Pervasive Protein Thermal Stability Variation during the Cell Cycle

Data analysis

Frank Stein

26.04.2018

Initialising R session

Defining a working directory

Definition of the working directory. This folder should contain a metadata file and a data folder containing raw output files from isobarquant for each analysis.

```
setwd("~/home/path to the right file location")
```

Defining some analysis functions

```
add_col_to_df<-function(df,col_name,colvar_init=NA,Vector,data_col="Experiment.ID",
                        Vectornames=NULL,create_column=T){
  if(create_column)df[,col_name]<-colvar_init
  df[,col_name]<-as.character(df[,col_name])
  if(length(Vectornames)==0)Vectornames<-Vector
  count=1
  for(i in Vector){
    x<-Vectornames[count][1]
    df[,col_name][grep(i,df[,data_col])]<-x
    count=count+1
  }
  df[,col_name]<-factor(df[,col_name],ordered=T,levels=c(colvar_init,Vector))
  return(df)
}
sampleWithoutSurprises <- function(x) {
  if (length(x) <= 1) {
    return(x)
  } else {
    return(base::sample(x,1))
  }
}
customPlot <- list(
  theme_bw(base_size=8),
  scale_fill_brewer(palette="Set1"),
  scale_colour_brewer(palette="Set1")
)
```

Loading required packages

```
library(plyr)
library(reshape)
library(ggplot2)
library(vsn)
library(limma)
library(smoothmest)
library(Biobase)
library(MSnbase)
library(matrixStats)
library(gplots)
library(fdrtool)
library(purrr)
library(LSD)
library(gridExtra)
```

2D-TPP analysis

Loading and annotating data

```
conditions<-read.csv("metadata_2DTPP_cellcycle.csv")
conditions<-subset(conditions,Experiment!="X")
conditions$RefCol<-NULL;conditions$Compound<-NULL
conditions<-melt(conditions,id.vars = c("Experiment","Temperature","rep","Path"),
                 variable_name = "tmt.label")
names(conditions)[grep("value",names(conditions))]<-"cell.cycle"
conditions$tmt.label<-gsub("^X","",conditions$tmt.label)
conditions<-na.omit(conditions)
head(conditions)
```

```
##   Experiment Temperature  rep
## 1       S075        37.0 rep2
## 2       S076        40.4 rep2
## 3       S085        44.0 rep2
## 4       S086        46.9 rep2
## 5       S087        49.8 rep2
## 6       S088        52.9 rep2
##                                                          Path tmt.label
## 1   S075_rep2_37_C_merged_results_20161111_1325_proteins.txt       126
## 2 S076_rep2_40-4_C_merged_results_20161111_1325_proteins.txt       126
## 3   S085_rep2_44_C_merged_results_20161111_1325_proteins.txt       126
## 4  S086_rep2_46_9C_merged_results_20161111_1733_proteins.txt       126
## 5  S087_rep2_49_8C_merged_results_20161111_1733_proteins.txt       126
## 6  S088_rep2_52_9C_merged_results_20161111_1733_proteins.txt       126
##   cell.cycle
## 1       G1_S
## 2       G1_S
## 3       G1_S
## 4       G1_S
## 5       G1_S
```

```
## 6       G1_S
```

```
files<-unique(conditions$Path)
files<-files[file.exists(file.path("2DTPP_data",files))]
temperatures<-gsub("C_.+","",files)
temperatures<-gsub("_$","",temperatures)
temperatures<-gsub("_",".",temperatures)
temperatures<-gsub("-",".",temperatures)
temperatures<-gsub("^.+rep[0-9][.]","",temperatures)
data<-NULL
for(i in 1:length(files)){
  file=files[i]
  x<-read.delim(file.path("./2DTPP_data/",file))
  x<-subset(x,!grepl("[K,k][R,r][T,t][0-9]",gene_name))
  x<-subset(x,!grepl("#+",protein_id))
  x$Temperature<-temperatures[i]
  x$Path<-file
  x<-x[,c("protein_id","description","gene_name","top3","Temperature","Path","qupm",
          grep("signal_sum",names(x),value = T))]
  data<-rbind(data,x)
  rm(x)
}
data$description<-gsub(" OS.+","",data$description)
data$description<-gsub(", isoform.+","",data$description)
data$description<-gsub(" [(]Fragment.+","",data$description)
rm(files,i,temperatures,file)
names(data)
```

```
##  [1] "protein_id"      "description"     "gene_name"
##  [4] "top3"            "Temperature"     "Path"
##  [7] "qupm"            "signal_sum_126"  "signal_sum_127L"
## [10] "signal_sum_127H" "signal_sum_128L" "signal_sum_128H"
## [13] "signal_sum_129L" "signal_sum_129H" "signal_sum_130L"
## [16] "signal_sum_130H" "signal_sum_131L"
```

Building an expression set

For the analysis of proteomics data, many popular methods from the microarray-field are borrowed. In order to directly apply the methods, the proteomics data set is stored as an "Expression Set" data type.

Restructuring data

```
data2<-data
data2$description<-NULL
data2$protein_id<-NULL
data2<-unique(data2)
mdata<-melt(data2,id.vars = c("gene_name","top3","Path","Temperature","qupm"),
            variable_name = "tmt.label")
rm(data2)
mdata$measurement<-gsub("_[0-9]+.+","",mdata$tmt.label)
mdata$tmt.label<-gsub("([a-z,A-Z]+_)+","",mdata$tmt.label)
mdata$tmt.label<-as.character(mdata$tmt.label)
mdata<-merge(mdata,conditions)
```

```
mdata$Experiment<-NULL;mdata$Path<-NULL
mdata<-subset(mdata,!is.na(gene_name))
mdata$m.top3<-"m.top3"
mdata$found<-"found.in.reps"
cdata<-cast(mdata,formula=gene_name~measurement+cell.cycle+Temperature+rep+tmt.label,
            value = "value",fun.aggregate = mean,na.rm=T)
cdata2<-cast(subset(mdata,measurement=="signal_sum"),formula=gene_name~m.top3,
             value = "top3",fun.aggregate = mean,na.rm=T)
cdata<-merge(cdata,cdata2)
cdata2<-cast(subset(mdata,measurement=="signal_sum"),formula=gene_name~found,
             value = "top3",fun.aggregate = length)
cdata2$found.in.reps<-cdata2$found.in.reps/7
cdata<-merge(cdata,cdata2)
#found.in.reps for each Temperature
cdata2<-cast(subset(mdata,measurement=="signal_sum"),formula=gene_name~found+Temperature,
             value = "top3",fun.aggregate = length)
cols<-names(cdata2)[names(cdata2)!="gene_name"]
for(i in cols){
  cdata2[,i]<-cdata2[,i]/7
}
rm(i)
cdata<-merge(cdata,cdata2)

mdata$max.qupm<-"max.qupm"
cdata2<-cast(subset(mdata,measurement=="signal_sum"),formula=gene_name~max.qupm,
             value = "qupm",fun.aggregate = max,na.rm=T)
cdata<-merge(cdata,cdata2)
#max.qupm for each Temperature
cdata2<-cast(subset(mdata,measurement=="signal_sum"),formula=gene_name~max.qupm+Temperature,
             value = "qupm",fun.aggregate = max,na.rm=T)
cdata<-merge(cdata,cdata2)
#min.qupm for each Temperature
mdata$max.qupm<-"min.qupm"
cdata2<-cast(subset(mdata,measurement=="signal_sum"),formula=gene_name~max.qupm+Temperature,
             value = "qupm",fun.aggregate = min,na.rm=T)
cdata<-merge(cdata,cdata2)

cdata2<-cast(subset(mdata,rep=="rep2"&measurement=="signal_sum"),formula=gene_name~found,
             value = "top3",fun.aggregate = length)
cdata2$found.in.reps<-cdata2$found.in.reps/7
names(cdata2)[2]<-"found.in.rep2"
cdata<-merge(cdata,cdata2,all=T)

cdata2<-cast(subset(mdata,rep=="rep3"&measurement=="signal_sum"),formula=gene_name~found,
             value = "top3",fun.aggregate = length)
cdata2$found.in.reps<-cdata2$found.in.reps/7
names(cdata2)[2]<-"found.in.rep3"
cdata<-merge(cdata,cdata2,all=T)

cdata2<-cast(subset(mdata,rep=="rep4"&measurement=="signal_sum"),formula=gene_name~found,
             value = "top3",fun.aggregate = length)
cdata2$found.in.reps<-cdata2$found.in.reps/7
names(cdata2)[2]<-"found.in.rep4"
```

```
cdata<-merge(cdata,cdata2,all=T)
rm(cdata2)
```

Filtering data

```
dim(cdata)
```

```
## [1] 10064   247
```

```
cdata<-subset(cdata,max.qupm>=2&found.in.reps>=3&found.in.rep2>=4&found.in.rep3>=4&
                found.in.rep4>=4)
dim(cdata)
```

```
## [1] 5392  247
```

Constructing assay data

```
raw_data<-cdata
rownames(raw_data)<-raw_data$gene_name
raw_data$gene_name<-NULL
raw_data<-raw_data[,grep("signal_sum",names(raw_data),value=T)]
names(raw_data)<-gsub("signal_sum_","",names(raw_data))
raw_data<-as.data.frame(raw_data)
```

Constructing metadata

```
metadata<-data.frame(col.name=names(raw_data))
metadata<-add_col_to_df(metadata,data_col = "col.name",col_name = "tmt.label",
                        Vector = c("126","127L","127H","128L","128H","129L","129H",
                                   "130L","130H","131L","131H"))
metadata<-add_col_to_df(metadata,data_col = "col.name",col_name = "Temperature",
                        Vector = c("37", "40.4", "44", "46.9", "49.8", "52.9", "55.5",
                                   "58.6", "62","66.3"))
metadata<-add_col_to_df(metadata,data_col = "col.name",col_name = "rep",
                        Vector = c("rep1","rep2","rep3","rep4"))
metadata<-merge(metadata,conditions,sort=F)
metadata$ID<-metadata$col.name
metadata$condition<-with(metadata,paste0(cell.cycle,"_",Temperature))
head(metadata)
```

```
##   tmt.label Temperature  rep           col.name Experiment
## 1       126          37 rep2   G1_S_37_rep2_126       S075
## 2       126          37 rep3   G1_S_37_rep3_126       S077
## 3       126          37 rep4   G1_S_37_rep4_126       S146
## 4       126        40.4 rep2 G1_S_40.4_rep2_126       S076
## 5       126        40.4 rep3 G1_S_40.4_rep3_126       S078
## 6       126        40.4 rep4 G1_S_40.4_rep4_126       S147
##                                                        Path cell.cycle
## 1   S075_rep2_37_C_merged_results_20161111_1325_proteins.txt       G1_S
## 2    S077_rep3_37C_merged_results_20161114_1324_proteins.txt       G1_S
## 3    S146_rep4_37C_merged_results_20161107_1553_proteins.txt       G1_S
```

```
## 4 S076_rep2_40-4_C_merged_results_20161111_1325_proteins.txt       G1_S
## 5  S078_rep3_40_4C_merged_results_20161114_1335_proteins.txt       G1_S
## 6  S147_rep4_40-4C_merged_results_20161107_1553_proteins.txt       G1_S
##                    ID condition
## 1   G1_S_37_rep2_126   G1_S_37
## 2   G1_S_37_rep3_126   G1_S_37
## 3   G1_S_37_rep4_126   G1_S_37
## 4 G1_S_40.4_rep2_126 G1_S_40.4
## 5 G1_S_40.4_rep3_126 G1_S_40.4
## 6 G1_S_40.4_rep4_126 G1_S_40.4
```

Constructing feature data

```
rownames(metadata)<-metadata$ID
colnames(raw_data)<-metadata$ID
```

Transformation of raw signal_sum data

The log2 is computed from the raw signal intensities. Furthermore, Infinite values are transformed into missing (NA) values.

```
raw_data_m<-log2(as.matrix(raw_data))
raw_data_m[is.infinite((raw_data_m))]<-NA
```

Constructing the Expression set

```
raw_dataE <- ExpressionSet(assayData = raw_data_m,
                           phenoData = AnnotatedDataFrame(metadata))
validObject(raw_dataE)
```

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

```
rm(raw_data_m,metadata,raw_data,mdata)
```

Normalization

The vsn package from Wolfgang Huber is used to apply a variance stabilization normalization method on the log2 raw data. Since it is a TPP experiment, each temperature is normalized separately. Therefore, the expression set has to be reconstructed again.

```
exprs(raw_dataE)[is.na(exprs(raw_dataE))]<-NA
Ts<-unique(raw_dataE$Temperature)
norm_dataM<-NULL
for(temp in Ts){
  im<-exprs(raw_dataE[,raw_dataE$Temperature==temp])
  vsn.fit<-vsn2(2^im)
  norm_im<-predict(vsn.fit,2^im)
  norm_dataM<-cbind(norm_dataM,norm_im)
  rm(im,norm_im)
}
rm(Ts,temp)
```

