline(h=0,v=0)
text(-3.8,3.8,paste0("R = ",format(cor(R1,R2,use="pairwise.complete"),digits=2)),adj = c(0,1))
```



The data is saved.

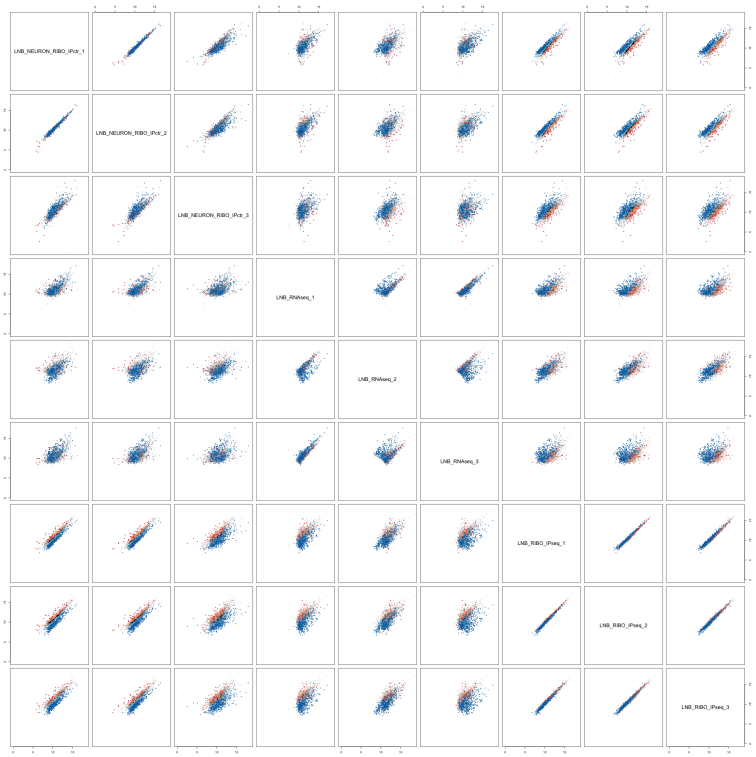
```
save(EC, file=file.path(d, "EC.rda"))
save(res, file=file.path(d, "res.rda"))
save(fitc, file=file.path(d, "fitc.rda"))
save(ExpVar, file=file.path(d, "ExpVar.rda"))
```

```
col = rep("#00000022", dim(C)[1])
col[which(fitc$padj <= 0.1)] = "orange"
col[which(fitc$padj <= 0.01)] = "red"
I = order(factor(col, levels=c("#00000022","orange","red")))
lim = range(as.vector(C))
pairs(C[I,],pch=20,cex=0.5,xlim=lim, ylim=lim,col=col[I])
```



```
col=rep("#00000022", nrow(res))
col[which(res$significant == "*" & res$log2FoldChange > 0)] = brewer.pal(4,"RdBu")[2]
col[which(res$significant == "*" & res$log2FoldChange < 0)] = brewer.pal(4,"RdBu")[3]
col[which(res$significant == "**" & res$log2FoldChange > 0)] = brewer.pal(4,"RdBu")[1]
```

```
col[which(res$significant == "**" & res$log2FoldChange < 0)] = brewer.pal(4,"RdBu")[4]

I = order(factor(col, levels=c("#00000022",brewer.pal(4,"RdBu")[c(2,3,1,4)])))
lim = range(as.vector(C))
pairs(C[I,],pch=20,cex=0.5,xlim=lim, ylim=lim,col=col[I])
```



```
fgup = row.names(res)[which(res$significant != "" & res$log2FoldChange > 0)]
fgdown = row.names(res)[which(res$significant != "" & res$log2FoldChange < 0)]
bg = row.names(res)[which(!is.na(res$padj))]
GSEAup = doGSEA(fgup,bg,ENSG2category, k=3)
GSEAdown = doGSEA(fgdown,bg,ENSG2category, k=3)
```

```
res2 = res[order(factor(res$significant, levels=c("**","*","")),
                 sign(-res$log2FoldChange),res$symbol),]
res2$UnspecBinders = ""
res2$UnspecBinders[row.names(res2) %in% names(UnspecBinders)] = "UnspecBinder"
write.table(res2, file=file.path(d, "resulttable.txt"), sep="\t", quote=FALSE)
page = openPage(file.path(d,"resulttable.html"), link.css="hwriter.css")
hwriteSidebar(page=page, link=mainmenu)
hwrite(sprintf("Translation efficiency (%s); result table",n),
       heading=1, page=page)
hwrite("** = 1% FDR; * = 10% FDR",page=page,br=TRUE)
col = rep("",nrow(res))
col[which(res2$significant %in% c("*","**") & res2$log2FoldChange > 0)] =
  brewer.pal(3,"Pastel1")[1]
col[which(res2$significant %in% c("*","**") & res2$log2FoldChange < 0)] =
  brewer.pal(3,"Pastel1")[2]
hwrite(as.data.frame(res2), row.bgcolor=col, page=page)
closePage(page, splash=FALSE)
```

3.4 Filter unspecific binders in NEURON

Create an output directory.

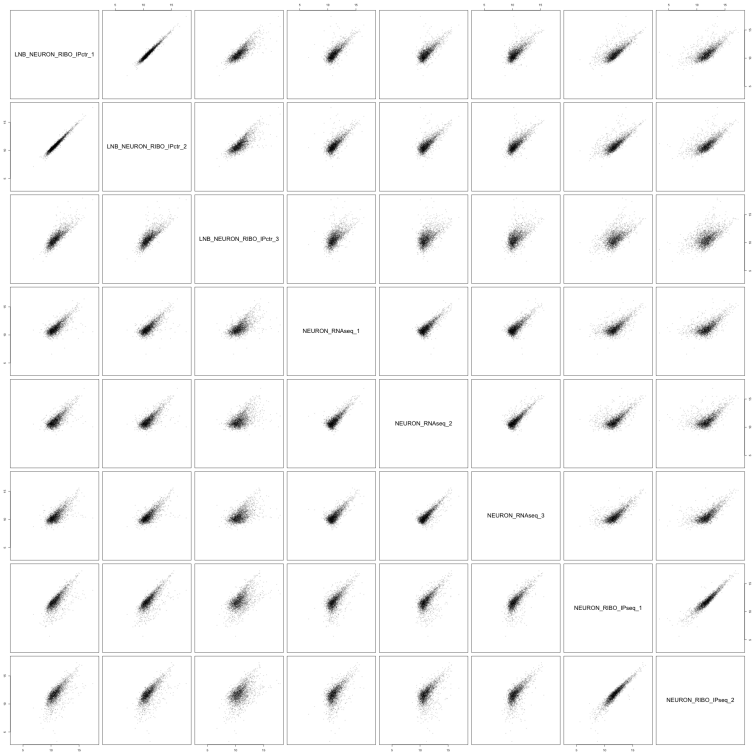
```
n = "NEURON"
d = file.path("result","TranslationEfficiency", n)
dir.create(d, recursive = TRUE, showWarnings = FALSE)
mainmenu = file.path("..","..","..","mainmenu.html")
source(file.path("R","doGSEA.R"))
```

A DESeq dataset is created that contains the respective samples.

```
Design = DesignTranslationEfficiency
EC_all <- expectedCounts[,SamplesTranslationEfficiency[[n]]]
SL <- SampleList[SamplesTranslationEfficiency[[n]],]
# dds_all <- DESeqDataSetFromMatrix(
#     countData = expectedCounts[,SamplesTranslationEfficiency[[n]]],
#     colData = SampleList[SamplesTranslationEfficiency[[n]],],
#     design = ~ Type)
# mcols(dds_all) = Anno
# colData(dds_all)
```

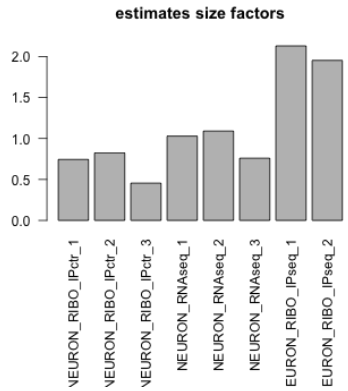
Log counts are computed and plotted as a pairs plot.

```
I = which(rowMeans(log2(EC_all[,
            SL$Type == "RNAseq"]+1)) >= 10 &
            Anno$transript_type == "protein_coding")
EC = EC_all[I,]
# I = which(rowMeans(log2(counts(dds_all)[,
#             colData(dds_all)£Type == "RNAseq"]+1)) >= 10 &
#             mcols(dds_all)£transript_type == "protein_coding")
# dds = dds_all[I,]

C = log2(EC+1)
Call = log2(EC_all+1)
# C = log2(counts(dds)+1)
# Call = log2(counts(dds_all)+1)
lim = range(as.vector(C))
pairs(C,pch=20,cex=0.5,col="#00000022",xlim=lim, ylim=lim)
```



The genes in the cluster with lower read counts in the control sample are used to estimate the size factors.

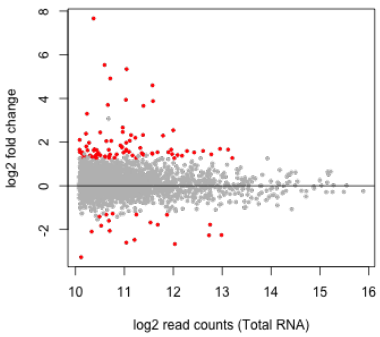
```
s = sizeFactors(estimateSizeFactors(DESeqDataSetFromMatrix(
    countData = round(EC), colData = SL, design = ~ Type),
    controlGenes=which(rowMeans(log2(EC[,SL$Type == "RNAseq"]+1)) >= 13)))

## converting counts to integer mode

#sizeFactors(dds) = s
p = par(mar=c(10,4,4,2)+0.1)
barplot(s, main="estimates size factors",las=2)
par(p)
```



The count data is normalized and unspecific binders are identified. Therefore, RiboControl samples are compared to total RNA samples. The genes that are significantly higher in RiboControl samples are potential unspecific binders of the tag. These genes are filtered as well.

```
D = C
m = log2(s)
for (i in 1:ncol(EC)) {
  D[,i] = D[,i] - m[i]
}
```

```
Dall = Call
for (i in 1:ncol(EC)) {
  Dall[,i] = Dall[,i] - m[i]
}

IR = which(SL$Type == "RIBO_IPseq")
IT = which(SL$Type == "RNAseq")
IC = which(SL$Type == "RIBO_IPctr")

fitc = eBayes(lmFit(cbind(D[,IC[1]]-D[,IT[1]],D[,IC[2]]-D[,IT[2]])))
padjc = p.adjust(fitc$p.value[,1],method="BH")
lfcc = fitc$coefficients[,1]

plot(rowMeans(D[,SL$Type == "RNAseq"]),lfcc,
     pch=20,cex=0.7,col=ifelse(padjc < 0.01, "red","gray"),
     xlab="log2 read counts (Total RNA)", ylab="log2 fold change")
abline(h=0)

UnspecBinders = which(padjc <= 0.01 & lfcc > 0)

fitc$padj = padjc
```



To estimate the translation efficiency, the RiboTag data is modeled as a function of RiboControl and total RNA.

```
fit1 = list()
for (i in 1:2) {
  D2 = data.frame("Ribo" = D[,IR[i]], "Total" = D[,IT[i]], "ctrl" = D[,IC[i]])
  fit1[[i]] = lm(Ribo ~ Total+ctrl, data=D2)
  fit1[[i]]$anova = anova(fit1[[i]])
  fit1[[i]]$expVar = fit1[[i]]$anova["Sum Sq",drop=FALSE]
  fit1[[i]]$expVar = fit1[[i]]$expVar/sum(fit1[[i]]$expVar)
  D2all = data.frame("Ribo" = Dall[,IR[i]], "Total" = Dall[,IT[i]], "ctrl" = Dall[,IC[i]], row.names =
  p = predict(fit1[[i]],newdata = D2all)
  fit1[[i]]$residualsAll = D2all$Ribo - p
}

R1 = fit1[[1]]$residuals
R2 = fit1[[2]]$residuals
R1all = fit1[[1]]$residualsAll
R2all = fit1[[2]]$residualsAll
R1all = R1all - shorth(R1,na.rm=TRUE)
R2all = R2all - shorth(R2,na.rm=TRUE)
R1 = R1 - shorth(R1,na.rm=TRUE)
R2 = R2 - shorth(R2,na.rm=TRUE)
```

```
D2 = data.frame("R1" = R1, "R2" = R2)
fit2 = lm(data=D2)
fit2$anova = anova(fit2)
fit2$expVar = fit2$anova[,"Sum Sq",drop=FALSE]
fit2$expVar = fit2$expVar/sum(fit2$expVar)

fita = eBayes(lmFit(cbind(R1,R2)))
padja = p.adjust(fita$p.value[,1],method="BH")
lfca = fita$coefficients[,1]
```

```
ExpVar = t(as.matrix(cbind(rep1=fit1[[1]]$expVar,rep2=fit1[[2]]$expVar)))
ExpVar = cbind(ExpVar[,1:2],ExpVar[,3] * fit2$expVar[1,1], ExpVar[,3] * fit2$expVar[2,1])
colnames(ExpVar) = c("total RNA", "background", "transl. eff.", "residuals")
row.names(ExpVar) = paste0("rep ",1:2)
# barplot(ExpVar,beside=TRUE,
#         legend=TRUE,ylab="explained variance")
# barplot(t(ExpVar),beside=FALSE,
#         legend=TRUE,ylab="explained variance",xlim=c(0,4))
```

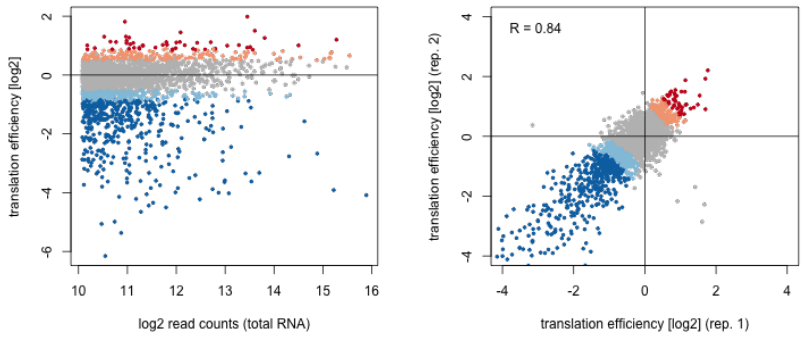
```
res = as.data.frame(Anno[row.names(EC),])
#res$symbol[is.na(res$symbol)] = row.names(res)[is.na(res$symbol)]

res$pval = fita$p.value[,1]
res$padj = padja
res$log2FoldChange = lfca
res$significant = ""
res$significant[which(padja <= 0.1)] = "*"
res$significant[which(padja <= 0.01)] = "**"
attr(res,"log2FC") = rowMeans(cbind(R1all,R2all))
attr(res,"log2total") = rowMeans(Dall[,IT])
attr(res,"log2FCsd") = rowSds(cbind(R1all,R2all))
attr(res,"log2TotalSD") = rowSds(Dall[,IT])
```

The R/Bioconductor package limma shrinks the standard deviation of each gene towards a global estimate of standard deviation along all genes. In the case of large outliers, this can result in spurious hits. To avoid these, genes with standard deviation larger than the absolute value of the log-fold change are not regarded as significant.