Reconstruct Expression set

```
metadata<-data.frame(col.name=colnames(norm_dataM))
metadata<-add_col_to_df(metadata,data_col = "col.name",col_name = "tmt.label",
                        Vector = c("126","127L","127H","128L","128H","129L","129H",
                                   "130L","130H","131L","131H"))
metadata<-add_col_to_df(metadata,data_col = "col.name",col_name = "Temperature",
                        Vector = c("37", "40.4", "44", "46.9", "49.8", "52.9", "55.5",
                                   "58.6", "62","66.3"))
metadata<-add_col_to_df(metadata,data_col = "col.name",col_name = "rep",
                        Vector = c("rep1","rep2","rep3","rep4"))
metadata<-merge(metadata,conditions,sort=F)
metadata$ID<-metadata$col.name
metadata$condition<-with(metadata,paste0(cell.cycle,"_",Temperature))
head(metadata)
```

```
##   tmt.label Temperature  rep            col.name Experiment
## 1       126          37 rep2    G1_S_37_rep2_126       S075
## 2       126          37 rep3    G1_S_37_rep3_126       S077
## 3       126          37 rep4    G1_S_37_rep4_126       S146
## 4      127L          37 rep2 earlyS_37_rep2_127L       S075
## 5      127L          37 rep3 earlyS_37_rep3_127L       S077
## 6      127L          37 rep4 earlyS_37_rep4_127L       S146
##                                                    Path cell.cycle
## 1 S075_rep2_37_C_merged_results_20161111_1325_proteins.txt       G1_S
## 2  S077_rep3_37C_merged_results_20161114_1324_proteins.txt       G1_S
## 3  S146_rep4_37C_merged_results_20161107_1553_proteins.txt       G1_S
## 4 S075_rep2_37_C_merged_results_20161111_1325_proteins.txt     earlyS
## 5  S077_rep3_37C_merged_results_20161114_1324_proteins.txt     earlyS
## 6  S146_rep4_37C_merged_results_20161107_1553_proteins.txt     earlyS
##                     ID condition
## 1    G1_S_37_rep2_126   G1_S_37
## 2    G1_S_37_rep3_126   G1_S_37
## 3    G1_S_37_rep4_126   G1_S_37
## 4 earlyS_37_rep2_127L earlyS_37
## 5 earlyS_37_rep3_127L earlyS_37
## 6 earlyS_37_rep4_127L earlyS_37
```

```
rownames(metadata)<-metadata$ID
colnames(norm_dataM)<-metadata$ID
norm_dataE <- ExpressionSet(assayData = norm_dataM,
                            phenoData = AnnotatedDataFrame(metadata))
rm(norm_dataM,metadata,vsn.fit)
```

Remove Batcheffects

Batcheffects are removed by fitting a linear model to the data that try to explain the replicates. The batch effect is than subtracted from the data applying the limma package.

```
batchcl_norm_dataE<-norm_dataE
exprs(batchcl_norm_dataE)<-removeBatchEffect(exprs(batchcl_norm_dataE),
                            batch=as.character(pData(batchcl_norm_dataE)$rep),
                            design=model.matrix(~as.character(pData(batchcl_norm_dataE)$condition)))
```

Merge normalized data back into 'data'

```
cd<-as.data.frame(2^exprs(batchcl_norm_dataE))
names(cd)<-paste0("norm_signal_sum_",names(cd))
cd$gene_name<-rownames(cd)
cdata<-merge(cdata,cd)
rm(cd)
rm(raw_dataE,batchcl_norm_dataE,norm_dataE)
```

Calculation of fold changes

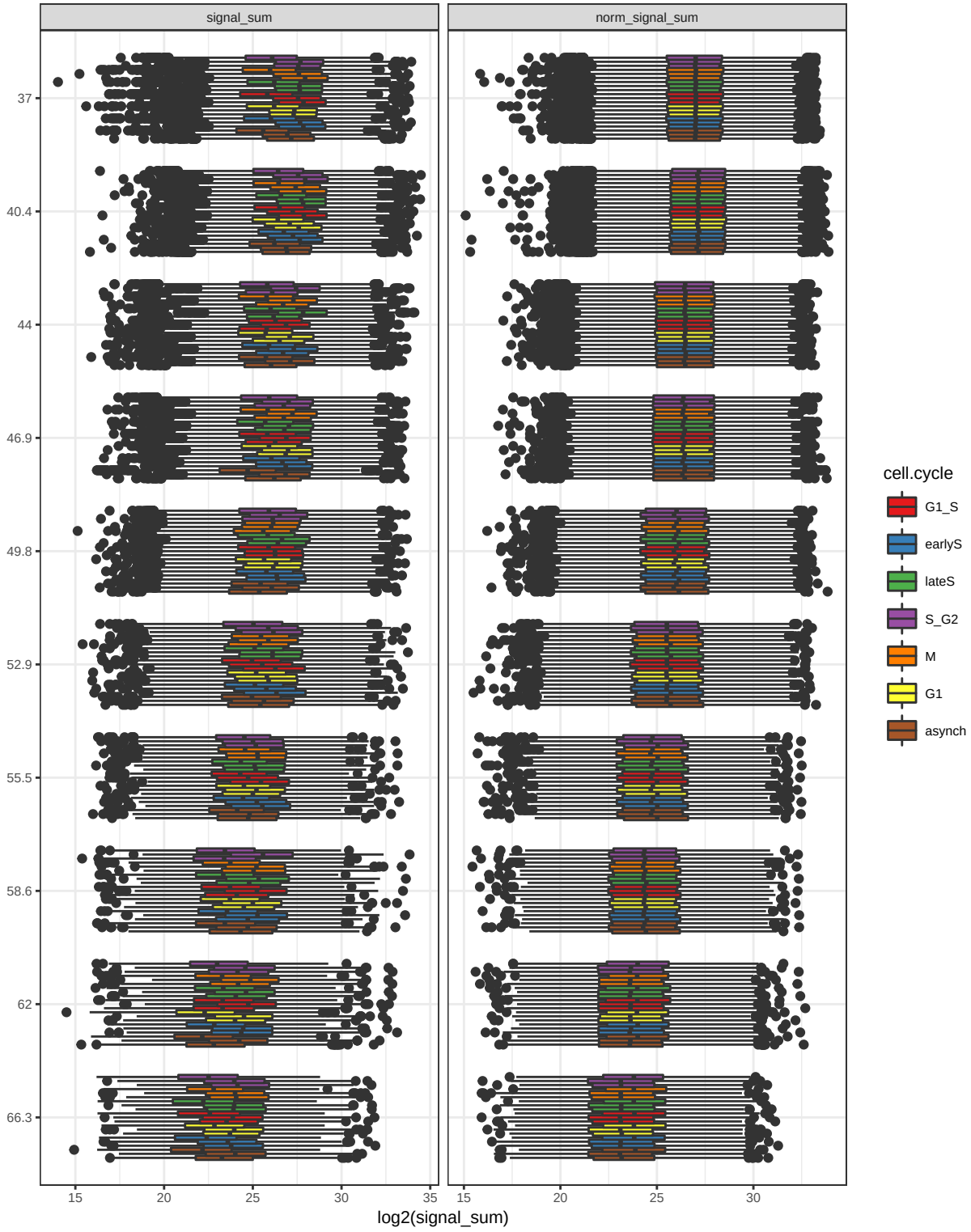
```
fc<-cdata[,c("gene_name",grep("norm_signal_sum",names(cdata),value=T),
             grep("max.qupm_",names(cdata),value=T))]
names(fc)[grepl("norm_signal_sum",names(fc))]<-gsub("_[0-9]+.$","",
           names(fc)[grepl("norm_signal_sum",names(fc))])
names(fc)<-gsub("norm_signal_sum_","",names(fc))
cols<-ncol(fc)

temps<-as.character(unique(conditions$Temperature))
cycles<-as.character(na.omit(unique(conditions$cell.cycle)))
reps<-as.character(unique(conditions$rep))
for(temp in temps){
  for(cycle in cycles){
    for(rep in reps){
      name<-paste0(cycle,".G1S_",temp,"_",rep,".cycr")
      fc[,name]<-fc[,paste0(cycle,"_",temp,"_",rep)]/fc[,paste0("G1_S_",temp,"_",rep)]
      fc[,name][fc[,paste0("max.qupm_",temp)]<2]<-NA
    }
  }
}

for(cycle in cycles){
  for(temp in temps){
    for(rep in reps){
      name<-paste0(cycle,"_",temp,"_",rep,".37.tempr")
      fc[,name]<-fc[,paste0(cycle,"_",temp,"_",rep)]/fc[,paste0(cycle,"_37","_",rep)]
      fc[,name][fc[,paste0("max.qupm_",temp)]<2]<-NA
    }
  }
}

fc<-fc[,-c(2:cols)]
rm(temps,cycles,temp,cycle,rep,reps,name)
cdata<-merge(cdata,fc)
rm(fc,cols)
write.csv(cdata,file.path("processed_data","2D_TPP_cdata_V1.csv"),row.names = F)
```

Reshaping full data set

```
mfc<-melt(cdata[,c("gene_name","m.top3","max.qupm",grep("^signal_sum",names(cdata),value=T),
                   grep("norm_signal_sum",names(cdata),value=T),
                   grep("cycr",names(cdata),value=T),
                   grep("tempr",names(cdata),value=T))],
          id.vars = c("gene_name","m.top3","max.qupm"))
mfc<-na.omit(mfc)
mfc<-add_col_to_df(mfc,col_name = "measurement",data_col = "variable",
                   Vector=c("signal_sum","norm_signal_sum","cycr","tempr"))
mfc$variable<-gsub(".+ignal_sum_","",mfc$variable)
mfc$variable<-gsub("_[0-9]+$","",mfc$variable)
mfc$variable<-gsub("_[0-9]+[H,L]$","",mfc$variable)
mfc$variable<-gsub(".cycr$","",mfc$variable)
mfc$variable<-gsub(".G1S","",mfc$variable)
mfc$variable<-gsub(".37.tempr$","",mfc$variable)
mfc$rep<-mfc$variable
mfc$variable<-gsub("_rep[0-9]+$","",mfc$variable)
names(mfc)[grep("variable",names(mfc))]<-c("condition")
mfc$experiment<-mfc$condition
mfc$rep<-gsub(".+_","",mfc$rep)
mfc$measurement<-revalue(mfc$measurement,
                         replace = c("cycr"="G1S.ratio","tempr"="T37C.ratio"))
mdata<-cast(mfc,formula=gene_name+m.top3+max.qupm+condition+experiment+rep~measurement,
            variable="value")

conditions$condition<-with(conditions,paste0(cell.cycle,"_",Temperature))
mdata<-merge(mdata,conditions)
mdata$temp<-as.numeric(as.character(mdata$Temperature))
mdata$Temperature<-factor(mdata$Temperature)
write.csv(mdata,file.path("processed_data","2D_TPP_mdata_V1.csv"),row.names = F)
mfc<-merge(mfc,conditions)
mfc$Temperature<-factor(mfc$Temperature)
mfc$cell.cycle<-factor(mfc$cell.cycle,ordered=T,levels=c("G1_S","earlyS","lateS",
                                                        "S_G2","M","G1","asynch"))
rm(conditions)
```

Quality control of normalization

```
ggplot(data=subset(mfc,measurement%in%c("signal_sum","norm_signal_sum")),
       aes(Temperature,log2(value),fill=cell.cycle))+facet_grid(.~measurement,scale="free")+
  geom_boxplot(aes(group=paste(Temperature,cell.cycle,rep)))+theme_bw(base_size=8)+
  scale_fill_brewer(palette="Set1")+ylab("log2(signal_sum)")+coord_flip()+
  scale_x_discrete(name="",limits=rev(levels(mfc$Temperature)))
```



Create Excel-Sheet for data browsing

```
mfc3<-mdata
mfc3<-subset(mfc3,signal_sum!=0)
#median G1S.ratio
odata<-cast(mfc3,formula=gene_name+Temperature~cell.cycle,value="G1S.ratio",
            fun.aggregate=median,na.rm=T)
names(odata)[!names(odata)%in%c("gene_name","Temperature")]<-
  paste0(names(odata)[!names(odata)%in%c("gene_name","Temperature")],".median")
#range G1S.ratio
odata2<-cast(mfc3,formula=gene_name+Temperature~cell.cycle,
             value="G1S.ratio",fun.aggregate=function(x){return(diff(range(log2(x),
                                                                          na.rm = T)))})
names(odata2)[!names(odata2)%in%c("gene_name","Temperature")]<-
  paste0(names(odata2)[!names(odata2)%in%c("gene_name","Temperature")],".log2range")
odata<-merge(odata,odata2)

#found in replicates
odata2<-cast(mfc3,formula=gene_name+Temperature~cell.cycle,value="G1S.ratio",
             fun.aggregate=length)
odata2$found.in.reps<-odata2$G1_S
odata2<-odata2[,c("gene_name","Temperature","found.in.reps")]
odata<-merge(odata,odata2)
rm(odata2)
#
modata<-melt(cdata[,c("gene_name","m.top3",grep(".qupm_",names(cdata),value=T))],
             id.vars = c("gene_name","m.top3"))
modata$Temperature<-gsub(".+_","",modata$variable)
modata$variable<-gsub("_.+","",modata$variable)
modata<-cast(modata,formula=gene_name+m.top3+Temperature~variable)
odata<-merge(odata,modata)
rm(modata)

odata$path<-file.path(".","plots",paste0(gsub("\\|","-",odata$gene_name),".pdf"))
des<-data[,c("gene_name","description")]
des<-subset(des,gene_name%in%unique(odata$gene_name))
des<-unique(des)
dupnames<-names(which(sort(table(des$gene_name),decreasing = T)>1))
des.dups<-subset(des,gene_name%in%dupnames)
des<-subset(des,!gene_name%in%dupnames)
des.dups.new<-NULL
for(gene in as.character(unique(des.dups$gene_name))){
  sub<-subset(des.dups,gene_name==gene)
  sub<-sub[which(nchar(sub$description)==max(nchar(sub$description)))[1],]
  sub<-data.frame(gene_name=sub$gene_name,description=sub$description)
  des.dups.new<-rbind(des.dups.new,sub)
  rm(sub)
}
rm(gene,dupnames)
des<-rbind(des,des.dups.new)
rm(des.dups.new,des.dups)
odata<-merge(odata,des)
rm(des)
odata<-odata[,c("gene_name","description","found.in.reps", "m.top3", "max.qupm",
                "min.qupm","Temperature","G1_S.median", "earlyS.median", "lateS.median",
```

```
                "S_G2.median", "M.median", "G1.median","asynch.median", "G1_S.log2range",
                "earlyS.log2range", "lateS.log2range", "S_G2.log2range", "M.log2range",
                "G1.log2range", "asynch.log2range","path")]

write.csv(odata,file.path("processed_data","2D_TPP_final_cell_cycle_results_V2.csv"))
rm(odata)
rm(mfc3)
```