```
K = which(res$significant != "" & (fita$sigma > abs(fita$coefficients[,1])))

write.table(res[K,], sep="\t", file = file.path(d, "filteredGenes.txt"), quote=FALSE)

res$padj[K] = NA
res$significant[K] = ""
```

```
col=rep("gray", length(padja))
col[which(res$significant == "*" & lfca > 0)] = brewer.pal(4,"RdBu")[2]
col[which(res$significant == "*" & lfca < 0)] = brewer.pal(4,"RdBu")[3]
col[which(res$significant == "**" & lfca > 0)] = brewer.pal(4,"RdBu")[1]
col[which(res$significant == "**" & lfca < 0)] = brewer.pal(4,"RdBu")[4]
plot(rowMeans(D[,SL$Type == "RNAseq"]),lfca,
     pch=20,cex=0.7,col=col,
     xlab="log2 read counts (total RNA)", ylab="translation efficiency [log2]")
abline(h=0)

plot(R1,R2,
     pch=20,cex=0.7,col=col,
     xlab="translation efficiency [log2] (rep. 1)",
     ylab="translation efficiency [log2] (rep. 2)",
     xlim=c(-4,4),ylim=c(-4,4))
```

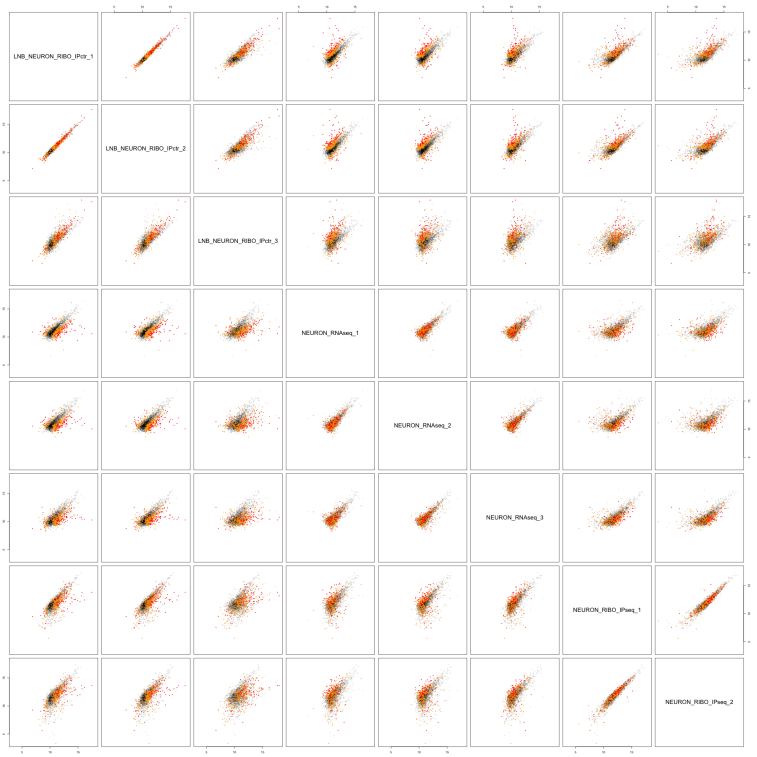
```
abline(h=0,v=0)
text(-3.8,3.8,paste0("R = ",format(cor(R1,R2,use="pairwise.complete"),digits=2)),adj = c(0,1))
```



The data is saved.

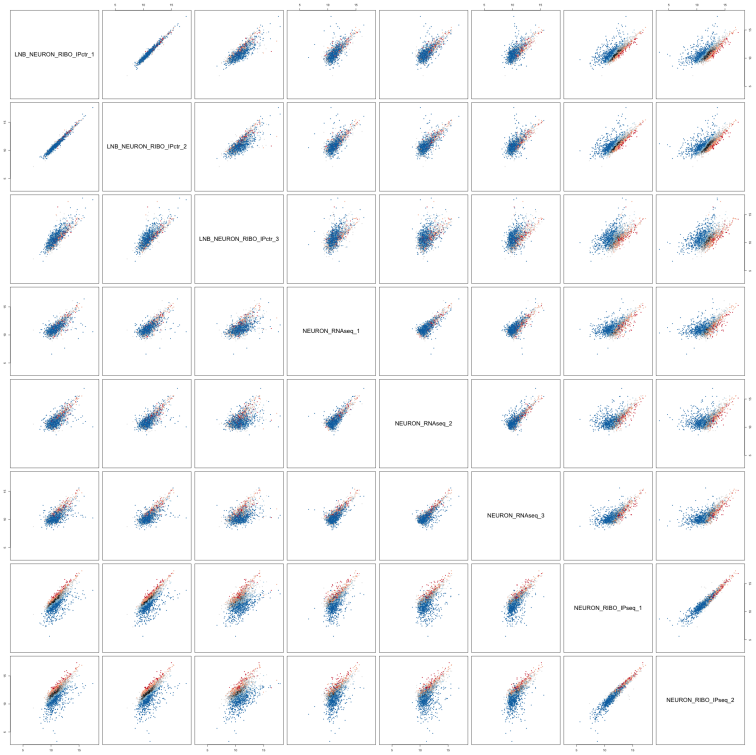
```
save(EC, file=file.path(d, "EC.rda"))
save(res, file=file.path(d, "res.rda"))
save(fitc, file=file.path(d, "fitc.rda"))
save(ExpVar, file=file.path(d, "ExpVar.rda"))
```

```
col = rep("#00000022", dim(C)[1])
col[which(fitc$padj <= 0.1)] = "orange"
col[which(fitc$padj <= 0.01)] = "red"
I = order(factor(col, levels=c("#00000022","orange","red")))
lim = range(as.vector(C))
pairs(C[I,],pch=20,cex=0.5,xlim=lim, ylim=lim,col=col[I])
```



```
col=rep("#00000022", nrow(res))
col[which(res$significant == "*" & res$log2FoldChange > 0)] = brewer.pal(4,"RdBu")[2]
col[which(res$significant == "*" & res$log2FoldChange < 0)] = brewer.pal(4,"RdBu")[3]
col[which(res$significant == "**" & res$log2FoldChange > 0)] = brewer.pal(4,"RdBu")[1]
```

```
col[which(res$significant == "**" & res$log2FoldChange < 0)] = brewer.pal(4,"RdBu")[4]

I = order(factor(col, levels=c("#00000022",brewer.pal(4,"RdBu")[c(2,3,1,4)])))
lim = range(as.vector(C))
pairs(C[I,],pch=20,cex=0.5,xlim=lim, ylim=lim,col=col[I])
```




```
fgup = row.names(res)[which(res$significant != "" & res$log2FoldChange > 0)]
fgdown = row.names(res)[which(res$significant != "" & res$log2FoldChange < 0)]
bg = row.names(res)[which(!is.na(res$padj))]
GSEAup = doGSEA(fgup,bg,ENSG2category, k=3)
GSEAdown = doGSEA(fgdown,bg,ENSG2category, k=3)
```

```
res2 = res[order(factor(res$significant, levels=c("**","*","")),
                 sign(-res$log2FoldChange),res$symbol),]
res2$UnspecBinders = ""
res2$UnspecBinders[row.names(res2) %in% names(UnspecBinders)] = "UnspecBinder"
write.table(res2, file=file.path(d, "resulttable.txt"), sep="\t", quote=FALSE)
page = openPage(file.path(d,"resulttable.html"), link.css="hwriter.css")
hwriteSidebar(page=page, link=mainmenu)
hwrite(sprintf("Translation efficiency (%s); result table",n),
       heading=1, page=page)
hwrite("** = 1% FDR; * = 10% FDR",page=page,br=TRUE)
col = rep("",nrow(res))
col[which(res2$significant %in% c("*","**") & res2$log2FoldChange > 0)] =
  brewer.pal(3,"Pastel1")[1]
col[which(res2$significant %in% c("*","**") & res2$log2FoldChange < 0)] =
  brewer.pal(3,"Pastel1")[2]
hwrite(as.data.frame(res2), row.bgcolor=col, page=page)
closePage(page, splash=FALSE)
```

3.5 Compare neural differentiation stages

Create an output directory.

```
require(grid) # added
d = file.path("result","CompareStages","")
dir.create(d, recursive = TRUE, showWarnings = FALSE)
mainmenu = file.path("..","..","mainmenu.html")

library(grid)
source(file.path("R","myballoonplot.R"))
load(file.path("data","ENSG2category.rda"))
source(file.path("R","doGSEA.R"))
```

```
log2FC = log2total = log2FCsd = log2totalSD = list()
S = names(SamplesTranslationEfficiency)
TransEff = list()
for (s in S) {
  load(file.path("result","TranslationEfficiency",s,"res.rda"))
  TransEff[[s]] = res
  a = attributes(res)
  log2FC[[s]] = a$log2FC
  log2total[[s]] = a$log2total
  log2FCsd[[s]] = a$log2FCsd
  log2totalSD[[s]] = a$log2totalSD
}
log2FC = do.call(cbind, log2FC)
log2total = do.call(cbind, log2total)
log2FCsd = do.call(cbind, log2FCsd)
log2totalSD = do.call(cbind, log2totalSD)
```

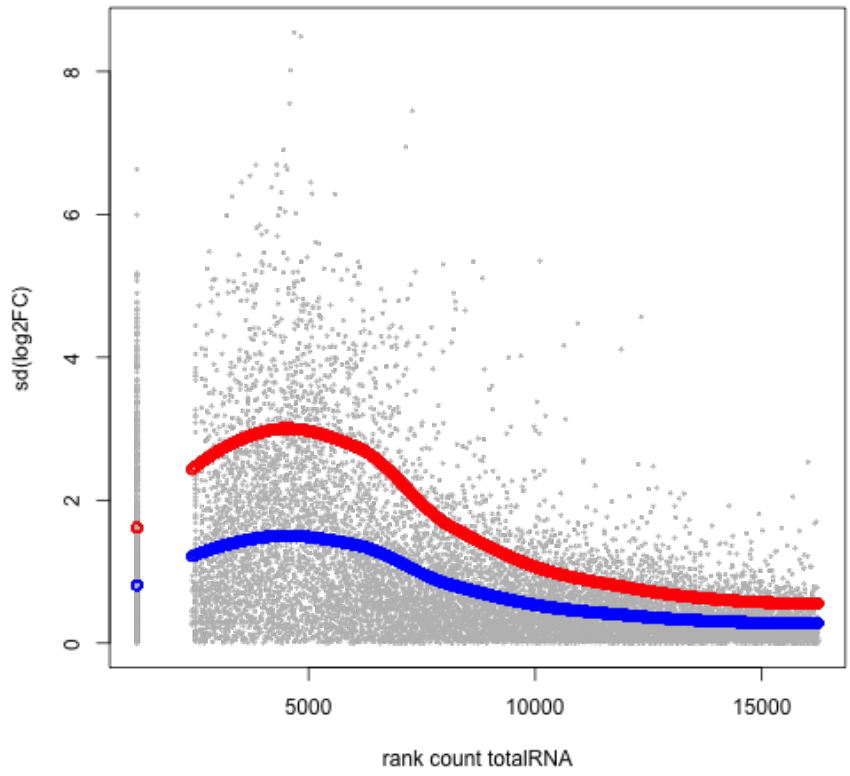
```
for (s in names(TransEff)) {
    TransEff[[s]]$ENSG = row.names(TransEff[[s]])
    TransEff[[s]] = TransEff[[s]][,c("ENSG","symbol","log2FoldChange",
                                     "pval","padj","significant")]
    colnames(TransEff[[s]])[3:6] = paste(s,colnames(TransEff[[s]])[3:6],sep=".")
}