Clean global environment

```
rm(cdata,data,mdata,mfc)
```

SDS data analysis

Loading and annotating data

```
conditions<-read.csv("metadata_SDS_cellcycle.csv")
conditions$RefCol<-NULL
conditions$treatment<-NULL
conditions<-melt(conditions,id.vars = c("rep","Path"),variable_name = "tmt.label")
names(conditions)[grep("value",names(conditions))]<-"cell.cycle"
conditions$tmt.label<-gsub("^X","",conditions$tmt.label)
conditions<-na.omit(conditions)
conditions<-subset(conditions,cell.cycle!="")
conditions
```

```
##     rep                                                Path tmt.label
## 1  rep2      rep2_merged_results_20170505_1510_proteins.txt       126
## 2  rep3 rep3_redo_merged_results_20170508_1557_proteins.txt       126
## 3  rep4 rep4_redo_merged_results_20170427_1643_proteins.txt       126
## 4  rep2      rep2_merged_results_20170505_1510_proteins.txt      127L
## 5  rep3 rep3_redo_merged_results_20170508_1557_proteins.txt      127L
## 6  rep4 rep4_redo_merged_results_20170427_1643_proteins.txt      127L
## 7  rep2      rep2_merged_results_20170505_1510_proteins.txt      127H
## 8  rep3 rep3_redo_merged_results_20170508_1557_proteins.txt      127H
## 9  rep4 rep4_redo_merged_results_20170427_1643_proteins.txt      127H
## 10 rep2      rep2_merged_results_20170505_1510_proteins.txt      128L
## 11 rep3 rep3_redo_merged_results_20170508_1557_proteins.txt      128L
## 12 rep4 rep4_redo_merged_results_20170427_1643_proteins.txt      128L
## 13 rep2      rep2_merged_results_20170505_1510_proteins.txt      128H
## 14 rep3 rep3_redo_merged_results_20170508_1557_proteins.txt      128H
## 15 rep4 rep4_redo_merged_results_20170427_1643_proteins.txt      128H
## 16 rep2      rep2_merged_results_20170505_1510_proteins.txt      129L
## 18 rep4 rep4_redo_merged_results_20170427_1643_proteins.txt      129L
## 19 rep2      rep2_merged_results_20170505_1510_proteins.txt      129H
## 20 rep3 rep3_redo_merged_results_20170508_1557_proteins.txt      129H
## 21 rep4 rep4_redo_merged_results_20170427_1643_proteins.txt      129H
##    cell.cycle
## 1        G1_S
## 2        G1_S
```

```
## 3        G1_S
## 4      earlyS
## 5      earlyS
## 6      earlyS
## 7       lateS
## 8       lateS
## 9       lateS
## 10       S_G2
## 11       S_G2
## 12       S_G2
## 13          M
## 14          M
## 15          M
## 16         G1
## 18         G1
## 19     asynch
## 20     asynch
## 21     asynch
```

```
files<-unique(conditions$Path)
files<-files[file.exists(file.path("SDS_data",files))]
data<-NULL
for(i in 1:length(files)){
  file=files[i]
  x<-read.delim(file.path("./SDS_data/",file))
  x<-subset(x,!grepl("[K,k][R,r][T,t][0-9]",gene_name))
  x<-subset(x,!grepl("#+",protein_id))
  x$Temperature<-37
  x$Path<-file
  x<-x[,c("protein_id","description","gene_name","top3","Temperature","Path","qupm",
          grep("signal_sum",names(x),value = T))]
  data<-rbind(data,x)
  rm(x)
}
data$description<-gsub(" OS.+","",data$description)
data$description<-gsub(", isoform.+","",data$description)
data$description<-gsub(" [(]Fragment.+","",data$description)
rm(files,i,file)
names(data)
```

```
##  [1] "protein_id"      "description"     "gene_name"
##  [4] "top3"            "Temperature"     "Path"
##  [7] "qupm"            "signal_sum_126"  "signal_sum_127L"
## [10] "signal_sum_127H" "signal_sum_128L" "signal_sum_128H"
## [13] "signal_sum_129L" "signal_sum_129H" "signal_sum_130L"
## [16] "signal_sum_130H" "signal_sum_131L"
```

Building an expression set

Restructuring data

```
data2<-data
data2$description<-NULL
```

```
data2$protein_id<-NULL
data2<-unique(data2)
mdata<-melt(data2,id.vars = c("gene_name","top3","Path","Temperature","qupm"),
            variable_name = "tmt.label")
rm(data2)
mdata$measurement<-gsub("_[0-9]+.+","",mdata$tmt.label)
mdata$tmt.label<-gsub("([a-z,A-Z]+_)+","",mdata$tmt.label)
mdata$tmt.label<-as.character(mdata$tmt.label)
mdata<-merge(mdata,conditions)
mdata$Path<-NULL
mdata<-subset(mdata,!is.na(gene_name))
mdata$m.top3<-"m.top3"
mdata$found<-"found.in.reps"
mdata$SDS<-"SDS"
cdata<-cast(mdata,
            formula=gene_name~measurement+SDS+cell.cycle+Temperature+rep+tmt.label,
            value = "value",fun.aggregate = mean,na.rm=T)
cdata2<-cast(subset(mdata,measurement=="signal_sum"),formula=gene_name~m.top3,
             value = "top3",fun.aggregate = mean,na.rm=T)
cdata<-merge(cdata,cdata2)
cdata2<-cast(subset(mdata,measurement=="signal_sum"),formula=gene_name~found,
             value = "top3",fun.aggregate = length)
cdata2$found.in.reps<-cdata2$found.in.reps/7
cdata<-merge(cdata,cdata2)

mdata$max.qupm<-"max.qupm"
cdata2<-cast(subset(mdata,measurement=="signal_sum"),formula=gene_name~max.qupm,
             value = "qupm",fun.aggregate = max,na.rm=T)
cdata<-merge(cdata,cdata2)
rm(cdata2)
```

Filtering data

```
cdata<-subset(cdata,max.qupm>=2&found.in.reps>=2)
```

Constructing assay data

```
raw_data<-cdata
rownames(raw_data)<-raw_data$gene_name
raw_data$gene_name<-NULL
raw_data<-raw_data[,grep("signal_sum",names(raw_data),value=T)]
names(raw_data)<-gsub("signal_sum_","",names(raw_data))
raw_data<-as.data.frame(raw_data)
```

Constructing metadata

```
metadata<-data.frame(col.name=names(raw_data))
metadata<-add_col_to_df(metadata,data_col = "col.name",col_name = "tmt.label",
                        Vector = c("126","127L","127H","128L","128H","129L","129H",
```

```
                                    "130L","130H","131L","131H"))
metadata<-add_col_to_df(metadata,data_col = "col.name",col_name = "rep",
                        Vector = c("rep1","rep2","rep3","rep4"))
metadata<-merge(metadata,conditions,sort=F)
metadata$ID<-metadata$col.name
metadata$Temperature<-37
metadata$condition<-with(metadata,paste0(cell.cycle,"_",Temperature))
metadata
```

```
##    tmt.label  rep                col.name
## 1        126 rep2    SDS_G1_S_37_rep2_126
## 2        126 rep3    SDS_G1_S_37_rep3_126
## 3        126 rep4    SDS_G1_S_37_rep4_126
## 4       127L rep2 SDS_earlyS_37_rep2_127L
## 5       127L rep3 SDS_earlyS_37_rep3_127L
## 6       127L rep4 SDS_earlyS_37_rep4_127L
## 7       127H rep2  SDS_lateS_37_rep2_127H
## 8       127H rep3  SDS_lateS_37_rep3_127H
## 9       127H rep4  SDS_lateS_37_rep4_127H
## 10      128L rep2   SDS_S_G2_37_rep2_128L
## 11      128L rep3   SDS_S_G2_37_rep3_128L
## 12      128L rep4   SDS_S_G2_37_rep4_128L
## 13      128H rep2      SDS_M_37_rep2_128H
## 14      128H rep3      SDS_M_37_rep3_128H
## 15      128H rep4      SDS_M_37_rep4_128H
## 16      129L rep2     SDS_G1_37_rep2_129L
## 17      129L rep4     SDS_G1_37_rep4_129L
## 18      129H rep2 SDS_asynch_37_rep2_129H
## 19      129H rep3 SDS_asynch_37_rep3_129H
## 20      129H rep4 SDS_asynch_37_rep4_129H
##                                                  Path cell.cycle
## 1       rep2_merged_results_20170505_1510_proteins.txt       G1_S
## 2  rep3_redo_merged_results_20170508_1557_proteins.txt       G1_S
## 3  rep4_redo_merged_results_20170427_1643_proteins.txt       G1_S
## 4       rep2_merged_results_20170505_1510_proteins.txt     earlyS
## 5  rep3_redo_merged_results_20170508_1557_proteins.txt     earlyS
## 6  rep4_redo_merged_results_20170427_1643_proteins.txt     earlyS
## 7       rep2_merged_results_20170505_1510_proteins.txt      lateS
## 8  rep3_redo_merged_results_20170508_1557_proteins.txt      lateS
## 9  rep4_redo_merged_results_20170427_1643_proteins.txt      lateS
## 10      rep2_merged_results_20170505_1510_proteins.txt       S_G2
## 11 rep3_redo_merged_results_20170508_1557_proteins.txt       S_G2
## 12 rep4_redo_merged_results_20170427_1643_proteins.txt       S_G2
## 13      rep2_merged_results_20170505_1510_proteins.txt          M
## 14 rep3_redo_merged_results_20170508_1557_proteins.txt          M
## 15 rep4_redo_merged_results_20170427_1643_proteins.txt          M
## 16      rep2_merged_results_20170505_1510_proteins.txt         G1
## 17 rep4_redo_merged_results_20170427_1643_proteins.txt         G1
## 18      rep2_merged_results_20170505_1510_proteins.txt     asynch
## 19 rep3_redo_merged_results_20170508_1557_proteins.txt     asynch
## 20 rep4_redo_merged_results_20170427_1643_proteins.txt     asynch
##                          ID Temperature condition
## 1     SDS_G1_S_37_rep2_126          37   G1_S_37
## 2     SDS_G1_S_37_rep3_126          37   G1_S_37
```

```
## 3     SDS_G1_S_37_rep4_126          37   G1_S_37
## 4  SDS_earlyS_37_rep2_127L          37 earlyS_37
## 5  SDS_earlyS_37_rep3_127L          37 earlyS_37
## 6  SDS_earlyS_37_rep4_127L          37 earlyS_37
## 7   SDS_lateS_37_rep2_127H          37  lateS_37
## 8   SDS_lateS_37_rep3_127H          37  lateS_37
## 9   SDS_lateS_37_rep4_127H          37  lateS_37
## 10   SDS_S_G2_37_rep2_128L          37   S_G2_37
## 11   SDS_S_G2_37_rep3_128L          37   S_G2_37
## 12   SDS_S_G2_37_rep4_128L          37   S_G2_37
## 13      SDS_M_37_rep2_128H          37      M_37
## 14      SDS_M_37_rep3_128H          37      M_37
## 15      SDS_M_37_rep4_128H          37      M_37
## 16     SDS_G1_37_rep2_129L          37     G1_37
## 17     SDS_G1_37_rep4_129L          37     G1_37
## 18 SDS_asynch_37_rep2_129H          37 asynch_37
## 19 SDS_asynch_37_rep3_129H          37 asynch_37
## 20 SDS_asynch_37_rep4_129H          37 asynch_37
```

Constructing feature data

```
rownames(metadata)<-metadata$ID
colnames(raw_data)<-metadata$ID
```

Transformation of raw signal_sum data

The log2 is computed from the raw signal intensities. Furthermore, Infinite values are transformed into missing (NA) values.

```
raw_data_m<-log2(as.matrix(raw_data))
raw_data_m[is.infinite((raw_data_m))]<-NA
raw_data_m[is.nan((raw_data_m))]<-NA
```

Constructing the Expression set

```
raw_dataE <- ExpressionSet(assayData = raw_data_m,
                           phenoData = AnnotatedDataFrame(metadata))
validObject(raw_dataE)
```

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

```
rm(raw_data_m,metadata,raw_data,mdata)
```

Remove Batcheffects

Batcheffects are removed by fitting a linear model to the data that try to explain the replicates. The batch effect is than subtracted from the data applying the limma package.

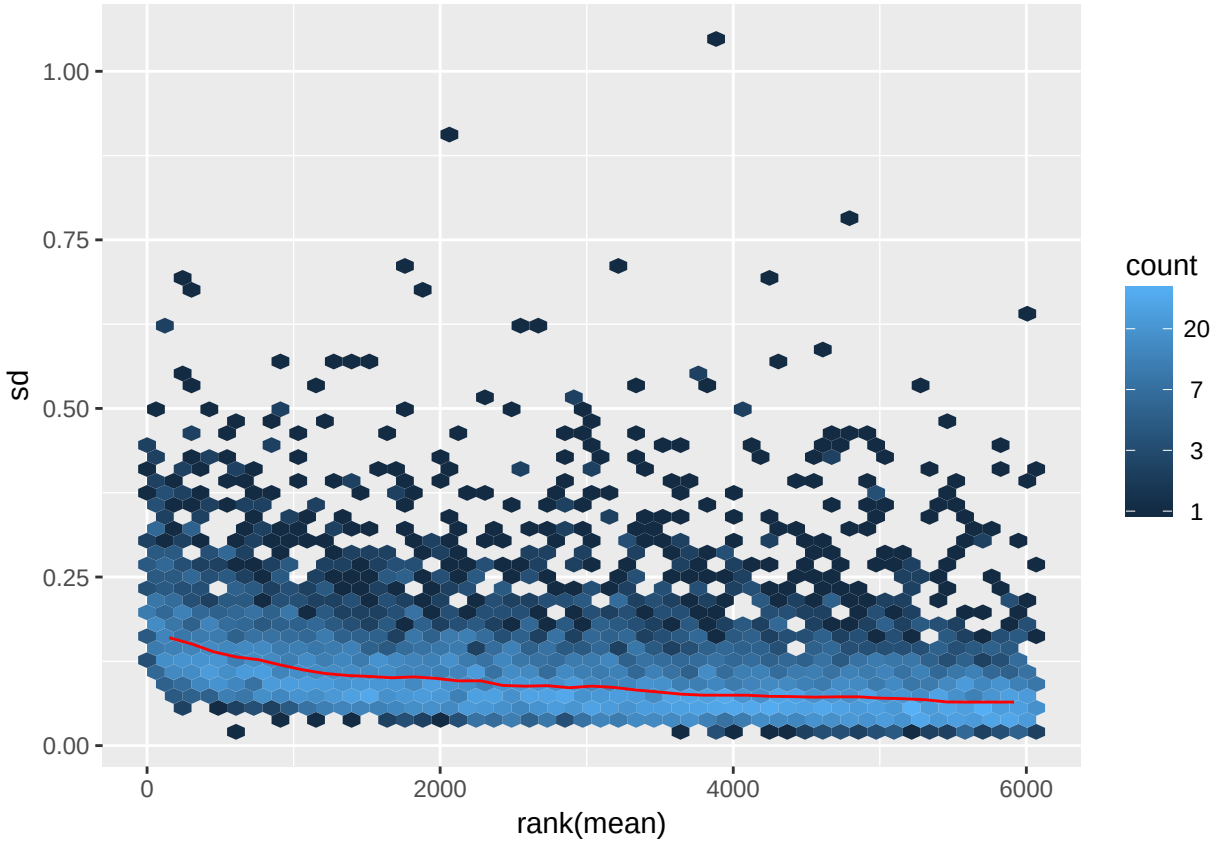
```
batchcl_raw_dataE<-raw_dataE
exprs(batchcl_raw_dataE)<-removeBatchEffect(exprs(batchcl_raw_dataE),
```

```
                        batch=as.character(pData(batchcl_raw_dataE)$rep),
                        design=model.matrix(~as.character(pData(batchcl_raw_dataE)$condition)))
```

Normalization

The vsn package from Wolfgang Huber is used to apply a variance stabilization normalization method on the log2 raw data. Since it is a TPP experiment, each temperature is normalized separately. Therefore, the expression set has to be reconstructed again.

```
vsn.fit<-vsn2(2^exprs(batchcl_raw_dataE))
norm_batchcl_dataE<-batchcl_raw_dataE
meanSdPlot((vsn.fit))
```




```
exprs(norm_batchcl_dataE)<-predict(vsn.fit,2^exprs(norm_batchcl_dataE))
rm(vsn.fit)
```

Merge normalized data back into 'data'

```
cd<-as.data.frame(2^exprs(batchcl_raw_dataE))
names(cd)<-paste0("batchcl_raw_signal_sum_",names(cd))
cd$gene_name<-rownames(cd)
cdata<-merge(cdata,cd)
cd<-as.data.frame(2^exprs(norm_batchcl_dataE))
names(cd)<-paste0("norm_batchcl_signal_sum_",names(cd))
```

```
cd$gene_name<-rownames(cd)
cdata<-merge(cdata,cd)
rm(cd)
```

Calculation of fold changes

```
fc<-cdata[,c("gene_name",grep("norm_batchcl_signal_sum_",names(cdata),value=T))]
names(fc)[grepl("norm_batchcl_signal_sum_",names(fc))]<-gsub("_[0-9]+.$","",
                              names(fc)[grepl("norm_batchcl_signal_sum_",names(fc))])
names(fc)<-gsub("norm_batchcl_signal_sum_","",names(fc))
names(fc)<-gsub("_37","",names(fc))
names(fc)<-gsub("SDS_","",names(fc))
cols<-ncol(fc)

cycles<-as.character(na.omit(unique(conditions$cell.cycle)))
reps<-as.character(unique(conditions$rep))
for(cycle in cycles){
  for(rep in reps){
    name<-paste0(cycle,".G1S_SDS_37_",rep,".cycr")
    try(fc[,name]<-fc[,paste0(cycle,"_",rep)]/fc[,paste0("G1_S_",rep)])
  }
}
rm(name)
fc<-fc[,-c(2:cols)]
rm(cycles,cycle,rep,reps,cols)
cdata<-merge(cdata,fc)
rm(fc)
write.csv(cdata,file = file.path("processed_data","SDS_cdata_V1.csv"))
```

Limma analysis

Construction of design matrix and defining conditions that will be compared

```
condition<-factor(as.character(pData(norm_batchcl_dataE)$condition))
replicate<-factor(as.character(pData(norm_batchcl_dataE)$rep))
condition_d<-model.matrix(~0+condition+replicate)
colnames(condition_d)<-gsub("condition","",colnames(condition_d))
colnames(condition_d)<-gsub("_37","",colnames(condition_d))
to_label<-c("asynch - G1_S","earlyS - G1_S","G1 - G1_S","lateS - G1_S",
            "M - G1_S","S_G2 - G1_S")
```

Fitting the model

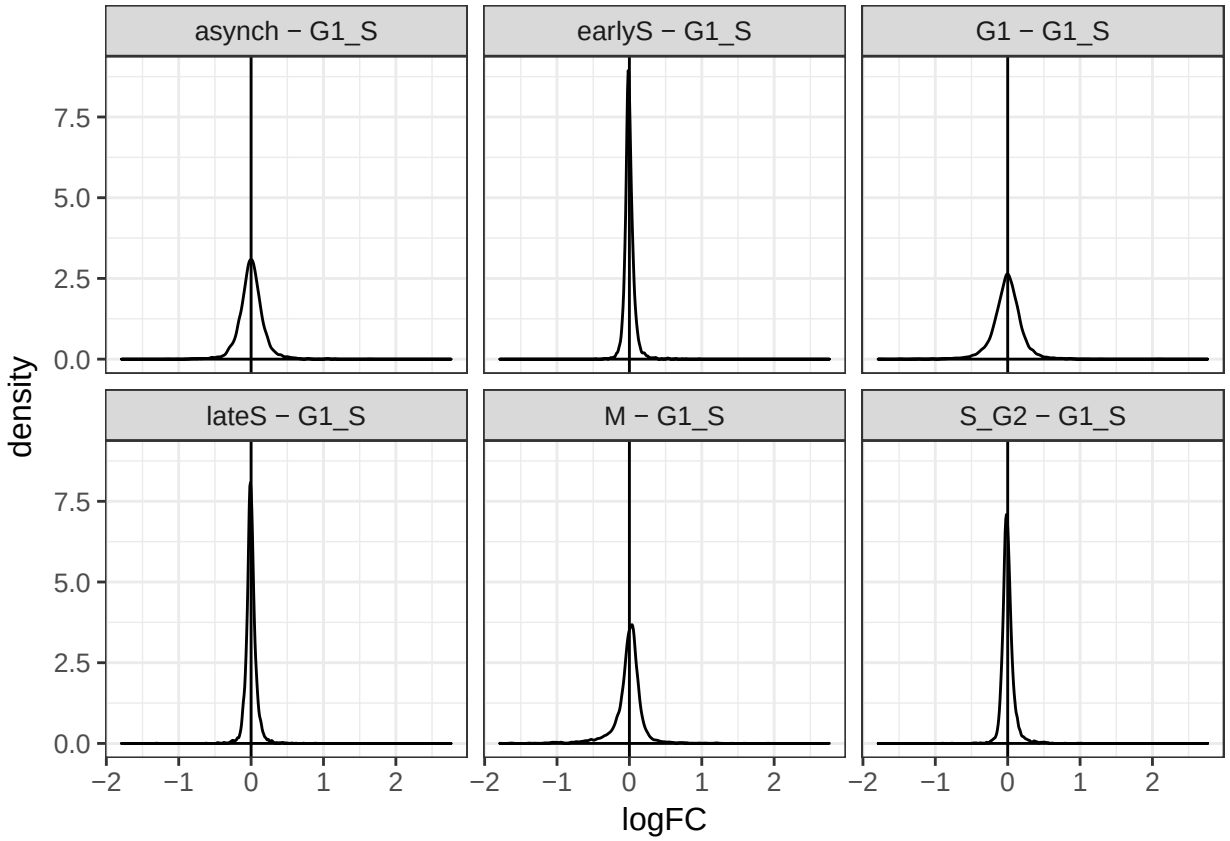
```
limma_comparison<-eBayes(contrasts.fit(lmFit(norm_batchcl_dataE,
                                design=condition_d),
                                makeContrasts(contrasts = to_label,levels=condition_d)))
rm(condition_d,condition,replicate)
```

Identification of top hits

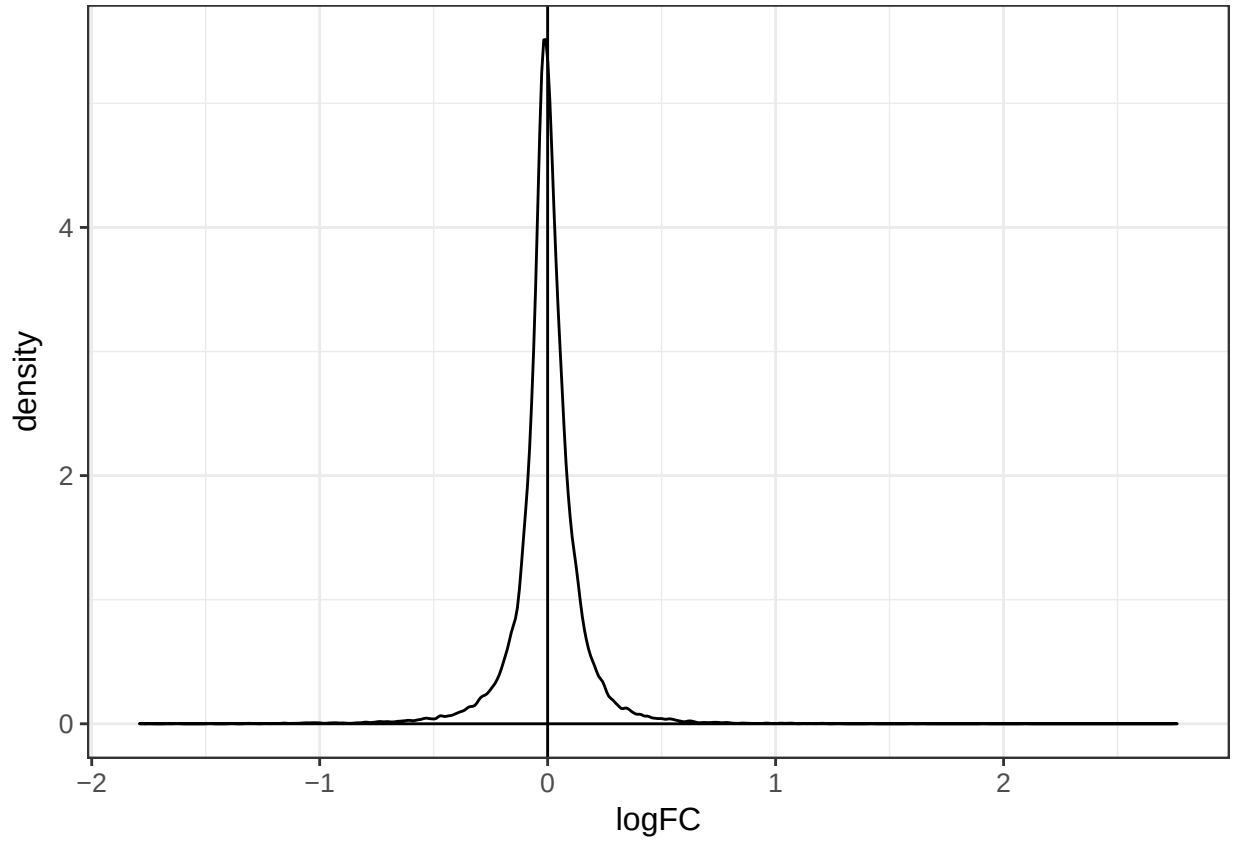
```
limma_res<-NULL
for(comp in to_label){
  res <- topTable(limma_comparison, sort.by = "t",  coef = comp, number = Inf)
  res<-na.omit(res)
  res$gene_name<-rownames(res)
  fdr_res <- fdrtool(res$t, plot =F, verbose = F)
  res$local.p.value<-fdr_res$pval
  res$qval <- fdr_res$qval
  res$lfdr <- fdr_res$lfdr
  rm(fdr_res)
  res$comparison <- rep(comp, dim(res)[1])
  limma_res<-rbind(limma_res,res)
}
limma_res$fdr<-p.adjust(limma_res$local.p.value,method = "fdr")
rm(res,limma_comparison,comp,to_label)
limma_res$hit_old<-with(limma_res,
                        ifelse(abs(logFC)>log2(1.5)&(adj.P.Val<=0.01|qval<=0.01),T,F))
```