X = merge(TransEff[[1]],TransEff[[2]], by=c("ENSG","symbol"),
          suffixes=c("",""), all=TRUE)
Y = merge(TransEff[[3]],TransEff[[4]], by=c("ENSG","symbol"),
          suffixes=c("",""), all=TRUE)
Z = merge(X,Y, by=c("ENSG","symbol"),suffixes=c("",""), all=TRUE)
write.table(Z, file = file.path(d, "TableS4-TranslationEfficiency.txt"),
            sep="\t", row.names=FALSE, quote=FALSE)
```

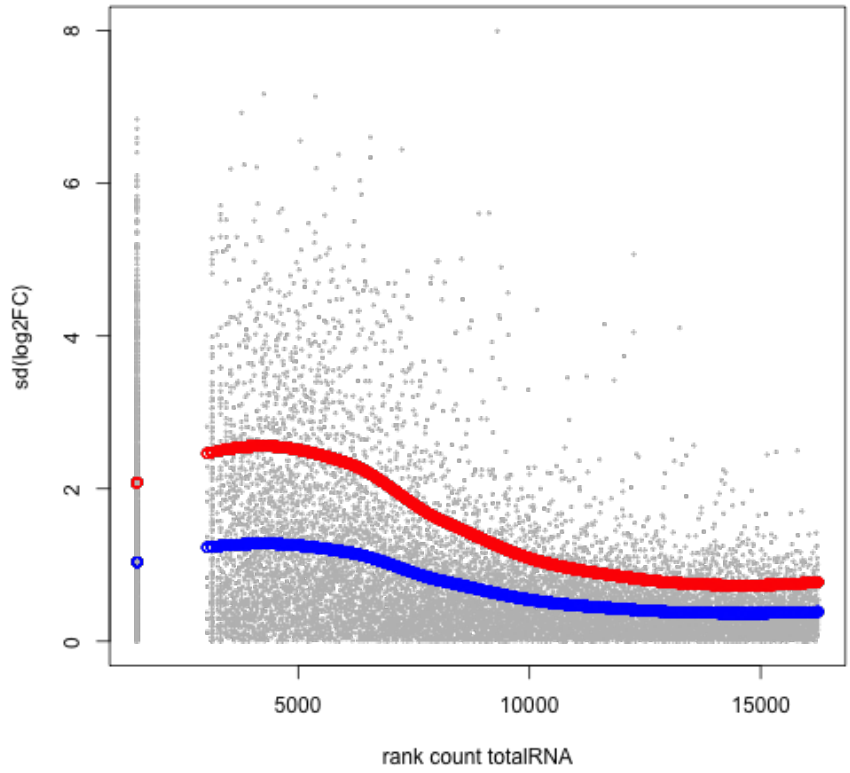
```
I = Anno$transript_type == "protein_coding"
log2FC = log2FC[I,]
log2total = log2total[I,]
log2FCsd = log2FCsd[I,]
log2totalSD = log2totalSD[I,]
log2totalRank = log2total
for (s in S) {
  log2totalRank[,s] = rank(log2total[,s])
}
```

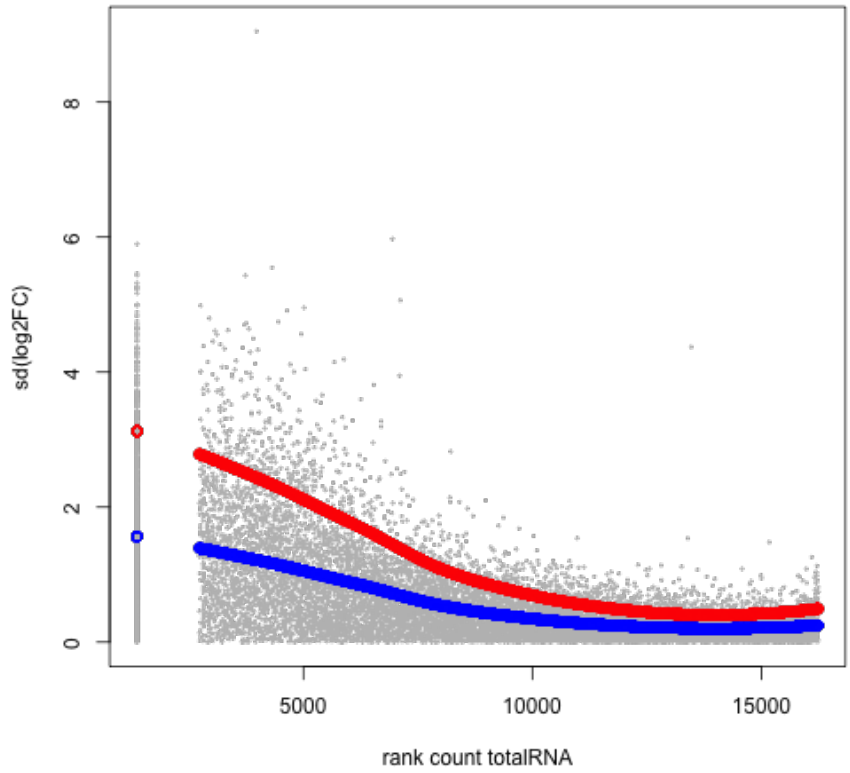
```
log2totalQ = log2total
Q = apply(log2totalQ, 2, quantile, probs=seq(0,1,length.out = 11))
q = apply(Q,1,mean)
for (s in S) {
```

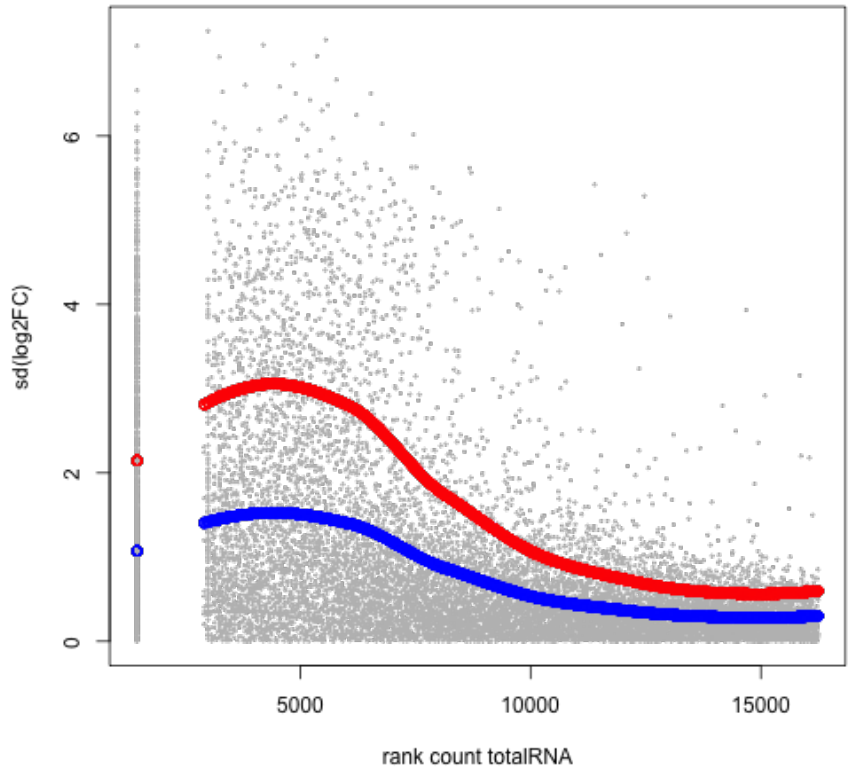
```
  log2totalQ[,s] = approx(x = Q[,s],q, xout=log2totalQ[,s],rule = 2)$y
}
```

```
for (s in S) {
  plot(log2totalRank[,s],log2FCsd[,s],pch=20,cex=0.3,col="grey",
       xlab="rank count totalRNA", ylab="sd(log2FC)")
  fit = loess(log2FCsd[,s] ~ rank(log2total[,s]))
  points(log2totalRank[,s],fit$fitted,col="blue")
  points(log2totalRank[,s],2*fit$fitted,col="red")
  SD = fit$fitted
  J = which(log2FCsd[,s] > 2*fit$fitted)
  SD[J] = log2FCsd[J,s]
  log2FCsd[,s] = log2FCsd[,s] / SD
}
```

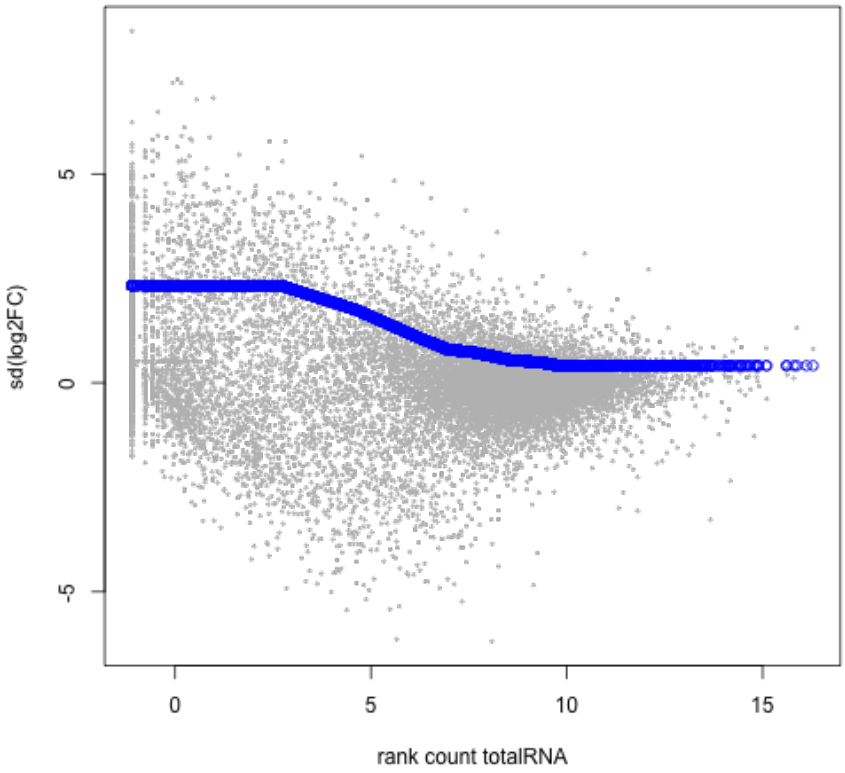


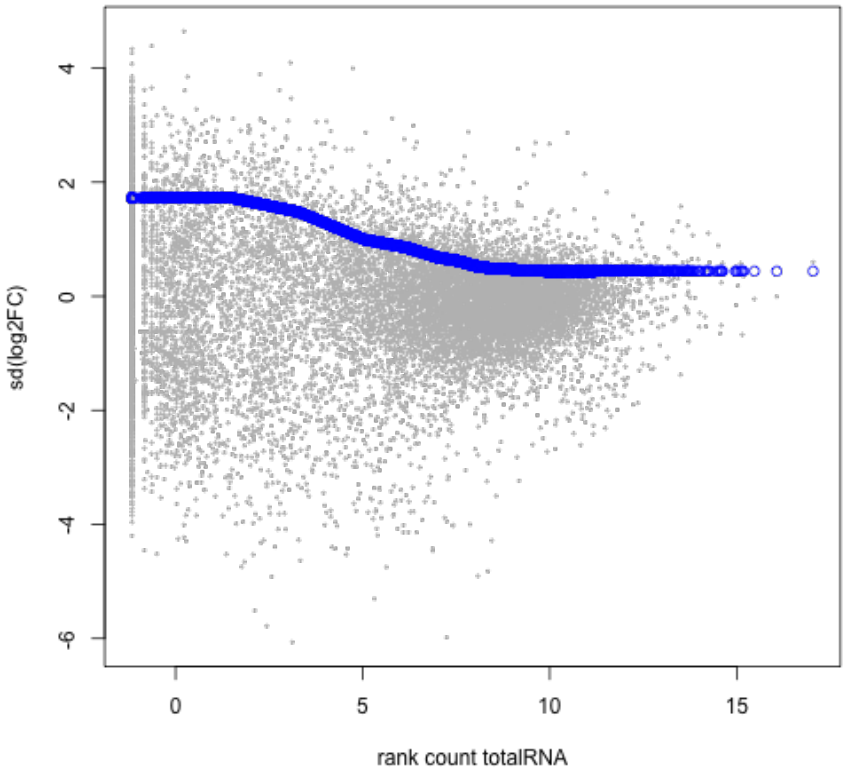

sd(log2FC)
0
2
4
6
8
5000
10000
15000
rank count totalRNA

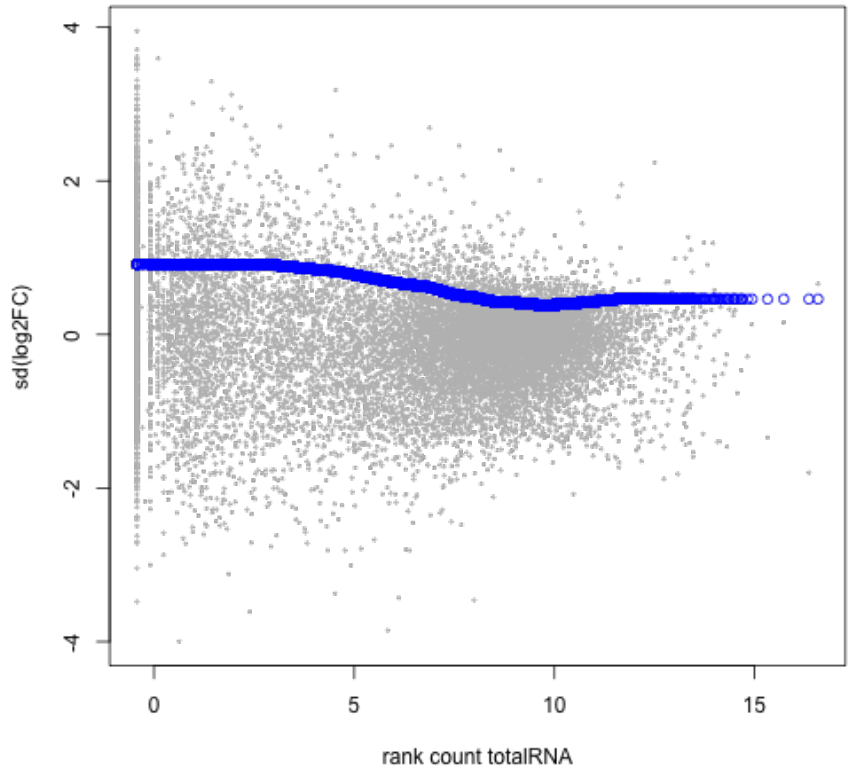

sd(log2FC)
0
2
4
6
8
5000
10000
15000
rank count totalRNA

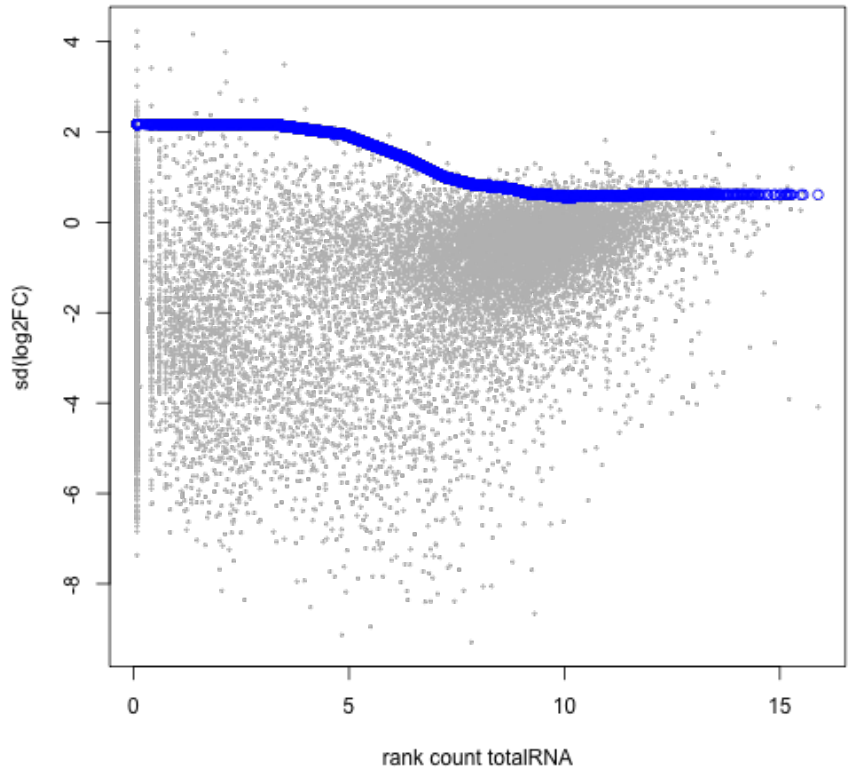



```
for (s in S) {
  b = cut(log2totalRank[,s],breaks=15)
#  M = tapply(log2FC[,s], b, median)
  MD = tapply(log2FC[,s], b, mad)
  x = tapply(log2total[,s], b, mean)
#  M[1:3] = M[4]
  MD[1:3] = MD[4]
#  Mnew = approx(x=x,y=M,xout=log2total[,s],rule = 2)
  MDnew = approx(x=x,y=MD,xout=log2total[,s],rule = 2)
  plot(log2total[,s],log2FC[,s],pch=20,cex=0.3,col="grey",
       xlab="rank count totalRNA", ylab="sd(log2FC)")
#  points(Mnew$x,Mnew$y,col="blue")
  points(MDnew$x,MDnew$y,col="blue")

  log2FC[,s] = log2FC[,s] / MDnew$y
#   log2FC[,s] =
#     (log2FC[,s] - Mnew$y) / MDnew$y

#   plot(log2FC[Annot$transript_type == "protein_coding",s],
#   log2total[Annot$transript_type == "protein_coding",s])
}
```

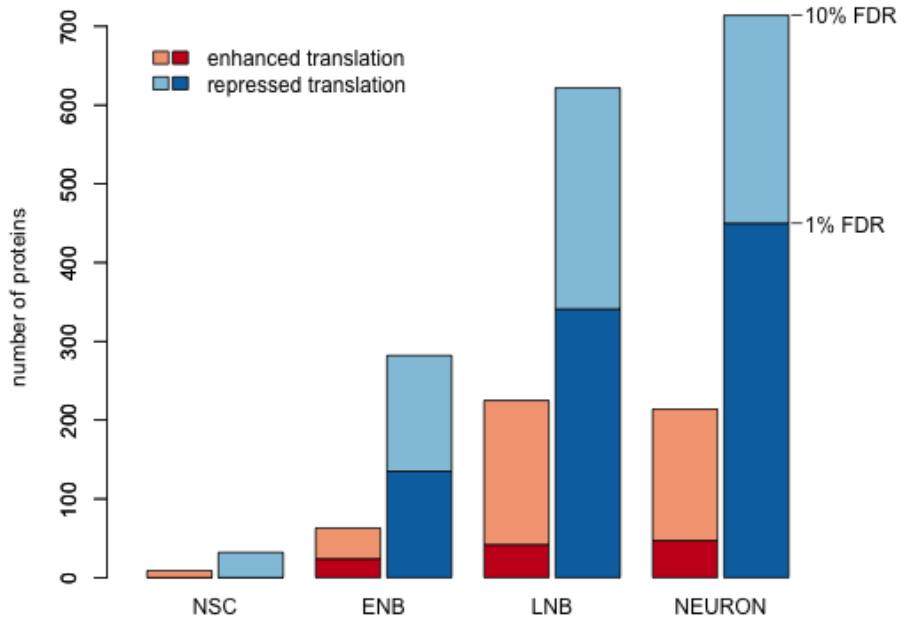

sd(log2FC)
5
0
-5
0
5
10
15
rank count totalRNA


sd(log2FC)
4
2
0
-2
-4
-6
0
5
10
15
rank count totalRNA


sd(log2FC)
4
2
0
-2
-4
0
5
10
15
rank count totalRNA

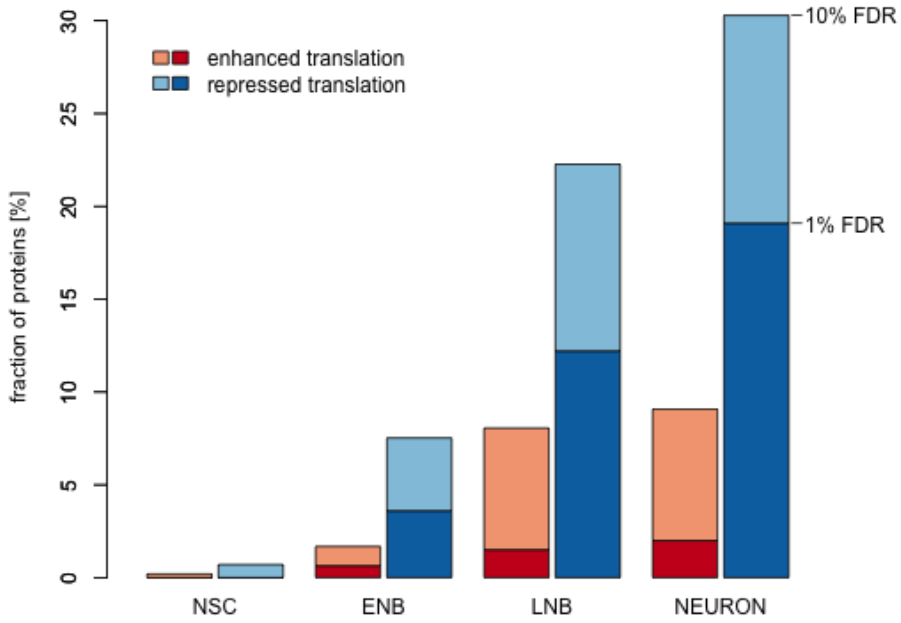

sd(log2FC)
4
2
0
-2
-4
-6
-8
0
5
10
15
rank count totalRNA

```
p = par(xpd=NA,mar=c(1.5,4,1.5,2.5))
H = matrix(NA, nrow=4, ncol=4)
colnames(H) = S
row.names(H) = c("highSigUp","sigUp","sigDown","highSigDown")
n = rep(NA,4)
names(n) = S
for (s in S) {
  load(file.path("result","TranslationEfficiency",s,"res.rda"))
  H["highSigUp",s] = sum(res$significant == "**" & res$log2FoldChange > 0,na.rm=TRUE)
  H["sigUp",s] = sum(res$significant == "*" & res$log2FoldChange > 0,na.rm=TRUE)
  H["sigDown",s] = sum(res$significant == "*" & res$log2FoldChange < 0,na.rm=TRUE)
  H["highSigDown",s] = sum(res$significant == "**" & res$log2FoldChange < 0,na.rm=TRUE)
  n[s] = sum(is.finite(res$padj))
}

col=brewer.pal(4,"RdBu")
ylim=c(0,max(H[c(1,4),]+H[c(2:3),]))
bp1 = barplot(H[1:2,],col=col[1:2],space=c(0.2,1.6,1.6,1.6),names.arg = rep("",4),
              xlim=c(0,10.4),ylim=ylim,ylab="number of proteins")
bp2 = barplot(H[4:3,],col=col[4:3],space=c(1.3,1.6,1.6,1.6),names.arg = rep("",4),add=TRUE)
text((bp1+bp2)/2,rep(-0.05*ylim[2],4),S)
legend(x=0.0,y=0.97*ylim[2],fill=col[c(2:3)],c("",""),bty = "n")
legend(x=0.3,y=0.97*ylim[2],fill=col[c(1,4)],c("enhanced translation","repressed translation"),bty = "n
x = bp2[4]+0.5
y = H["highSigDown","NEURON"]
lines(c(x+0.05,x+0.2),c(y,y))
text(x+0.25,y = y,"1% FDR",adj = c(0,0.5))
y = H["sigDown","NEURON"] + H["highSigDown","NEURON"]
lines(c(x+0.05,x+0.2),c(y,y))
text(x+0.25,y = y,"10% FDR",adj = c(0,0.5))
```




```
for (i in 1:4) {
    H[,i] = H[,i] / n[i]
}
H = H*100

ylim=c(0,max(H[c(1,4),]+H[c(2:3),]))
```

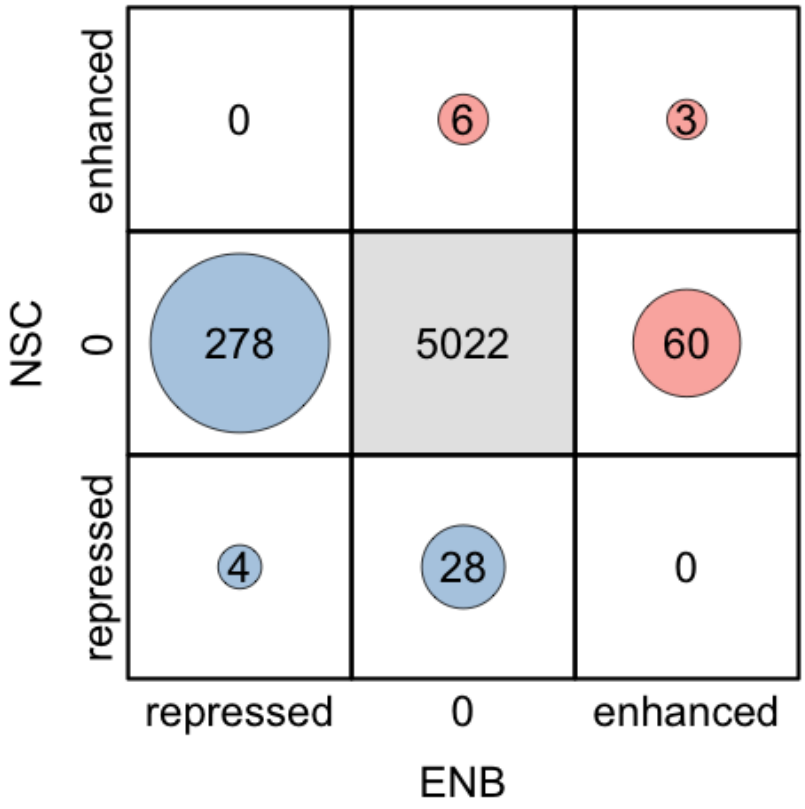
```
bp1 = barplot(H[1:2,],col=col[1:2],space=c(0.2,1.6,1.6,1.6),names.arg = rep("",4),
              xlim=c(0,10.4),ylim=ylim,ylab="fraction of proteins [%]")
bp2 = barplot(H[4:3,],col=col[4:3],space=c(1.3,1.6,1.6,1.6),names.arg = rep("",4),add=TRUE)
text((bp1+bp2)/2,rep(-0.05*ylim[2],4),S)
legend(x=0.0,y=0.97*ylim[2],fill=col[c(2:3)],c("",""),bty = "n")
legend(x=0.3,y=0.97*ylim[2],fill=col[c(1,4)],c("enhanced translation","repressed translation"),bty = "n
x = bp2[4]+0.5
y = H["highSigDown","NEURON"]
lines(c(x+0.05,x+0.2),c(y,y))
text(x+0.25,y = y,"1% FDR",adj = c(0,0.5))
y = H["sigDown","NEURON"] + H["highSigDown","NEURON"]
lines(c(x+0.05,x+0.2),c(y,y))
text(x+0.25,y = y,"10% FDR",adj = c(0,0.5))
```



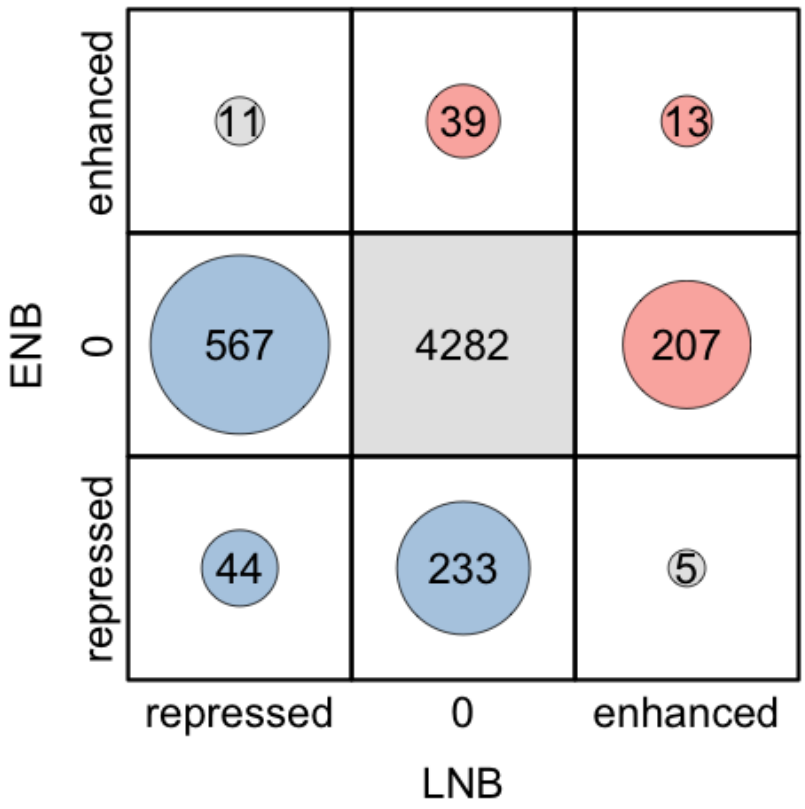