FC distribution plot

```
qplot(logFC,data=limma_res,geom="density")+facet_wrap(~comparison)+
  theme_bw(base_size=12)+geom_vline(aes(xintercept=0))
```



```
qplot(logFC,data=limma_res,geom="density")+theme_bw(base_size=12)+
  geom_vline(aes(xintercept=0))
```



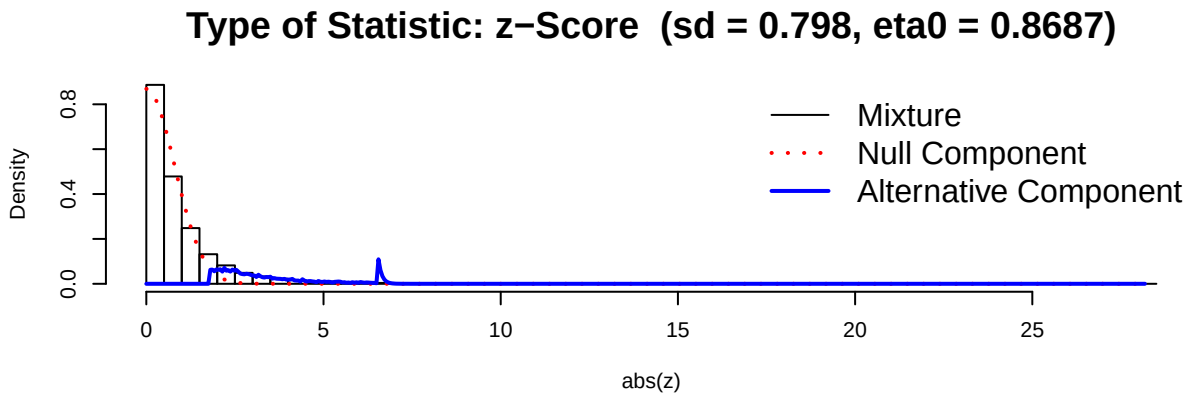
Transform logFC into z-scores

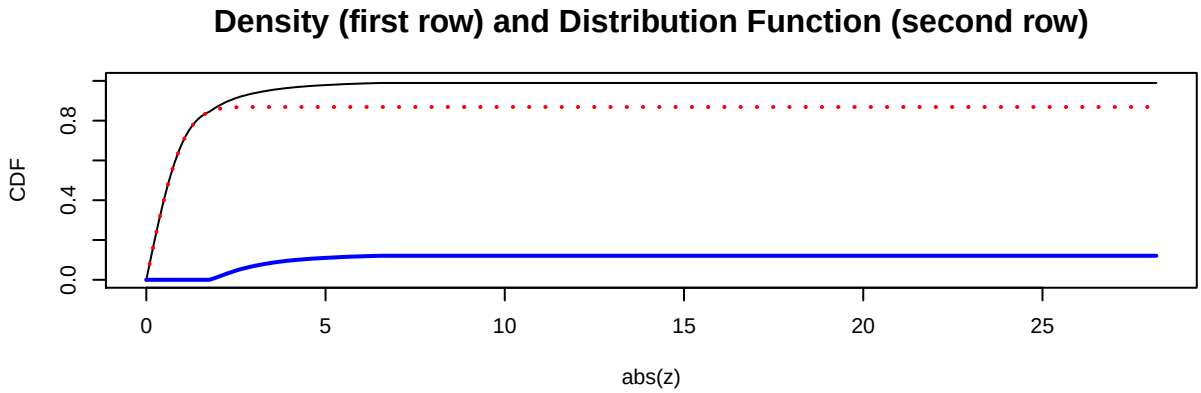
```
col.names<-c("logFC")
for(cn in col.names){
  ratios<-limma_res[,cn]
  ratios<-as.double(as.character(unlist(ratios)))
  print(paste(cn,length(ratios)))
  quants<-quantile(ratios,probs = c(0.1587,0.5,0.8413),na.rm=T)
  ratios[is.na(ratios)]<-0
  pratios<-ratios>=0
  nratios<-ratios<0
  z<-ratios
  z[nratios]<-ratios[nratios]/as.numeric(diff(quants)[1])
  z[pratios]<-ratios[pratios]/as.numeric(diff(quants)[2])
  p.values<-z
  p.values[nratios]<-pt(z[nratios],df = length(ratios))
  p.values[pratios]<-pt(z[pratios],lower.tail = F,df=length(ratios))
  limma_res[,paste0(cn,".z.perpop")]<-z
  rm(ratios,pratios,nratios,quants,p.values,z)
}
```

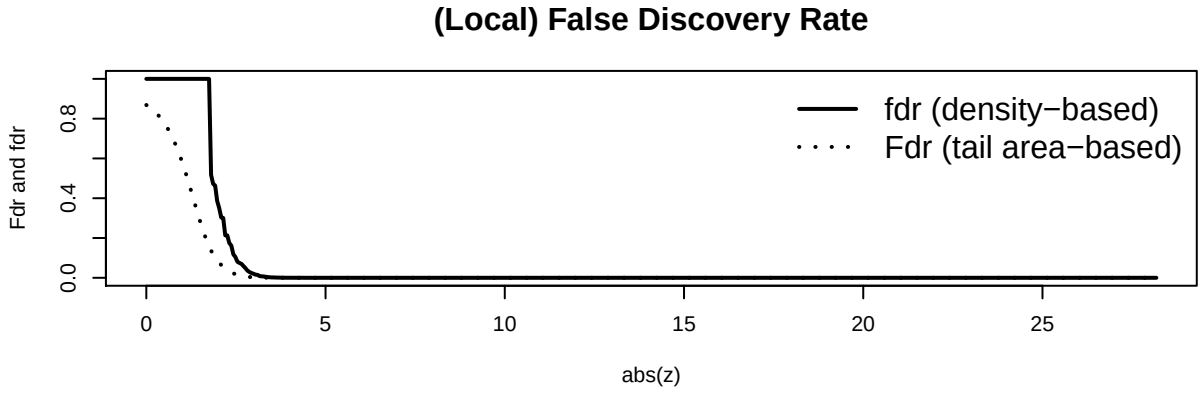
```
## [1] "logFC 35028"
```

```
rm(col.names,cn)
limma_res$expression.score<-limma_res$logFC.z.perpop
limma_res$fdr.local.expression.score<-limma_res$qval
limma_res$fdr.local.expression.score<-NA
for(i in 1:nrow(limma_res)){
  limma_res$fdr.local.expression.score[i]<-min(limma_res$qval[i],
                                               limma_res$adj.P.Val[i],na.rm=T)
}
rm(i)

limma_res$fdr.global.expression.score<-NA
limma_res$fdr.global.expression.score<-fdrtool(limma_res$expression.score)$qval
```

```
## Step 1... determine cutoff point
## Step 2... estimate parameters of null distribution and eta0
## Step 3... compute p-values and estimate empirical PDF/CDF
## Step 4... compute q-values and local fdr
## Step 5... prepare for plotting
```







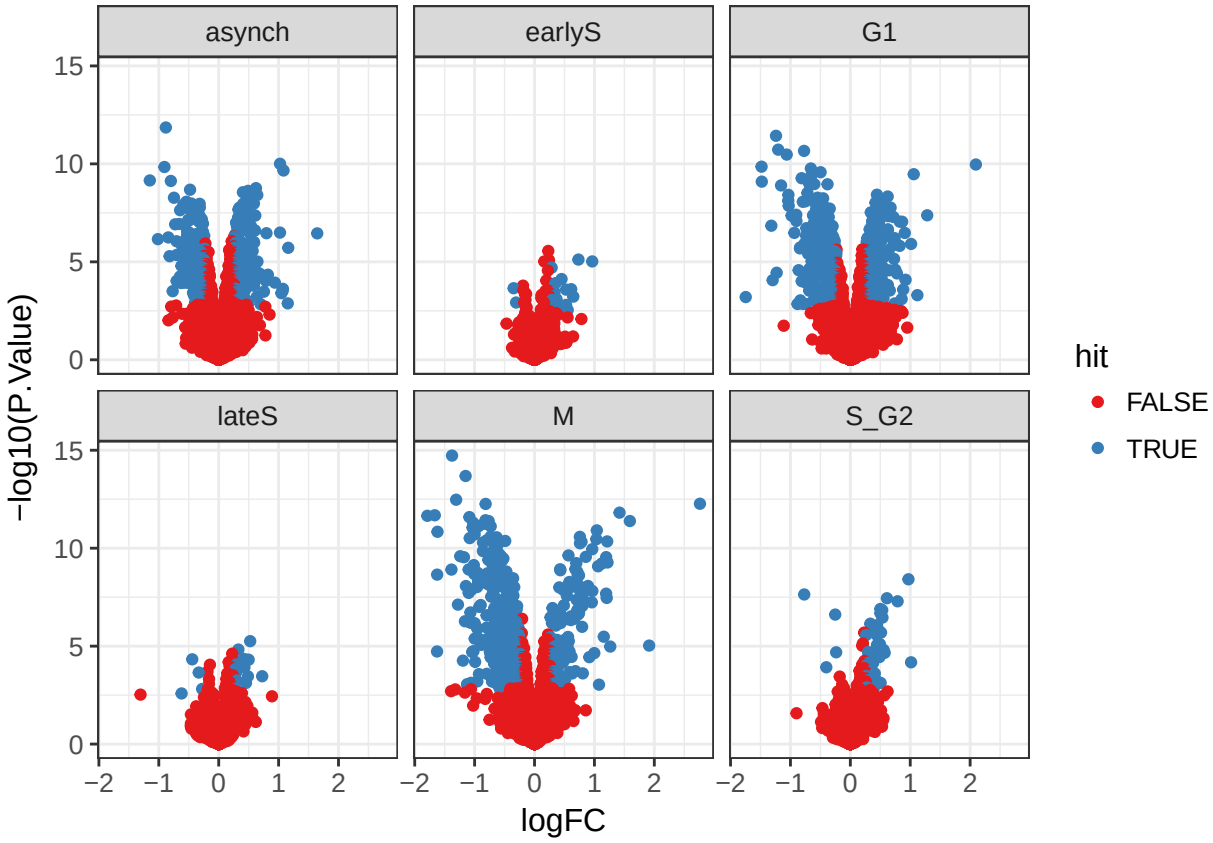

```
limma_res$hit<-with(limma_res,
                    ifelse(fdr.global.expression.score<=0.01&
                             fdr.local.expression.score<=0.01,T,F))
limma_res$cell.cycle<-gsub(" - G1_S","",limma_res$comparison)

score.data<-limma_res[,c("gene_name","cell.cycle","expression.score",
                         "fdr.local.expression.score","fdr.global.expression.score")]
mscore.data<-melt(score.data,id.vars=c("gene_name","cell.cycle"))

sum_score.data1<-ddply(.data = score.data,.variables = c("gene_name"),
                       summarise,min.global.fdr=min(fdr.global.expression.score,na.rm=T))
```

```
sum_score.data2<-ddply(.data = score.data,.variables = c("gene_name"),
                       summarise,min.local.fdr=min(fdr.local.expression.score,na.rm=T))
sum_score.data<-merge(sum_score.data1,sum_score.data2)
rm(sum_score.data1,sum_score.data2)

mscore.data<-merge(mscore.data,sum_score.data)
write.csv(mscore.data,file.path("processed_data","SDS_expression_score_data_V1.csv"),
          row.names = F)
```

Volcano plot

Volcano plots are plotted either with the local p.value or the tail area-based fdr (qval) estimated by the fdrtool function written by Strimmer et al. 2008).

```
qplot(logFC,-log10(P.Value),data=limma_res,colour=hit)+facet_wrap(~cell.cycle)+
  theme_bw(base_size=12)+scale_colour_brewer(palette="Set1")
```



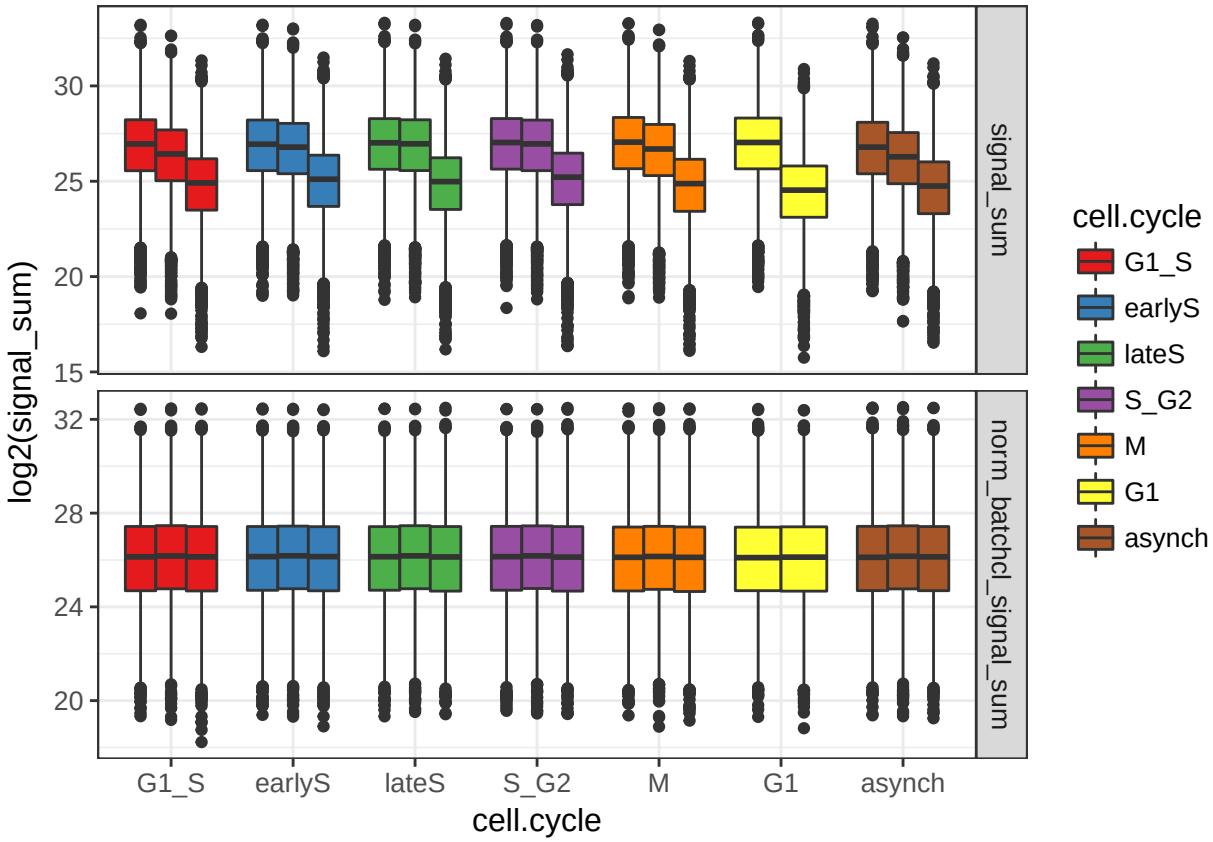
Reshaping full data set

```
mfc<-melt(cdata[,c("gene_name","m.top3","max.qupm",grep("^signal_sum",names(cdata),
                                value=T),grep("norm_batchcl_signal_sum",
                                 names(cdata),value=T),grep("cycr",
                                 names(cdata),value=T))],
          id.vars = c("gene_name","m.top3","max.qupm"))
```

```
mfc<-add_col_to_df(mfc,col_name = "measurement",data_col = "variable",
                   Vector=c("signal_sum","norm_batchcl_signal_sum","cycr","tempr"))
mfc$variable<-gsub(".+ignal_sum_","",mfc$variable)
mfc$variable<-gsub("_[0-9]+$","",mfc$variable)
mfc$variable<-gsub("_[0-9]+[H,L]$","",mfc$variable)
mfc$variable<-gsub(".cycr$","",mfc$variable)
mfc$variable<-gsub(".G1S","",mfc$variable)
mfc$variable<-gsub(".37.tempr$","",mfc$variable)
mfc$rep<-mfc$variable
mfc$variable<-gsub("_rep[0-9]+$","",mfc$variable)
names(mfc)[grep("variable",names(mfc))]<-c("condition")
mfc$experiment<-mfc$condition
mfc$condition<-gsub("SDS_","",mfc$condition)
mfc$rep<-gsub(".+_","",mfc$rep)
mfc<-na.omit(mfc)
mfc$measurement<-revalue(mfc$measurement,
                         replace = c("cycr"="G1S.ratio","tempr"="T37C.ratio"))
mfc2<-cast(mfc,formula=gene_name+m.top3+max.qupm+condition+rep~measurement,
           variable="value")
mfc2$cell.cycle<-gsub("_37","",mfc2$condition)
mfc3<-mfc2

conditions$condition<-with(conditions,paste0(cell.cycle,"_37"))
mfc2<-merge(mfc2,conditions)
mfc2$Temperature="37"
mfc<-merge(mfc,conditions)
```

Quality control of normalization

```
ggplot(data=subset(mfc,measurement%in%c("signal_sum","norm_batchcl_signal_sum")),
       aes(cell.cycle,log2(value),fill=cell.cycle))+facet_grid(measurement~.,scale="free")+
  geom_boxplot(aes(group=paste(condition,rep)))+theme_bw(base_size=12)+
  scale_fill_brewer(palette="Set1")+ylab("log2(signal_sum)")
```




```
ggsave(file.path("QC_plots","SDS_normalization_result.pdf"),width=15,height=10)
```

Create Excel-Sheet for data browsing

```
mfc3<-mfc2
mfc3<-subset(mfc3,signal_sum!=0)
#median G1S.ratio
odata<-cast(mfc3,formula=gene_name+Temperature~cell.cycle,value="G1S.ratio",
            fun.aggregate=median,na.rm=T)
names(odata)[!names(odata)%in%c("gene_name","Temperature")]<-
  paste0(names(odata)[!names(odata)%in%c("gene_name","Temperature")],".median")
#range G1S.ratio
odata2<-cast(mfc3,formula=gene_name+Temperature~cell.cycle,value="G1S.ratio",
             fun.aggregate=function(x){return(diff(range(log2(x),na.rm = T)))})
names(odata2)[!names(odata2)%in%c("gene_name","Temperature")]<-
  paste0(names(odata2)[!names(odata2)%in%c("gene_name","Temperature")],".log2range")
odata<-merge(odata,odata2)

#found in replicates
odata2<-cast(mfc3,formula=gene_name+Temperature~cell.cycle,value="G1S.ratio",
             fun.aggregate=length)
odata2$found.in.reps<-odata2$G1_S
odata2<-odata2[,c("gene_name","Temperature","found.in.reps")]
odata<-merge(odata,odata2)
rm(odata2)
```

```
#
modata<-cdata[,c("gene_name","m.top3",grep(".qupm",names(cdata),value=T))]
odata<-merge(odata,modata)
rm(modata)

des<-data[,c("gene_name","description")]
des<-subset(des,gene_name%in%unique(odata$gene_name))
des<-unique(des)
dupnames<-names(which(sort(table(des$gene_name),decreasing = T)>1))
des.dups<-subset(des,gene_name%in%dupnames)
des<-subset(des,!gene_name%in%dupnames)
des.dups.new<-NULL
for(gene in as.character(unique(des.dups$gene_name))){
  sub<-subset(des.dups,gene_name==gene)
  sub<-sub[which(nchar(sub$description)==max(nchar(sub$description)))[1],]
  sub<-data.frame(gene_name=sub$gene_name,description=sub$description)
  des.dups.new<-rbind(des.dups.new,sub)
  rm(sub)
}
des<-rbind(des,des.dups.new)
rm(des.dups.new,des.dups,dupnames,gene)
odata<-merge(odata,des)
rm(des)
odata<-odata[,c("gene_name","description","found.in.reps", "m.top3", "max.qupm",
                "Temperature","G1_S.median", "earlyS.median", "lateS.median", "S_G2.median",
                "M.median", "G1.median","asynch.median", "G1_S.log2range", "earlyS.log2range",
                "lateS.log2range", "S_G2.log2range", "M.log2range", "G1.log2range",
                "asynch.log2range")]

odata<-merge(odata,sum_score.data)
odata$hit<-with(odata,ifelse(min.global.fdr<=0.01&min.local.fdr<=0.01,"expression hit",""))

hit.class<-score.data
hit.class$hit<-ifelse(hit.class$fdr.local.expression.score<=0.01&hit.class$
                        fdr.global.expression.score<=0.01,
                      "expression hit","")
hit.class<-hit.class[,c("gene_name","cell.cycle","hit")]
hit.class$hit.name<-"hit"
chit.class<-cast(hit.class,formula=gene_name~hit.name+cell.cycle,value = "hit")
rm(hit.class)
odata<-merge(odata,chit.class)
rm(chit.class)
mscd<-melt(score.data,id.vars=c("gene_name","cell.cycle"))
mscd$cell.cycle<-factor(mscd$cell.cycle,ordered=T,
                        levels=c("earlyS","lateS","S_G2","M","G1","asynch","G1_S"))
cscd<-cast(mscd,formula=gene_name~variable+cell.cycle)
odata<-merge(odata,cscd)
rm(mscd,cscd)
odata$gene_name<-paste0("'",odata$gene_name)

write.csv(odata,file.path("processed_data","SDS_Final_cell_cycle_results_V1.csv"),
          row.names = F)
rm(mfc3,odata,mfc2)
```

Clean global environment

```
rm(cdata,data,mdata,mfc,conditions,mscore.data,score.data,sum_score.data,limma_res)
rm(raw_dataE,batchcl_raw_dataE,norm_batchcl_dataE)
```

Calculate abundance and stability score

Loading data and dplyr library

```
library(dplyr)
mdata<-read.csv(file.path("processed_data","2D_TPP_mdata_V1.csv"))
```

Defining quality criteria for proteins

Only proteins that have at least 2 data points for each of the first Temperatures (37 and 40.4 degrees) were used to calculate scores.

```
score.data.length<-mdata%>%
  filter(!is.na(G1S.ratio),Temperature%in%c(37,40.4))%>%
  group_by(gene_name,cell.cycle,Temperature)%>%
  summarise(rep=length(G1S.ratio))%>%
  filter(rep>=2)%>%
  select(gene_name,cell.cycle)%>%
  group_by(gene_name,cell.cycle)%>%
  summarise(Freq=length(cell.cycle))%>%
  filter(Freq>=2)%>%
  select(gene_name,cell.cycle)%>%
  mutate(score.valid=T)
mdata<-left_join(mdata,score.data.length)
rm(score.data.length)
score.data.length<-mdata%>%
  filter(!is.na(G1S.ratio))%>%
  group_by(gene_name,cell.cycle,Temperature)%>%
  summarise(rep=length(G1S.ratio))%>%
  filter(rep>=2)%>%
  mutate(bootstrap.valid.T=T)%>%
  select(gene_name,cell.cycle,Temperature,bootstrap.valid.T)
mdata<-left_join(mdata,score.data.length)
rm(score.data.length)
```