```
par(p)
```

```
ENSG = c()
for (s in S) {
  load(file.path("result","TranslationEfficiency",s,"res.rda"))
  ENSG = c(ENSG, row.names(res))
}
ENSG = sort(unique(ENSG))
X = matrix(factor("0",levels=c("repressed","0","enhanced")), nrow = length(ENSG), ncol=4)
row.names(X) = ENSG
colnames(X) = S
for (s in S) {
  load(file.path("result","TranslationEfficiency",s,"res.rda"))
  a = res$significant
  a[a == "**"] = "enhanced"
  a[a == "*"] = "enhanced"
  a[res$log2FoldChange < 0 & a == "enhanced"] = "repressed"
  a[a == ""] = "0"
  X[row.names(res),s] = a
}

t1 = table("NSC" = factor(X[,1],levels=c("repressed","0","enhanced")),
           "ENB"=factor(X[,2],levels=c("repressed","0","enhanced")))
t2 = table("ENB" = factor(X[,2],levels=c("repressed","0","enhanced")),
```

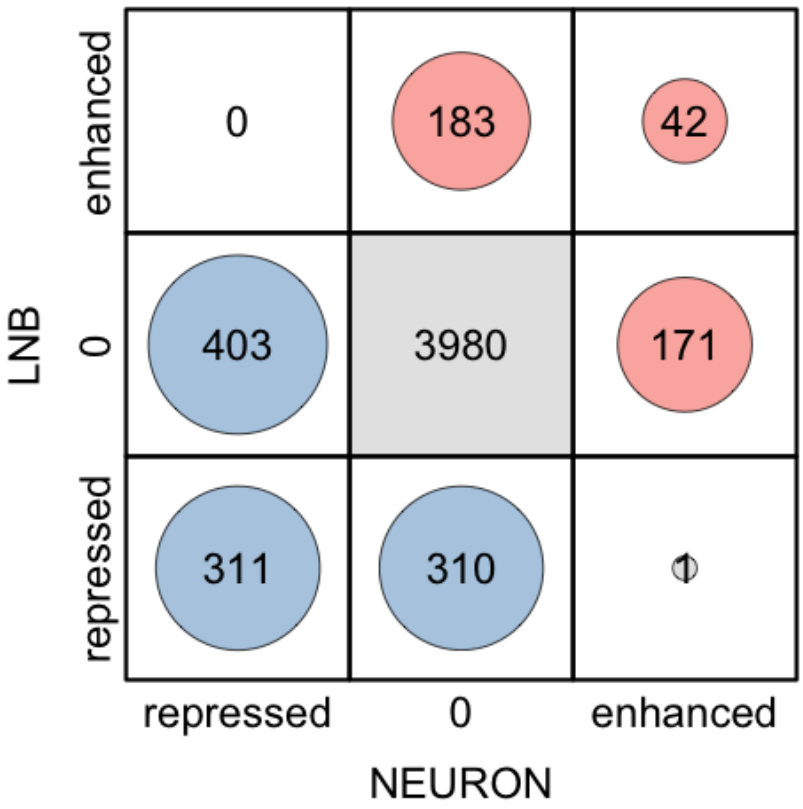
```
           "LNB"=factor(X[,3],levels=c("repressed","0","enhanced")))
t3 = table("LNB" = factor(X[,3],levels=c("repressed","0","enhanced")),
           "NEURON"=factor(X[,4],levels=c("repressed","0","enhanced")))

col = matrix("gray90", nrow=3, ncol=3)
col[1,1] = col[2,1] = col[1,2] = brewer.pal(9, "Pastel1")[2]
col[3,3] = col[2,3] = col[3,2] = brewer.pal(9, "Pastel1")[1]
myballoonplot(t1, col = col, cex = 2)
```




```
grid.newpage()
myballoonplot(t2, col = col, cex = 2)
```




```
grid.newpage()
myballoonplot(t3, col = col, cex = 2)
```



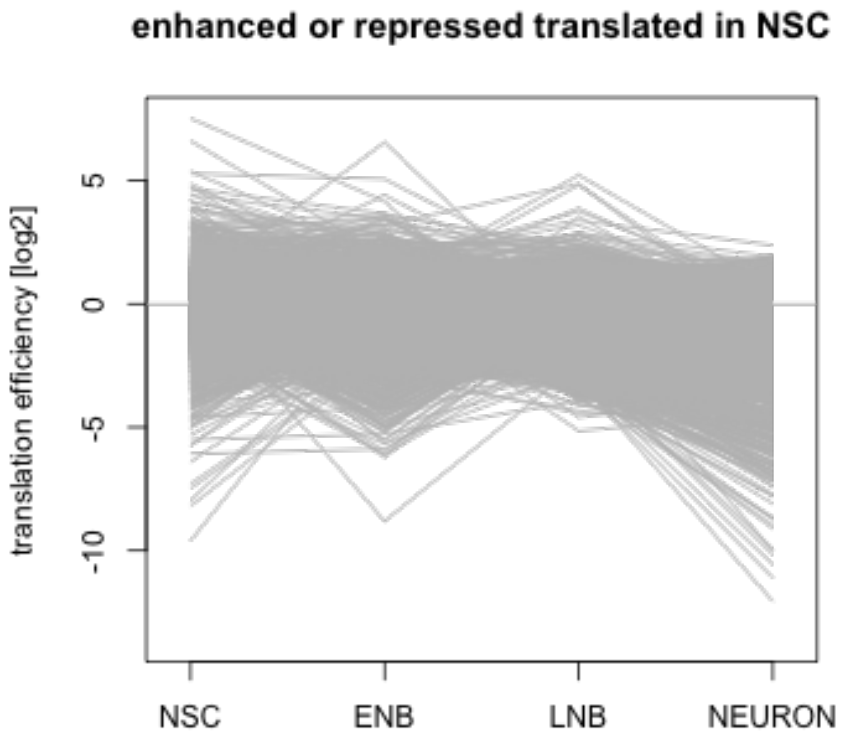

```
fgdown = row.names(X)[which(X[,3] == "repressed" & X[,4] == "repressed")]
fgup = row.names(X)[which(X[,3] == "enhanced" & X[,4] == "enhanced")]
load(file.path("result","TranslationEfficiency","LNB","res.rda"))
bg = row.names(res)
load(file.path("result","TranslationEfficiency","NEURON","res.rda"))
bg = bg[bg %in% row.names(res)]
GSEAup = doGSEA(fgup,bg,ENSG2category, k=3)
GSEAdown = doGSEA(fgdown,bg,ENSG2category, k=3)
```

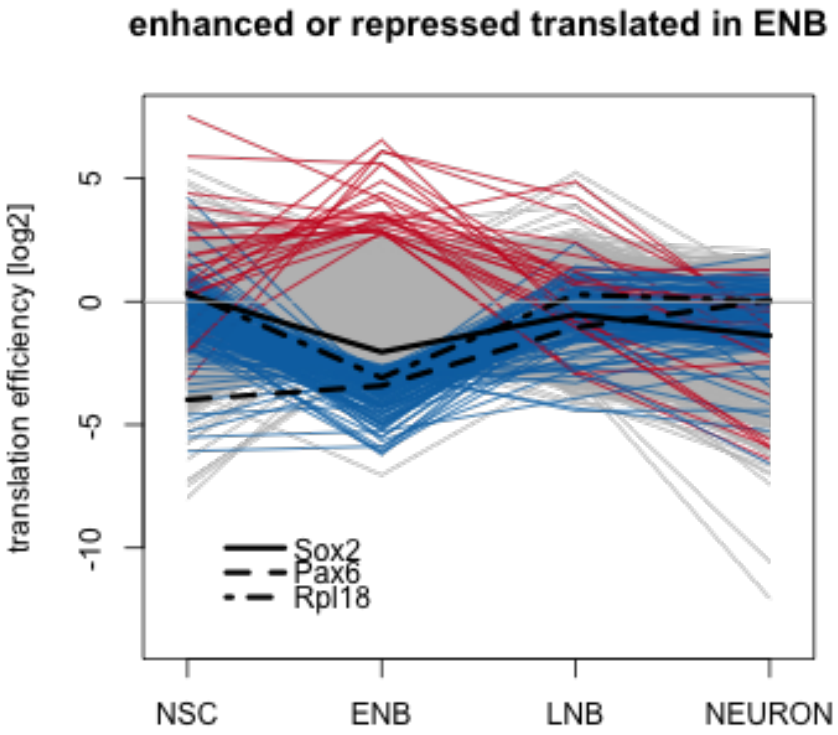
```
library(cluster)
colG = colH = G = H = list()
for (s in S) {
  load(file.path("result","TranslationEfficiency",s,"res.rda"))
  K = rep(3,nrow(res))
  K[which(res$significant == "*" & res$log2FoldChange > 0)] = 4
  K[which(res$significant == "**" & res$log2FoldChange > 0)] = 5
  K[which(res$significant == "*" & res$log2FoldChange < 0)] = 2
  K[which(res$significant == "**" & res$log2FoldChange < 0)] = 1

  G[[s]] = row.names(res)[which(K == 5)]
  H[[s]] = row.names(res)[which(K == 1)]

  # X = cbind(log2FC,log2totalQ)
  # for (i in seq_len(ncol(X))) {
  #   X[,i] = X[,i] - median(X[,i])
  #   X[,i] = X[,i] / mad(X[,i])
  # }
  #
  # if (length(G[[s]]) >= 10) {
  #   CL<-1:5
```

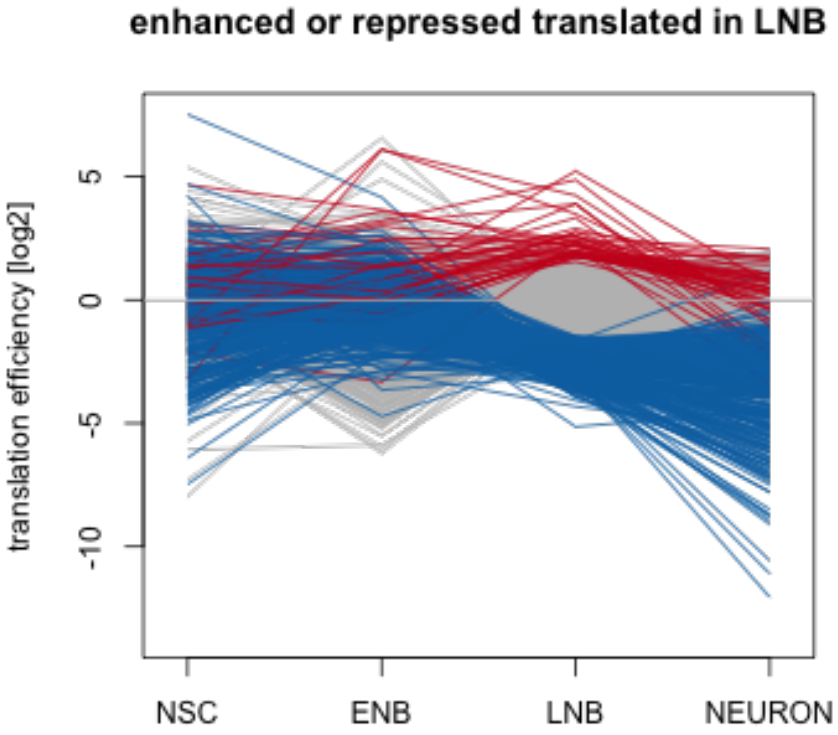
```
  #   avgSil<-rep(NA, length(CL))
  #   for(cl in CL[2:length(CL)])
  #     avgSil[cl]<-pam(X[G[[s]],], k=cl)$silinfo$avg.width
  #   barplot(avgSil, names.arg=CL, xlab="Number of clusters, K", ylab="Average silhoue",main=s)
  #   CL = which.max(avgSil)
  #   cl = pam(X[G[[s]],], k=CL)
  #   colG[[s]] = rainbow(CL)[cl$clustering]
  #   names(colG[[s]]) = G[[s]]
  # } else {
  #   colG[[s]] = rep("blue", length(G[[s]]))
  #   names(colG[[s]]) = G[[s]]
  # }
  #
  # if (length(H[[s]]) >= 10) {
  #   CL<-1:5
  #   avgSil<-rep(NA, length(CL))
  #   for(cl in CL[2:length(CL)])
  #     avgSil[cl]<-pam(X[H[[s]],], k=cl)$silinfo$avg.width
  #   barplot(avgSil, names.arg=CL, xlab="Number of clusters, K", ylab="Average silhoue",main=s)
  #   CL = which.max(avgSil)
  #   cl = pam(X[H[[s]],], k=CL)
  #   colH[[s]] = rainbow(CL)[cl$clustering]
  #   names(colH[[s]]) = H[[s]]
  # } else {
  #   colH[[s]] = rep("blue", length(H[[s]]))
  #   names(colH[[s]]) = H[[s]]
  # }
}
```

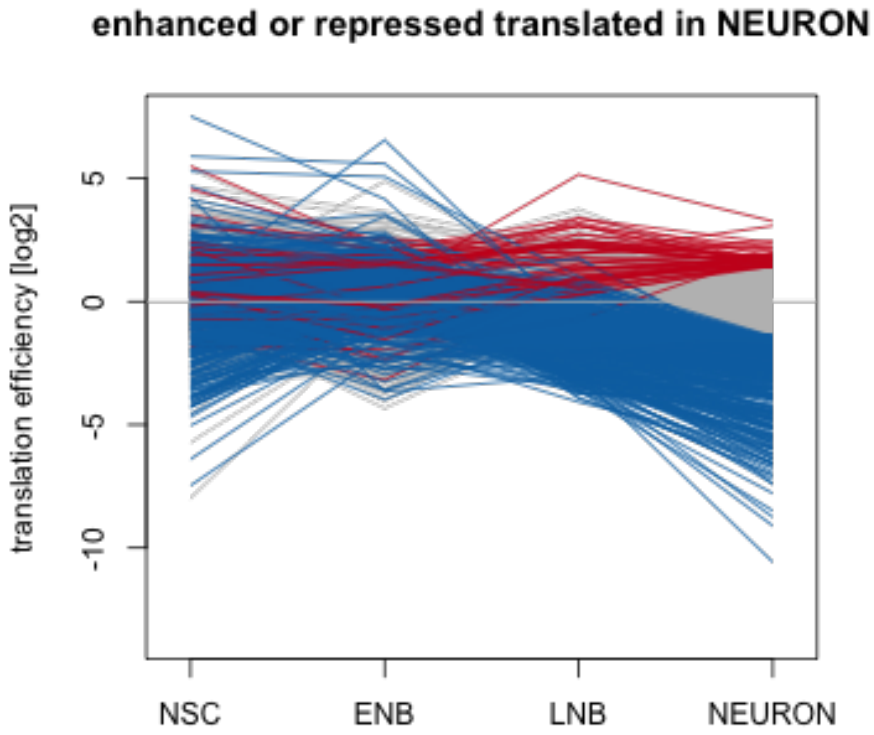
```
ylim = range(log2FC)
for (s in S) {
  load(file.path("result","TranslationEfficiency",s,"res.rda"))

  plot(-100000,xlim=c(0.9,4.1), ylim=ylim,xaxt="n",
       main=sprintf("enhanced or repressed translated in %s",s),
       xlab="",ylab="translation efficiency [log2]")
  axis(side=1,1:4,colnames(log2total))
  col = rep("gray", nrow(log2FC))
  names(col) = row.names(log2FC)
  col[H[[s]]] = brewer.pal(4,"RdBu")[4]
  col[G[[s]]] = brewer.pal(4,"RdBu")[1]
  K = rep(1, nrow(log2FC))
  names(K) = row.names(log2FC)
  K[c(G[[s]],H[[s]])] = 2
  I = sample(row.names(res))
  I = I[order(K[I])]
  for (k in I) {
    lines(log2FC[k,],col=col[k])
  }
  if (s == "ENB") {
      if ("ENSMUSG00000074637" %in% rownames(log2FC)) {
          lines(log2FC["ENSMUSG00000074637",],col="black",lwd=3) # Sox2
      }
      if ("ENSMUSG00000027168" %in% rownames(log2FC)) {
          lines(log2FC["ENSMUSG00000027168",],col="black",lwd=3, lty="dashed") # Pax6
      }
      if ("ENSMUSG00000059070" %in% rownames(log2FC)) {
          lines(log2FC["ENSMUSG00000059070",],col="black",lwd=3, lty="dotdash") # Rpl18
```

```
      }
      if (s == "ENB") {
          lines(c(1.2,1.5),c(-10,-10),col="black",lwd=3) # Sox2
          lines(c(1.2,1.5),c(-11,-11),col="black",lwd=3, lty="dashed") # Pax6
          lines(c(1.2,1.5),c(-12,-12),col="black",lwd=3, lty="dotdash") # Rpl18
          text(1.55,-10,"Sox2", adj = c(0,0.5))
          text(1.55,-11,"Pax6", adj = c(0,0.5))
          text(1.55,-12,"Rpl18", adj = c(0,0.5))
      }
  }
  abline(h=0,col="gray")
}
```

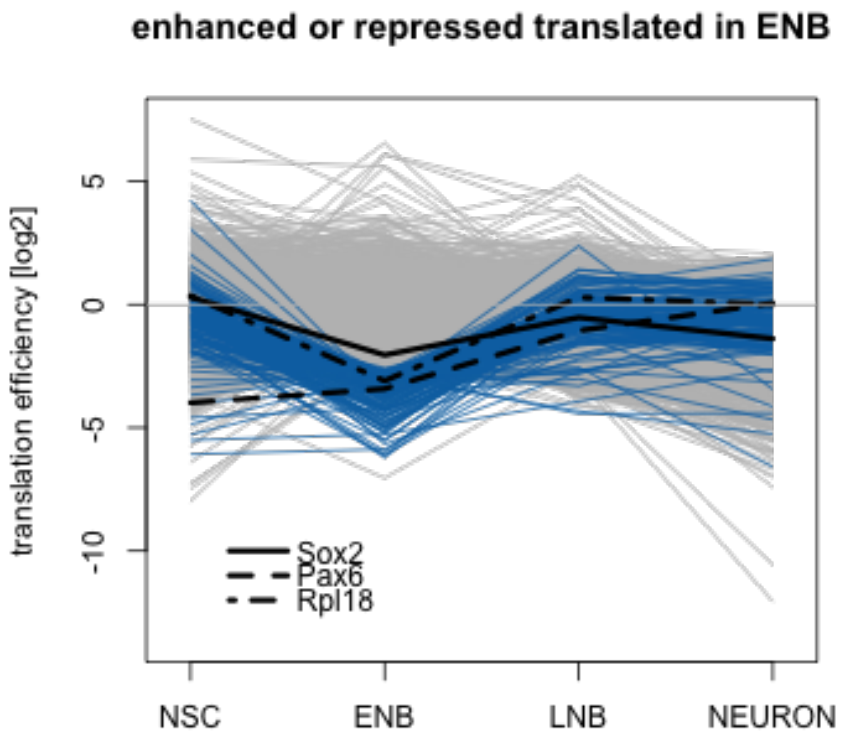



enhanced or repressed translated in ENB
translation efficiency [log2]
5
0
-5
-10
Sox2
Pax6
Rpl18
NSC
ENB
LNB
NEURON


enhanced or repressed translated in LNB
translation efficiency [log2]
5
0
-5
-10
NSC
ENB
LNB
NEURON

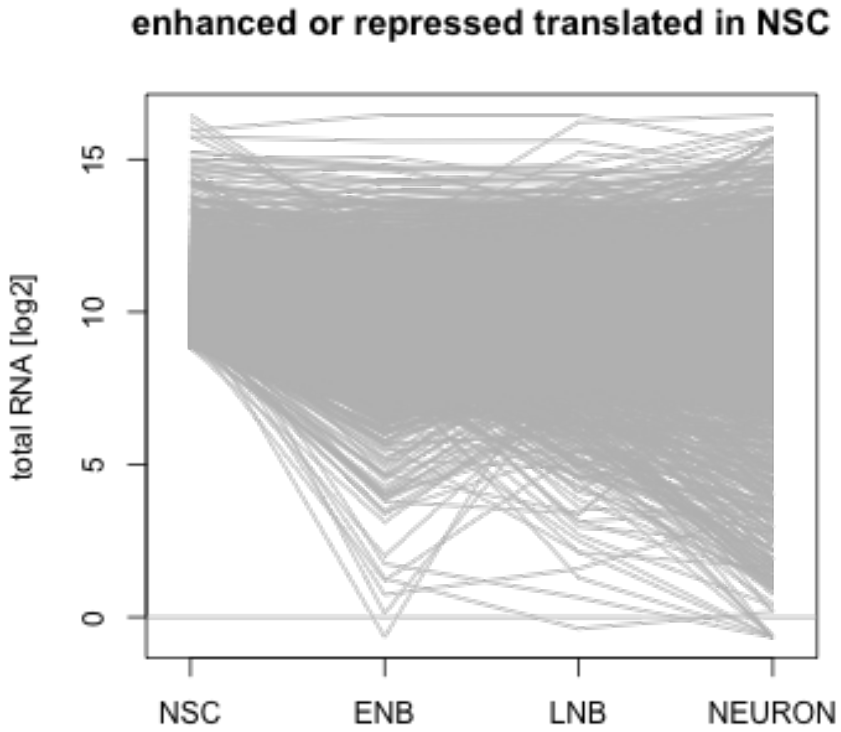


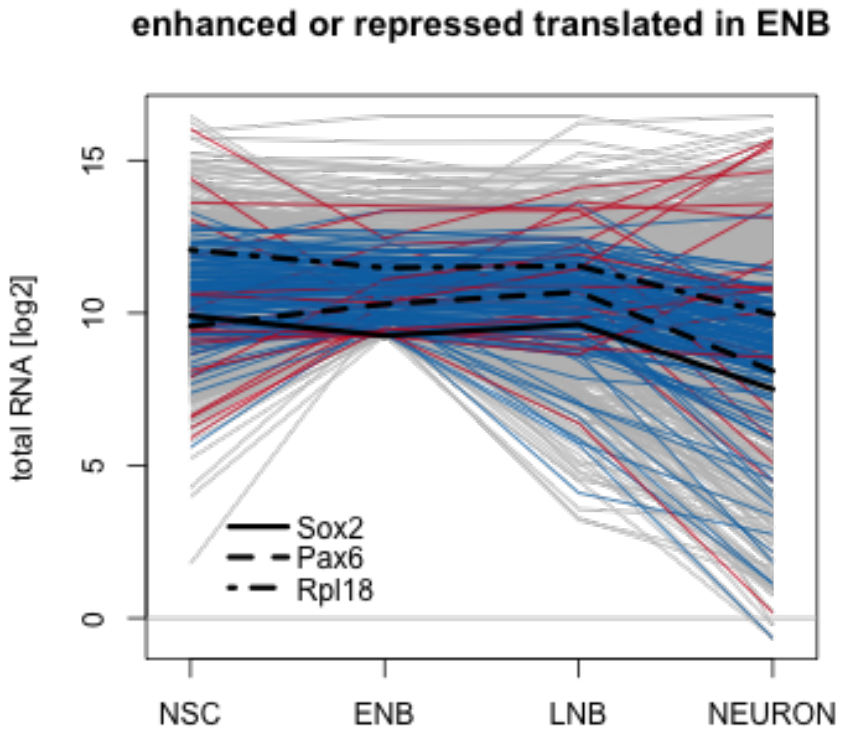