Computing abundance and stability score using a bootstrap approach

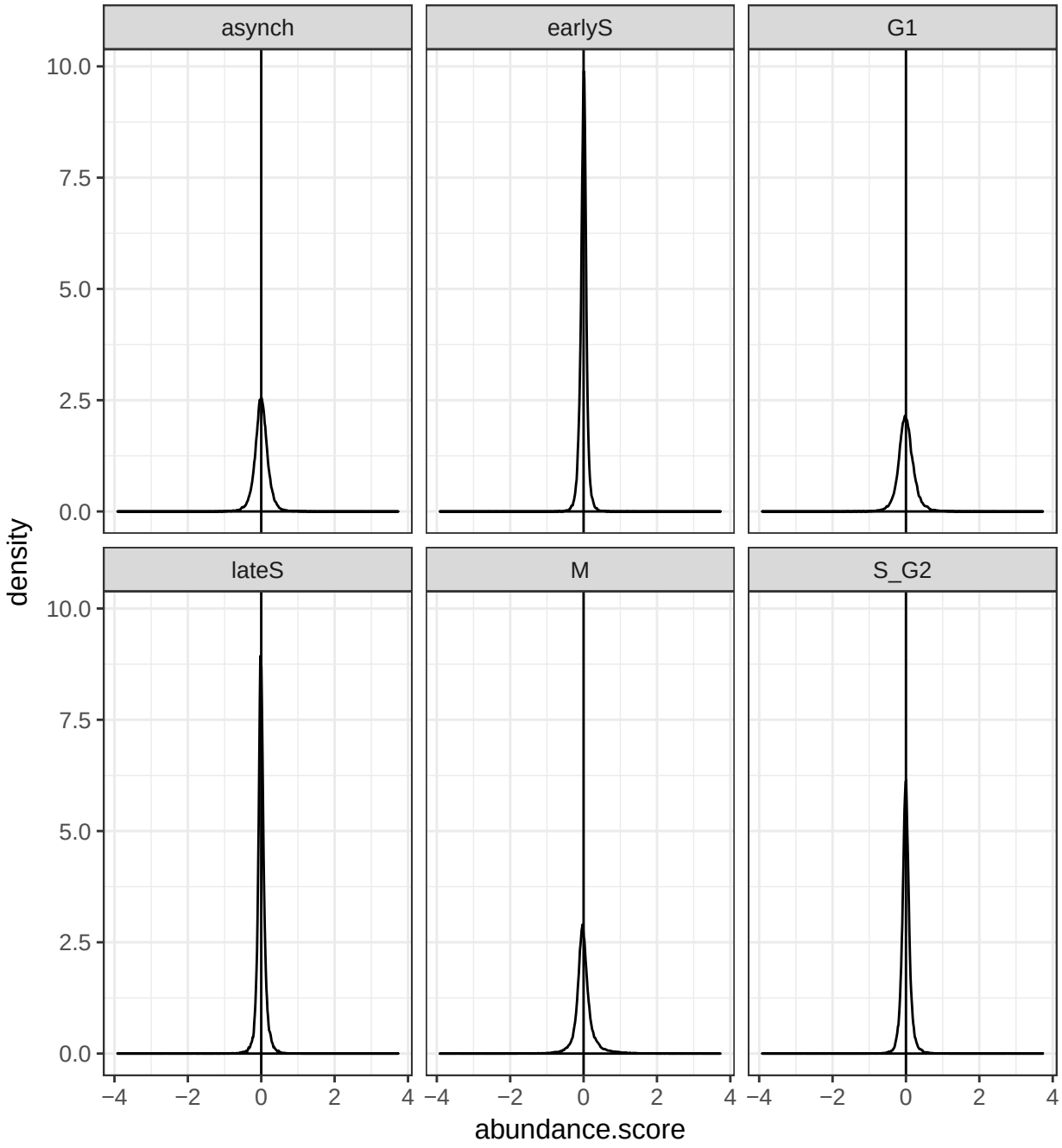
We chose 500 iterations.

```
score.data.sub.sum<-NULL
for(i in 1:500){
  score.data.sub<-mdata%>%
    filter(!is.na(G1S.ratio),score.valid,bootstrap.valid.T)%>%
    group_by(gene_name,cell.cycle,Temperature)%>%
    summarise(ratio=sampleWithoutSurprises(G1S.ratio))
```

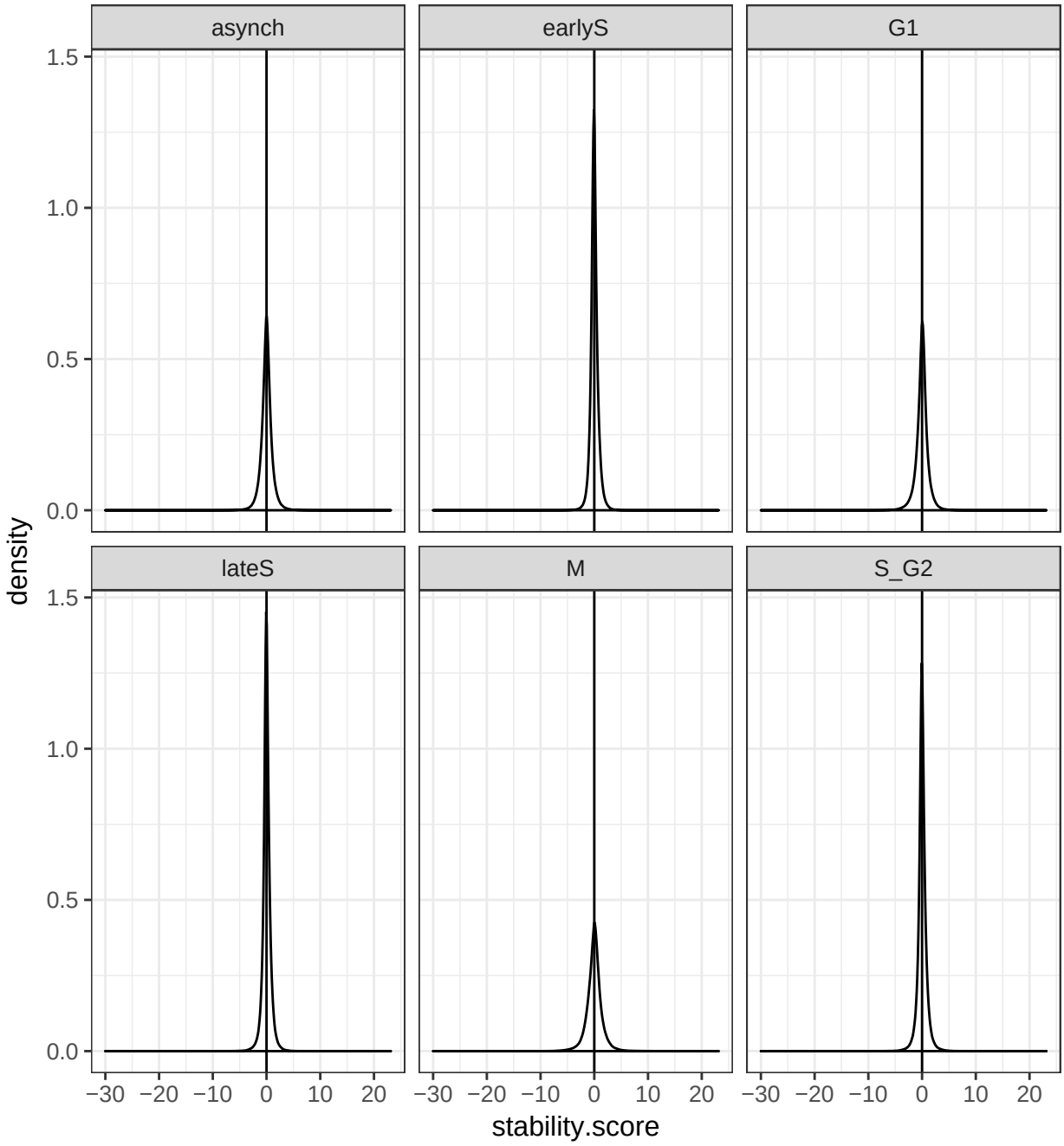
```
  abundance.score.sub<-score.data.sub%>%
    filter(Temperature%in%c(37,40.4))%>%
    group_by(gene_name,cell.cycle)%>%
    summarise(abundance.score=mean(log2(ratio),na.rm=T))
  score.data.sub<-left_join(score.data.sub,abundance.score.sub)
  score.data.sub<-score.data.sub%>%
    mutate(stability=log2(ratio)-abundance.score)
  stability.score.sub<-score.data.sub%>%
    group_by(gene_name,cell.cycle)%>%
    summarise(stability.score=sum(stability,na.rm=T))
  score.data.sub<-left_join(score.data.sub,stability.score.sub)
  rm(abundance.score.sub,stability.score.sub)
  score.data.sub<-score.data.sub%>%
    select(gene_name,cell.cycle,abundance.score,stability.score)
  score.data.sub<-unique(score.data.sub)
  score.data.sub$N<-i
  score.data.sub.sum<-bind_rows(score.data.sub.sum,score.data.sub)
  rm(score.data.sub)
}
```

Plotting distributions of stability and abundance scores

```
qplot(abundance.score,data=subset(score.data.sub.sum,cell.cycle!="G1_S"),geom="density")+
  facet_wrap(~cell.cycle)+geom_vline(aes(xintercept=0))+theme_bw(base_size=12)
```



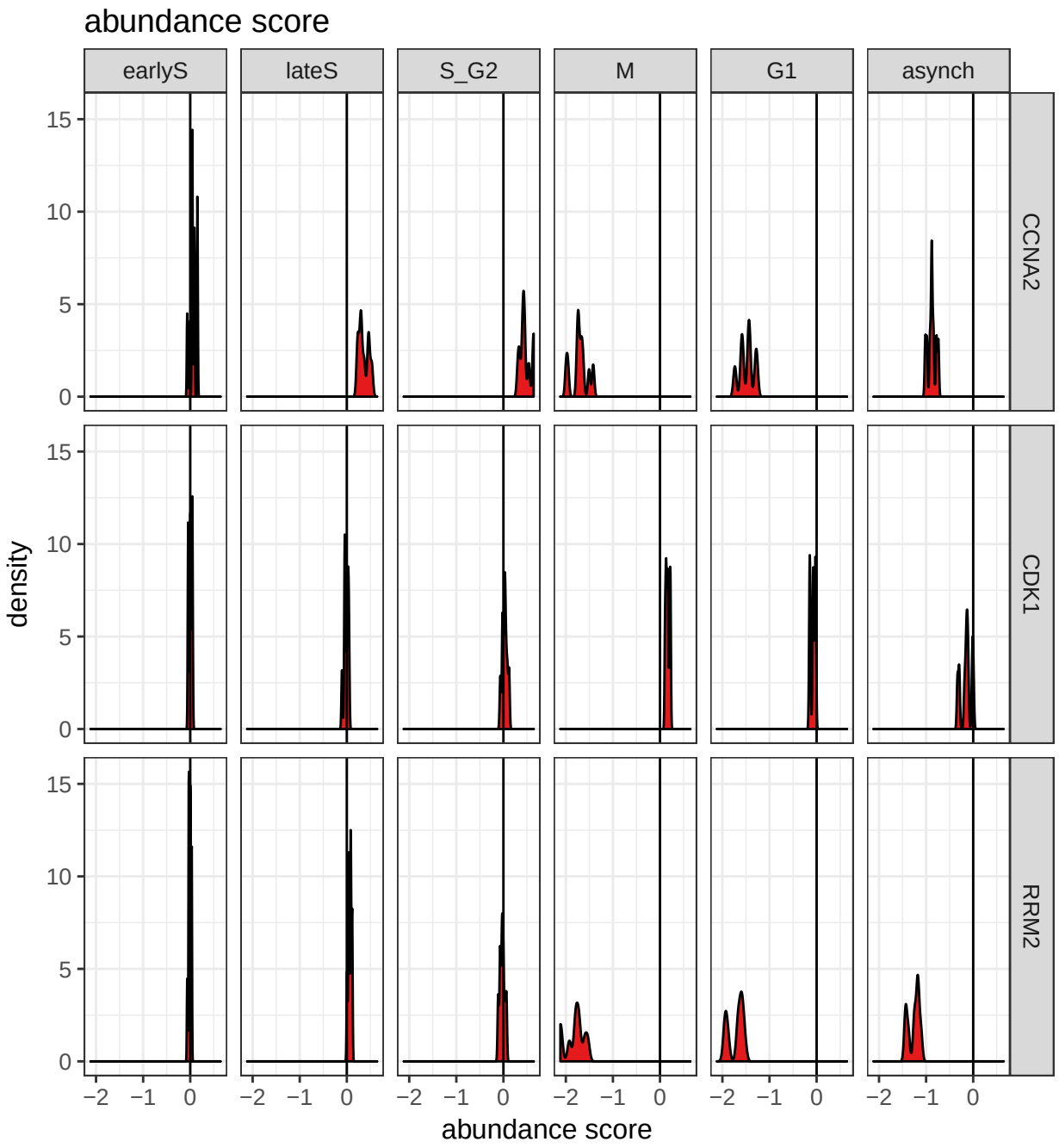

```
qplot(stability.score,data=subset(score.data.sub.sum,cell.cycle!="G1_S"),geom="density")+
  facet_wrap(~cell.cycle)+geom_vline(aes(xintercept=0))+theme_bw(base_size=12)
```