```
ylim = range(log2FC)
s = "ENB"
load(file.path("result","TranslationEfficiency",s,"res.rda"))

plot(-100000,xlim=c(0.9,4.1), ylim=ylim,xaxt="n",
   main=sprintf("enhanced or repressed translated in %s",s),
   xlab="",ylab="translation efficiency [log2]")
axis(side=1,1:4,colnames(log2total))
col = rep("gray", nrow(log2FC))
names(col) = row.names(log2FC)
col[H[[s]]] = brewer.pal(4,"RdBu")[4]
#col[G[[s]]] = brewer.pal(4,"RdBu")[1]
K = rep(1, nrow(log2FC))
names(K) = row.names(log2FC)
K[c(H[[s]])] = 2
I = sample(row.names(res))
I = I[order(K[I])]
for (k in I) {
  lines(log2FC[k,],col=col[k])
}
lines(log2FC["ENSMUSG00000074637",],col="black",lwd=3) # Sox2
lines(log2FC["ENSMUSG00000027168",],col="black",lwd=3, lty="dashed") # Pax6
lines(log2FC["ENSMUSG00000059070",],col="black",lwd=3, lty="dotdash") # Rpl18
lines(c(1.2,1.5),c(-10,-10),col="black",lwd=3) # Sox2
lines(c(1.2,1.5),c(-11,-11),col="black",lwd=3, lty="dashed") # Pax6
lines(c(1.2,1.5),c(-12,-12),col="black",lwd=3, lty="dotdash") # Rpl18
text(1.55,-10,"Sox2", adj = c(0,0.5))
text(1.55,-11,"Pax6", adj = c(0,0.5))
text(1.55,-12,"Rpl18", adj = c(0,0.5))
abline(h=0,col="gray")
```

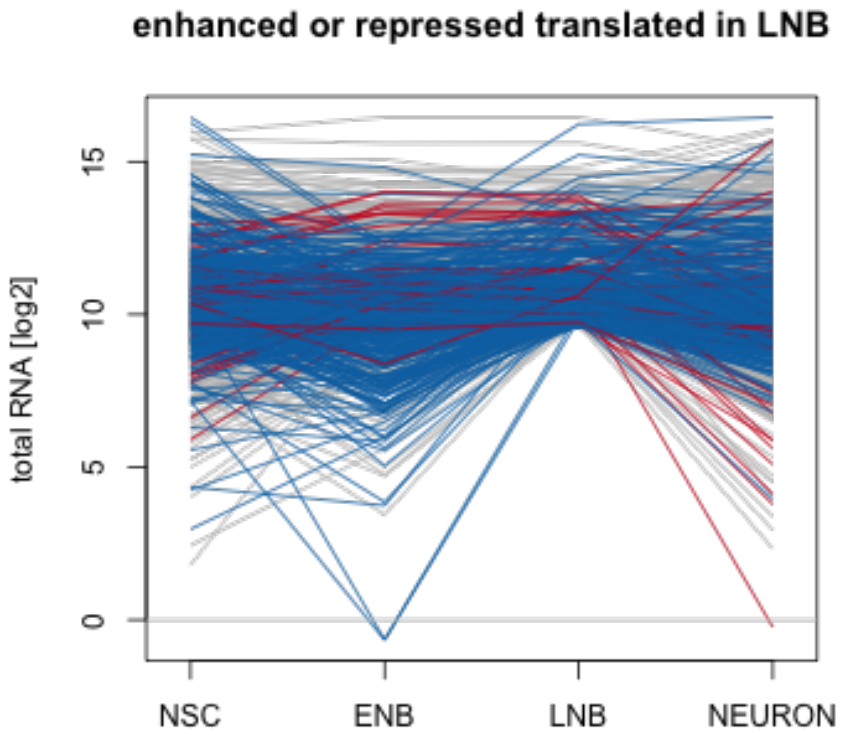


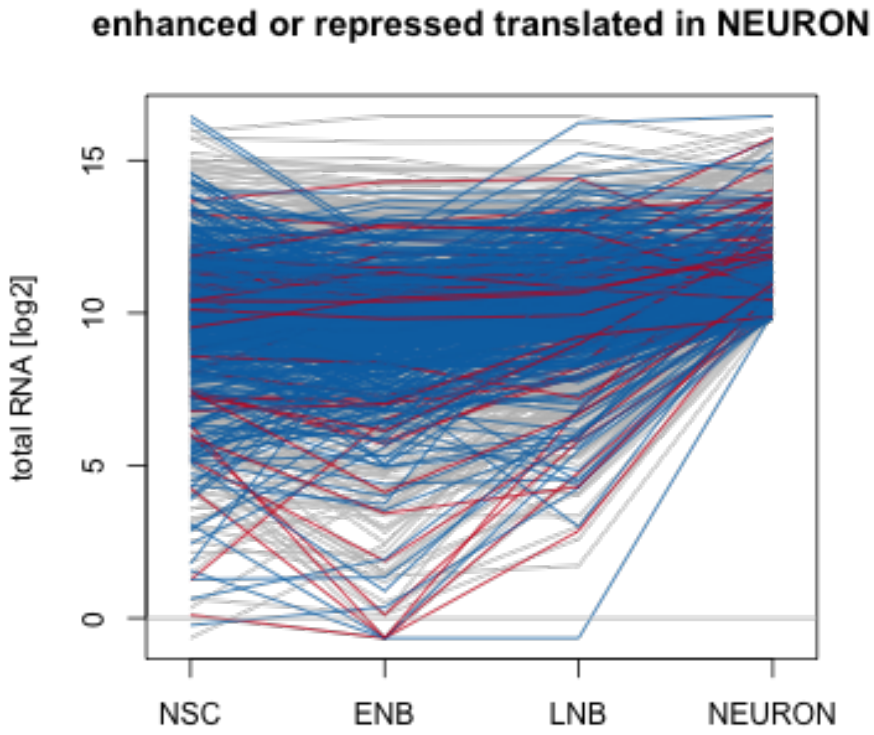

```
ylim = range(log2totalQ)
for (s in S) {
  load(file.path("result","TranslationEfficiency",s,"res.rda"))

  plot(-100000,xlim=c(0.9,4.1), ylim=ylim,xaxt="n",
       main=sprintf("enhanced or repressed translated in %s",s),
       xlab="",ylab="total RNA [log2]")
  axis(side=1,1:4,colnames(log2total))
  col = rep("gray", nrow(log2FC))
  names(col) = row.names(log2FC)
  col[H[[s]]] = brewer.pal(4,"RdBu")[4]
  col[G[[s]]] = brewer.pal(4,"RdBu")[1]
  K = rep(1, nrow(log2FC))
  names(K) = row.names(log2FC)
  K[c(G[[s]],H[[s]])] = 2
  I = sample(row.names(res))
  I = I[order(K[I])]
  for (k in I) {
    lines(log2totalQ[k,],col=col[k])
  }
  if (s == "ENB") {
      if ("ENSMUSG00000074637" %in% rownames(log2totalQ)) {
          lines(log2totalQ["ENSMUSG00000074637",],col="black",lwd=3) # Sox2
      }
      if ("ENSMUSG00000027168" %in% rownames(log2totalQ)) {
          lines(log2totalQ["ENSMUSG00000027168",],col="black",lwd=3, lty="dashed") # Pax6
      }
      if ("ENSMUSG00000059070" %in% rownames(log2totalQ)) {
```

```
        lines(log2totalQ["ENSMUSG00000059070",],col="black",lwd=3, lty="dotdash") # Rpl18
    }
    if (s == "ENB") {
        lines(c(1.2,1.5),c(3,3),col="black",lwd=3) # Sox2
        lines(c(1.2,1.5),c(2,2),col="black",lwd=3, lty="dashed") # Pax6
        lines(c(1.2,1.5),c(1,1),col="black",lwd=3, lty="dotdash") # Rpl18
        text(1.55,3,"Sox2", adj = c(0,0.5))
        text(1.55,2,"Pax6", adj = c(0,0.5))
        text(1.55,1,"Rpl18", adj = c(0,0.5))
    }
  }
  abline(h=0,col="gray")
}
```



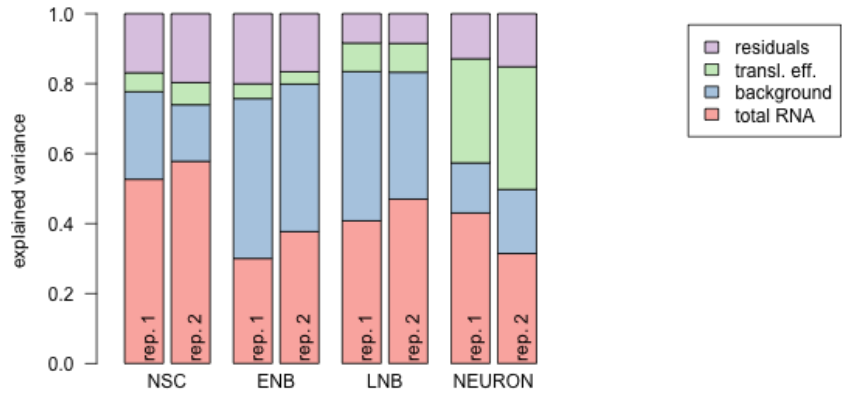

enhanced or repressed translated in ENB
total RNA [log2]
0
5
10
15
Sox2
Pax6
Rpl18
NSC
ENB
LNB
NEURON


enhanced or repressed translated in LNB
total RNA [log2]
0
5
10
15
NSC
ENB
LNB
NEURON

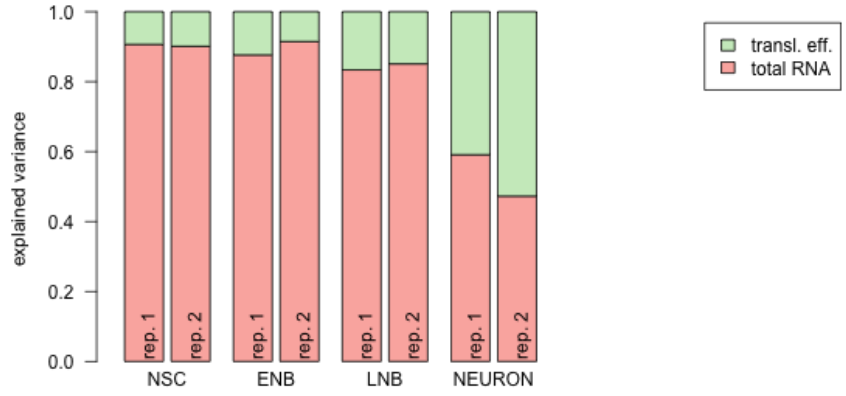


The explained variance is compared along the four differential stages.

```
AllExpVar = NULL
for (s in S) {
    load(file.path("result","TranslationEfficiency",s,"ExpVar.rda"))
#    row.names(ExpVar) = paste0(s, "rep. ", 1:2)
    AllExpVar = rbind(AllExpVar, ExpVar)
}

row.names(AllExpVar) = NULL

p = par(xpd = NA)
bp = barplot(t(AllExpVar), col=brewer.pal(4, "Pastel1"), las=2, space = c(0.0, 0.2, 0.6, 0.2, 0.6, 0.2,
text((bp[seq(1,8,by=2)]+bp[seq(2,8,by=2)])/2, y = -0.05, S)
text(bp[seq(1,8,by=2)], y = 0.01, "rep. 1", adj=c(0,0.8), srt=90)
text(bp[seq(2,8,by=2)], y = 0.01, "rep. 2", adj=c(0,0.8), srt=90)
```




```
par(p)

AllExpVar = AllExpVar[,c(1,3)]
s = rowSums(AllExpVar)
AllExpVar[,1] = AllExpVar[,1] / s
AllExpVar[,2] = AllExpVar[,2] / s

p = par(xpd = NA)
bp = barplot(t(AllExpVar), col=brewer.pal(4, "Pastel1")[c(1,3)], las=2, space = c(0.0, 0.2, 0.6, 0.2, 0
text((bp[seq(1,8,by=2)]+bp[seq(2,8,by=2)])/2, y = -0.05, S)
text(bp[seq(1,8,by=2)], y = 0.01, "rep. 1", adj=c(0,0.8), srt=90)
text(bp[seq(2,8,by=2)], y = 0.01, "rep. 2", adj=c(0,0.8), srt=90)
```




```
par(p)
```

3.6 Compare RNAseq and RiboTag data

Create an output directory.

```
d = file.path("result","CompareRNAseqRiboTag","")
dir.create(d, recursive = TRUE, showWarnings = FALSE)
```

```
mainmenu = file.path("..","..","mainmenu.html")

library(grid)
source(file.path("R","myballoonplot.R"))
load(file.path("data","ENSG2category.rda"))
source(file.path("R","doGSEA.R"))
```

```
RNAseq = RiboTag = list()
S = names(SamplesRNAseq)
for (s in S) {
  load(file.path("result","RNAseq",s,"res.rda"))
  RNAseq[[s]] = res
  load(file.path("result","RIBO_IPseq",s,"res.rda"))
  RiboTag[[s]] = res
}
```

```
DFa = data.frame(ENSG = row.names(RNAseq[[1]]),symbol=RNAseq[[1]][,"symbol"],
                 stringsAsFactors = FALSE)
DF1 = data.frame(RNAseq[[1]][,c("baseMean","log2FoldChange","pvalue","padj","significant")],
                 stringsAsFactors = FALSE)
colnames(DF1) = paste(S[1],colnames(DF1),sep=".")
DF2 = data.frame(RNAseq[[2]][,c("baseMean","log2FoldChange","pvalue","padj","significant")],
                 stringsAsFactors = FALSE)
colnames(DF2) = paste(S[2],colnames(DF2),sep=".")
DF3 = data.frame(RNAseq[[3]][,c("baseMean","log2FoldChange","pvalue","padj","significant")],
                 stringsAsFactors = FALSE)
colnames(DF3) = paste(S[3],colnames(DF3),sep=".")
DF = cbind(DFa,DF1,DF2,DF3)
DF = DF[order(DF$ENSG),]
write.table(DF, file = file.path(d, "TableS2-RNAseq.txt"), sep="\t", row.names=FALSE, quote=FALSE)

DFa = data.frame(ENSG = row.names(RiboTag[[1]]),symbol=RiboTag[[1]][,"symbol"],
                 stringsAsFactors = FALSE)
DF1 = data.frame(RiboTag[[1]][,c("baseMean","log2FoldChange","pvalue","padj","significant")],
                 stringsAsFactors = FALSE)
colnames(DF1) = paste(S[1],colnames(DF1),sep=".")
DF2 = data.frame(RiboTag[[2]][,c("baseMean","log2FoldChange","pvalue","padj","significant")],
                 stringsAsFactors = FALSE)
colnames(DF2) = paste(S[2],colnames(DF2),sep=".")
DF3 = data.frame(RiboTag[[3]][,c("baseMean","log2FoldChange","pvalue","padj","significant")],
                 stringsAsFactors = FALSE)
colnames(DF3) = paste(S[3],colnames(DF3),sep=".")
DF = cbind(DFa,DF1,DF2,DF3)
DF = DF[order(DF$ENSG),]
write.table(DF, file = file.path(d, "TableS3-RiboTag.txt"), sep="\t", row.names=FALSE, quote=FALSE)
```

```
GSEA = list(up = list(
    both = list(),
    RNAspecific = list(),
    RiboTagSpecific = list()
))
GSEA$down = GSEA$up
Nhits = list()
for (s in S) {
    Nhits[[s]] = matrix(NA_integer_, nrow=2, ncol = 3,
                        dimnames=list(c("up","down"),
                                      c("RNAspecific","both","RiboTagSpecific")))
```

```
    I = which(log2(RNAseq[[s]]$baseMean) > 2 | log2(RiboTag[[s]]$baseMean) > 2)
    R = RNAseq[[s]][I,]
    RT = RiboTag[[s]][I,]
    bg = row.names(R)

    J = which(R$padj <= 0.1 & R$log2FoldChange > 0 &
                    RT$padj <= 0.1 & RT$log2FoldChange > 0)
    fgUp = bg[J]
    K = which(R$padj <= 0.1 & R$log2FoldChange < 0 &
                     RT$padj <= 0.1 & RT$log2FoldChange < 0)
    fgDown = bg[K]
    GSEA$up$both[[s]] = doGSEA(fgUp,bg,ENSG2category, k=3)
    GSEA$down$both[[s]] = doGSEA(fgDown,bg,ENSG2category, k=3)
    Nhits[[s]]["up","both"] = length(J)
    Nhits[[s]]["down","both"] = length(K)

    J2 = which(R$padj <= 0.1 & R$log2FoldChange > 0)
    J2 = setdiff(J2, J)
    fgUp = bg[J2]
    K2 = which(R$padj <= 0.1 & R$log2FoldChange < 0)
    K2 = setdiff(K2, K)
    fgDown = bg[K2]
    GSEA$up$RNAspecific[[s]] = doGSEA(fgUp,bg,ENSG2category, k=3)
    GSEA$down$RNAspecific[[s]] = doGSEA(fgDown,bg,ENSG2category, k=3)
    Nhits[[s]]["up","RNAspecific"] = length(J2)
    Nhits[[s]]["down","RNAspecific"] = length(K2)

    J3 = which(RT$padj <= 0.1 & RT$log2FoldChange > 0)
    J3 = setdiff(J3, J)
    fgUp = bg[J3]
    K3 = which(RT$padj <= 0.1 & RT$log2FoldChange < 0)
    K3 = setdiff(K3, K)
    fgDown = bg[K3]
    GSEA$up$RiboTagSpecific[[s]] = doGSEA(fgUp,bg,ENSG2category, k=3)
    GSEA$down$RiboTagSpecific[[s]] = doGSEA(fgDown,bg,ENSG2category, k=3)
    Nhits[[s]]["up","RiboTagSpecific"] = length(J3)
    Nhits[[s]]["down","RiboTagSpecific"] = length(K3)
}
```

3.7 Sequence motifs

Create an output directory.

```
d = file.path("result","Motifs","")
dir.create(d, recursive = TRUE, showWarnings = FALSE)
mainmenu = file.path("..","..","mainmenu.html")
```

```
S = names(SamplesTranslationEfficiency)
TransEff = list()
for (s in S) {
  load(file.path("result","TranslationEfficiency",s,"res.rda"))
  TransEff[[s]] = res
}
```

```
alphabet = function() {
  list(A = c("A", "R", "N", "D", "C", "E", "Q", "G", "H", "I",
             "L", "K", "M", "F", "P", "S", "T", "W", "Y", "V"),
```

```
       N = c("T","C","G","A"))
}

getKMers <- function(seq, K, toMatrix, n) {
  SP = strsplit(as.character(seq), split="")
  tmp = sapply(SP, function(x) {
    res = rep("",length(x)-K+1)
    for (i in seq_len(K)) {
      res = paste(res,x[seq_len(length(x)-K+1)+i-1],sep="")
    }
    res
  })
  if (toMatrix) {
    a = alphabet()[[n]]
    A = a
    if (K > 1) {
      for (k in 2:K) {
        A = paste0(rep(a, each=length(A)),rep(A, times=length(a)))
      }
    }
    tmp = t(sapply(tmp, function(x) { table(factor(x, levels=A)) }))
  }
  tmp
}
```

```
load("data/Seq.rda")

Seq = Seq[1:3]
Seq = lapply(Seq, function(x) {
  I = which(x$gene_biotype == "protein_coding" & x$transcript_biotype == "protein_coding")
  x = x[I,]
  x
})

Seq = lapply(Seq, function(x) {
  n = nchar(x[[1]])
  I = order(n, decreasing = TRUE)
  I = I[!duplicated(x$ensembl_gene_id[I])]
  x = x[I,]
  x
  })
```

The samples are annotated.

```
K = 6
tests = list()
for (s in names(Seq)) {
  tests[[s]] = list()
  for (ct in S) {
#    load(file.path("result","controledTranslation",ct,"res.rda"))
    Seq2 = lapply(Seq, function(x) {
      x = x[which(x[,"ensembl_gene_id"] %in% row.names(TransEff[[ct]])),]
      x
    })
    Seq2 = Seq[[s]][which(Seq[[s]][,"ensembl_gene_id"] %in%
                            row.names(TransEff[[ct]])),]
    seq = Seq2[,1]
    seq[seq == "Sequence unavailable"] = ""
    I = which(nchar(seq) < K)
```

```
    seq[I] = ""
    I = which(nchar(seq) >= K)
    X = getKMers(seq[I],K = K,toMatrix = TRUE, n = "N")

    GN = factor(Seq2[I,"ensembl_gene_id"], levels=row.names(TransEff[[ct]]))
    KM = matrix(nrow=nrow(TransEff[[ct]]), ncol=ncol(X))
    row.names(KM) = levels(GN)
    colnames(KM) = colnames(X)
    for (i in seq_len(ncol(KM))) {
      KM[,i] = tapply(X[,i], GN, sum, na.rm=TRUE)
    }
    KM[KM>1] = 1
    for (upDown in c("up","down")) {
      c = paste0(ct,"_",upDown)
      tests[[s]][[c]] = data.frame(pval = rep(NA, ncol(X)),
                                   padj = NA, est = NA, p=NA)
      row.names(tests[[s]][[c]]) = colnames(X)

      if (upDown == "up") {
        Sel = row.names(TransEff[[ct]])[which(TransEff[[ct]]$padj <= 0.1 & TransEff[[ct]]$log2FoldChang
                                   !is.na(KM[,1]))]
      } else {
        Sel = row.names(TransEff[[ct]])[which(TransEff[[ct]]$padj <= 0.1 & TransEff[[ct]]$log2FoldChang
                                   !is.na(KM[,1]))]
      }
      BG = row.names(TransEff[[ct]])[which(is.finite(TransEff[[ct]]$padj) & !is.na(KM[,1]))]
      p = median(apply(KM[Sel,],1,sum)) / median(apply(KM[setdiff(BG,Sel),],1,sum))
      M = apply(KM[setdiff(BG,Sel),],2,sum)
      m = apply(KM[Sel,],2,sum)
      n = length(Sel)
      N = length(BG)
      Tab = matrix(0,nrow=2,ncol=2)
      for (i in seq_len(ncol(KM))) {
        Tab[2,2] = m[i]
        Tab[1,2] = M[i]
        Tab[2,1] = n-m[i]
        Tab[1,1] = N-M[i]-n
        test = fisher.test(Tab, or = p)
#         test = binom.test(x[i], n, p=p[i])
        tests[[s]][[c]]$pval[i] = test$p.value
        tests[[s]][[c]]$est[i] = test$estimate
      }
      tests[[s]][[c]]$padj[] = p.adjust(tests[[s]][[c]]$pval[],method = "BH")
      tests[[s]][[c]]$p = p
      cat("s=",s," ct=",ct," d=",upDown," FDR20%=",sum(tests[[s]][[c]]$padj < 0.2)," FDR5%=",sum(tests[
    }
  }
}
save(tests, file=file.path(d,"tests.rda"))
```

```
for (s in names(Seq)) {
  print(sapply(tests[[s]], function(x) { sum(x$padj <= 0.1) }))
}
```

```
##      NSC_up    NSC_down      ENB_up    ENB_down      LNB_up    LNB_down   NEURON_up
##           0           1           0           1           0           3           0
## NEURON_down
##           4
```

```
##      NSC_up    NSC_down      ENB_up    ENB_down      LNB_up    LNB_down   NEURON_up
##           0           0           0           9           0           2           0
## NEURON_down
##           6
##      NSC_up    NSC_down      ENB_up    ENB_down      LNB_up    LNB_down   NEURON_up
##           0           0           1          34           0           0           0
## NEURON_down
##        1064
```

```
M = matrix("",ncol = length(Seq), nrow=length(tests[[1]]))
row.names(M) = names(tests[[1]])
colnames(M) = names(Seq)
or = data.frame(pval=c(),padj=c(),est=c(),mRNA=c(),celltype=c())
for (s in names(Seq)) {
  for (ct in names(tests[[1]])) {
    M[ct,s] = sprintf("volcano_%s_%s.png",s,ct)
    png(file.path(d, M[ct,s]))
    x = log2(tests[[s]][[ct]]$est/tests[[s]][[ct]]$p)
    y = -log2(tests[[s]][[ct]]$padj)
    x[x > 2.5] = 2.5
    x[x < -2.5] = -2.5
    y[y > 10] = 10
    y[y < -10] = -10
    plot(x,y,col=ifelse(tests[[s]][[ct]]$padj < 0.1,"red","gray"),xlim=c(-2.5,2.5), ylim=c(0,10),
              main = sprintf("%s %s",s, ct))
    dev.off()
    I = which(tests[[s]][[ct]]$padj < 0.1)
    if (length(I) > 0) {
      x = tests[[s]][[ct]][I,]
      x$est = x$est / x$p
      x$p = NULL
      x$motif = row.names(x)
      x$mRNA = s
      x$celltype = ct
      or = rbind(or, x)
    }
  }
}
colnames(or)[3] = "oddsRatio"
row.names(or) = NULL
file.copy(system.file("images","hwriter.css", package="hwriter"),
          file.path(d, "hwriter.css"))
```

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

```
page = openPage(file.path(d, "index_motifs.html"), link.css="hwriter.css")
hwrite("Overrepresented motifs",heading=1,page=page)
write.table(or, file=file.path(d, "motifs.txt"), sep="\t",quote=FALSE,row.names=FALSE)
hwrite("[Download txt-file]", link="motifs.txt", page=page,br=TRUE)
hwrite(or, page=page)
closePage(page, splash = FALSE)
page = openPage(file.path(d, "index_volcano.html"), link.css="hwriter.css")
hwrite("Volcano plots",heading=1,page=page)
hwrite("[Download txt-file motifs]", link="motifs.txt", page=page,br=TRUE)
hwrite("[Table of motifs]", link="index_motifs.html", page=page,br=TRUE)
hwriteImage(M, page=page)
closePage(page, splash = FALSE)
```

```
p = par(xpd = NA)
```

```
TOPmRNAs = readLines(file.path("data/knownTOPmRNAs.txt"))
for (ct in S) {
  load(file.path("result","TranslationEfficiency",ct,"res.rda"))

  sig = res[,"significant"]
  reg = ifelse(res[,"log2FoldChange"] > 0,"up","down")
  reg = paste0(reg," (",sig,")")
  reg[sig == ""] = "notRegulated"
  reg = factor(reg, levels=c("down (**)", "down (*)", "notRegulated", "up (*)", "up (**)"))
  names(reg) = res$symbol
  png(file.path(d, sprintf("barplot_TOP_%s.png",ct)))
  t = table(reg[res$symbol %in% TOPmRNAs])
  bp = barplot(t,main=sprintf("known TOP mRNAs in %s",ct),ylim=c(0,max(t)*1.05))
  text(bp[,1],t+max(t)*0.03,t)
  dev.off()
  png(file.path(d, sprintf("barplot_all_%s.png",ct)))
  t = table(reg)
  bp = barplot(t,main=sprintf("all mRNAs in %s",ct),ylim=c(0,max(t)*1.05))
  text(bp[,1],t+max(t)*0.03,t)
  dev.off()
  r = reg[res$symbol %in% TOPmRNAs]
  r = r[order(r, names(r))]
  write.table(r, file=file.path(d, sprintf("table_TOP_%s.txt",ct)),sep="\t", col.names=FALSE)
}
par(p)

page = openPage(file.path(d, "index_TOP.html"), link.css="hwriter.css")
hwrite("TOP mRNAs",heading=1,page=page)

M = matrix(S, nr=length(S), ncol=1)
L = matrix(sprintf("table_TOP_%s.txt",S), nr=length(S), ncol=1)
hwrite(M, link = L, page=page,br=TRUE)

IMG = cbind(sprintf("barplot_TOP_%s.png",S),sprintf("barplot_all_%s.png",S))
M = hwriteImage(IMG, table=FALSE)
row.names(M) = S
colnames(M) = c("TOP mRNAs","all mRNAs")
hwrite(M, page=page)
closePage(page, splash = FALSE)
```

Programming code available under:

https://github.com/Martin-Villalba-lab/Data/tree/master/Nature_2019