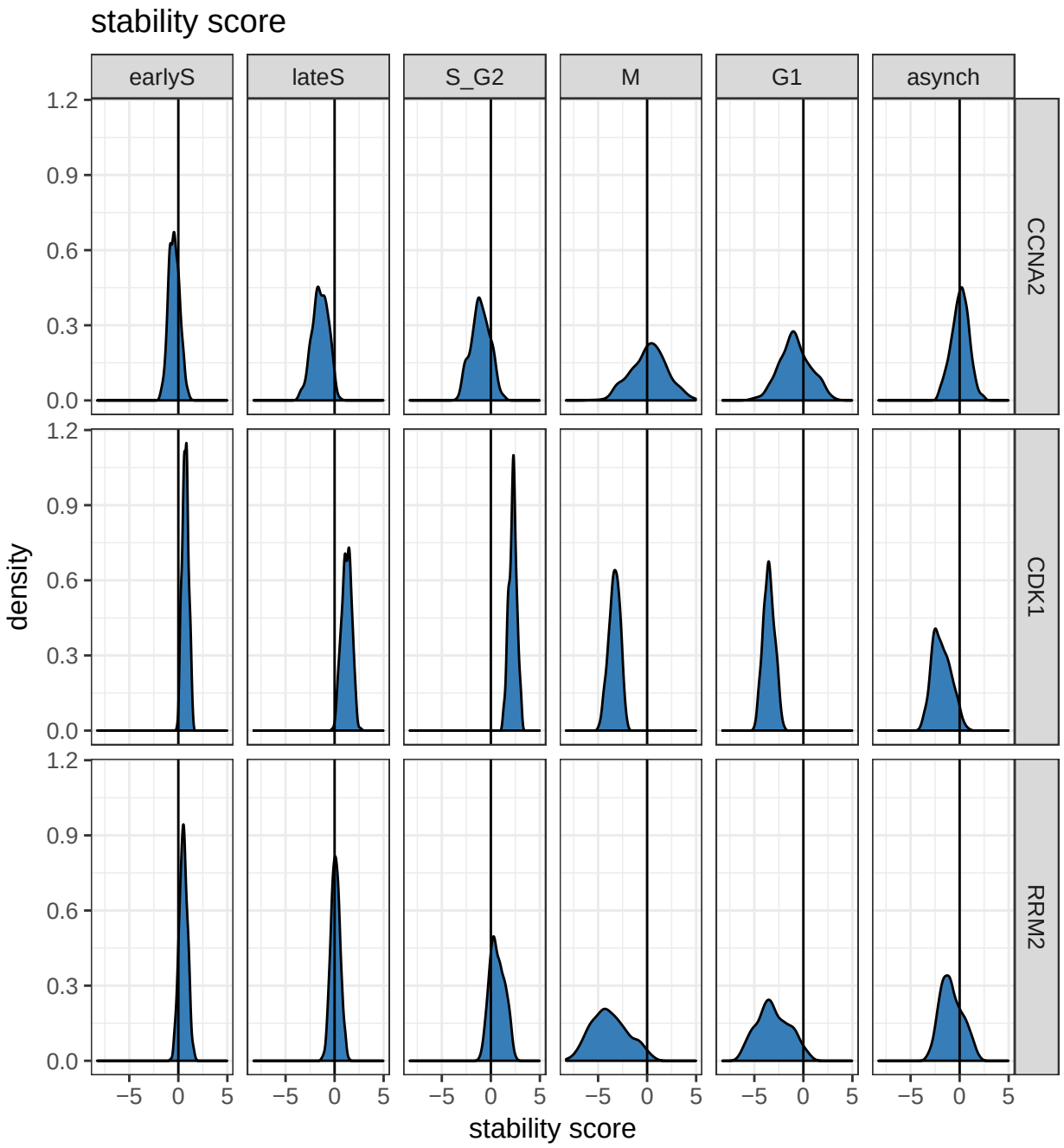
Plotting example bootstrap distribution for CDK1, CCNA2 and RRM2

```
sub<-subset(score.data.sub.sum,gene_name%in%c("CDK1","CCNA2","RRM2")&cell.cycle!="G1_S")
sub<-reshape::melt(sub,id.vars=c("gene_name","cell.cycle","N"))
sub$cell.cycle<-factor(sub$cell.cycle,ordered=T,
                       levels=c("earlyS","lateS","S_G2","M","G1","asynch","G1_S"))
ggplot(aes(value),data=subset(sub,variable=="abundance.score"))+
  geom_density(fill="#e41a1c",colour="black")+facet_grid(gene_name~cell.cycle)+
  ggtitle("abundance score")+geom_vline(aes(xintercept=0))+xlab("abundance score")+
```

```
  theme_bw(base_size=12)
```




```
ggsave(file.path("QC_plots","bootstrap_examples_abundance_scores.pdf"),width=15,height=15)
ggplot(aes(value),data=subset(sub,variable=="stability.score"))+
  geom_density(fill="#377eb8",colour="black")+facet_grid(gene_name~cell.cycle)+
  ggtitle("stability score")+geom_vline(aes(xintercept=0))+xlab("stability score")+
  theme_bw(base_size=12)
```




```
ggsave(file.path("QC_plots","bootstrap_examples_stability_scores.pdf"),width=15,height=15)
```

Summarizing bootstrap distributions

Calculating mean and standard deviation from bootstrap distributions

```
score.data<-score.data.sub.sum%>%
  group_by(gene_name,cell.cycle)%>%
  summarise(bootstrap.abundance.score.mean=mean(abundance.score,na.rm=T),
```

```
            bootstrap.stability.score.mean=mean(stability.score,na.rm=T),
            bootstrap.abundance.score.sd=sd(abundance.score,na.rm=T),
            bootstrap.stability.score.sd=sd(stability.score,na.rm=T))
```

Estimating number of replicates going into bootstrapping algorithm

```
score.data.length<-mdata%>%
  filter(!is.na(G1S.ratio),Temperature%in%c(37,40.4))%>%
  group_by(gene_name,cell.cycle)%>%
  summarise(N.abundance.score=length(G1S.ratio))
score.data<-left_join(score.data,score.data.length)
score.data.length<-mdata%>%
  filter(!is.na(G1S.ratio))%>%
  group_by(gene_name,cell.cycle)%>%
  summarise(N.stability.score=length(G1S.ratio))
score.data<-left_join(score.data,score.data.length)
rm(score.data.length)
```

Saving bootstrap results

```
write.csv(score.data.sub.sum,file.path("processed_data","2D_TPP_simulated_scores_data_V2.csv"))
write.csv(score.data,file.path("processed_data","2D_TPP_simulated_scores_data_summary_V2.csv"))
```

Transforming stability and abundance scores into z-distribution

Transform global score values

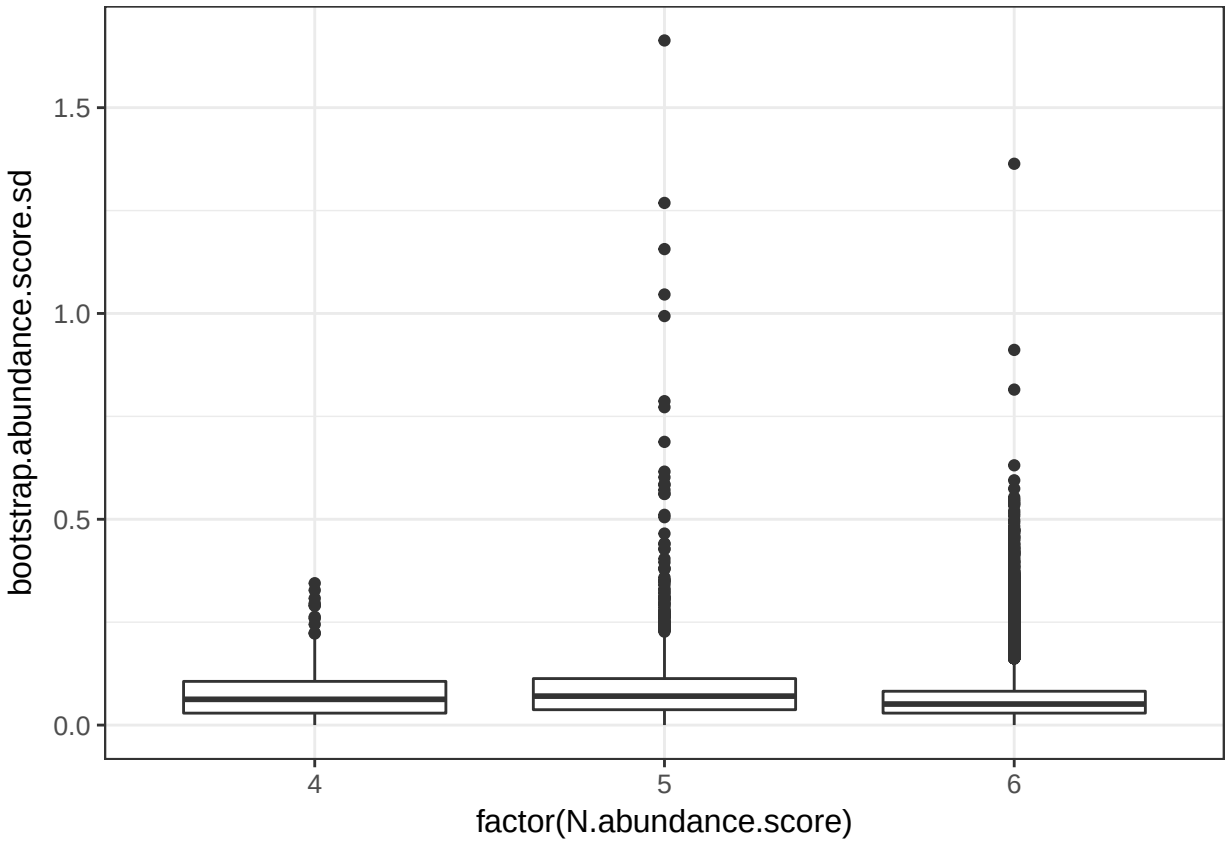
```
rm(score.data.sub.sum)
col.names<-c("bootstrap.abundance.score.mean","bootstrap.stability.score.mean")
for(cn in col.names){
  ratios<-score.data[,cn]
  ratios<-as.double(as.character(unlist(ratios)))
  print(paste(cn,length(ratios)))
  quants<-quantile(ratios,probs = c(0.1587,0.5,0.8413),na.rm=T)
  ratios[is.na(ratios)]<-0
  pratios<-ratios>=0
  nratios<-ratios<0
  z<-ratios
  z[nratios]<-ratios[nratios]/as.numeric(diff(quants)[1])
  z[pratios]<-ratios[pratios]/as.numeric(diff(quants)[2])
  score.data[,paste0(cn,".z.perpop")]<-z
  rm(ratios,pratios,nratios,quants,z)
}
```

```
## [1] "bootstrap.abundance.score.mean 34727"
## [1] "bootstrap.stability.score.mean 34727"
```

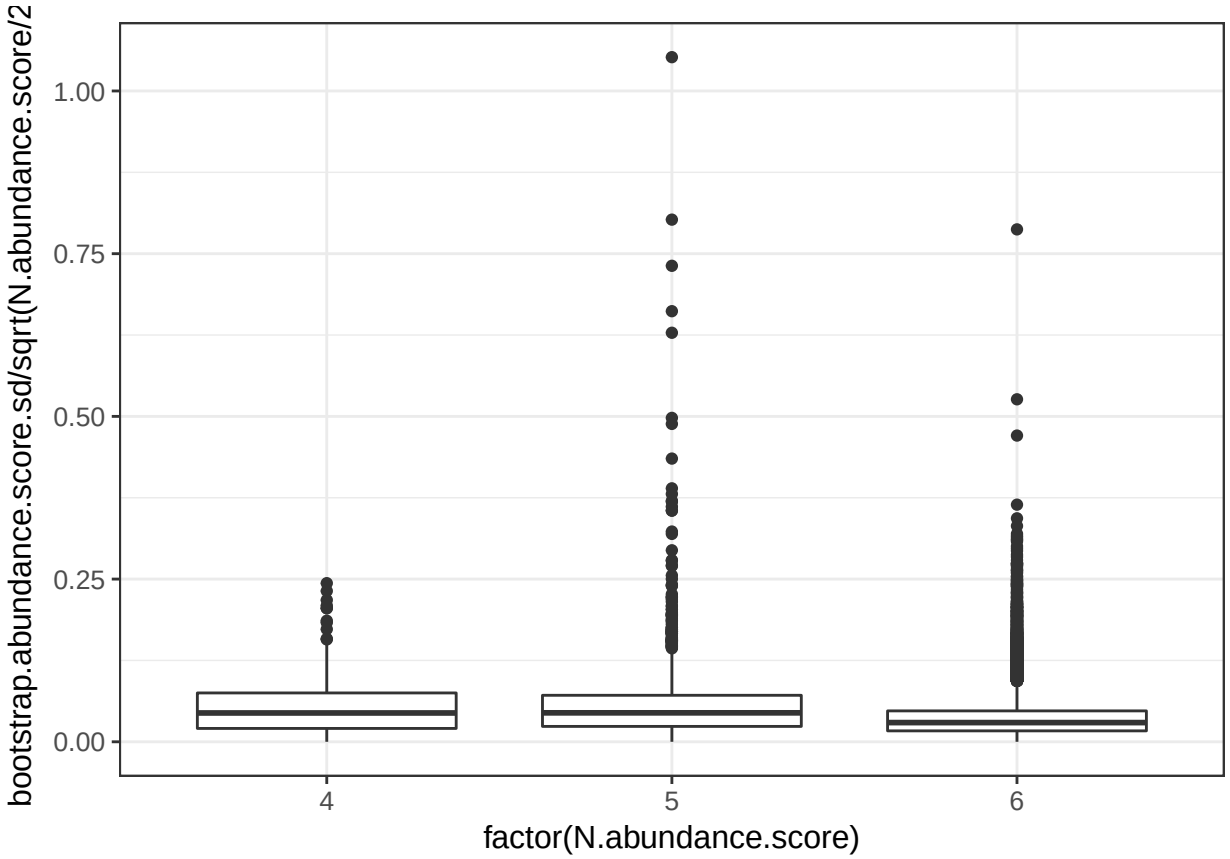
```
rm(col.names,cn)
```

Transform local score values

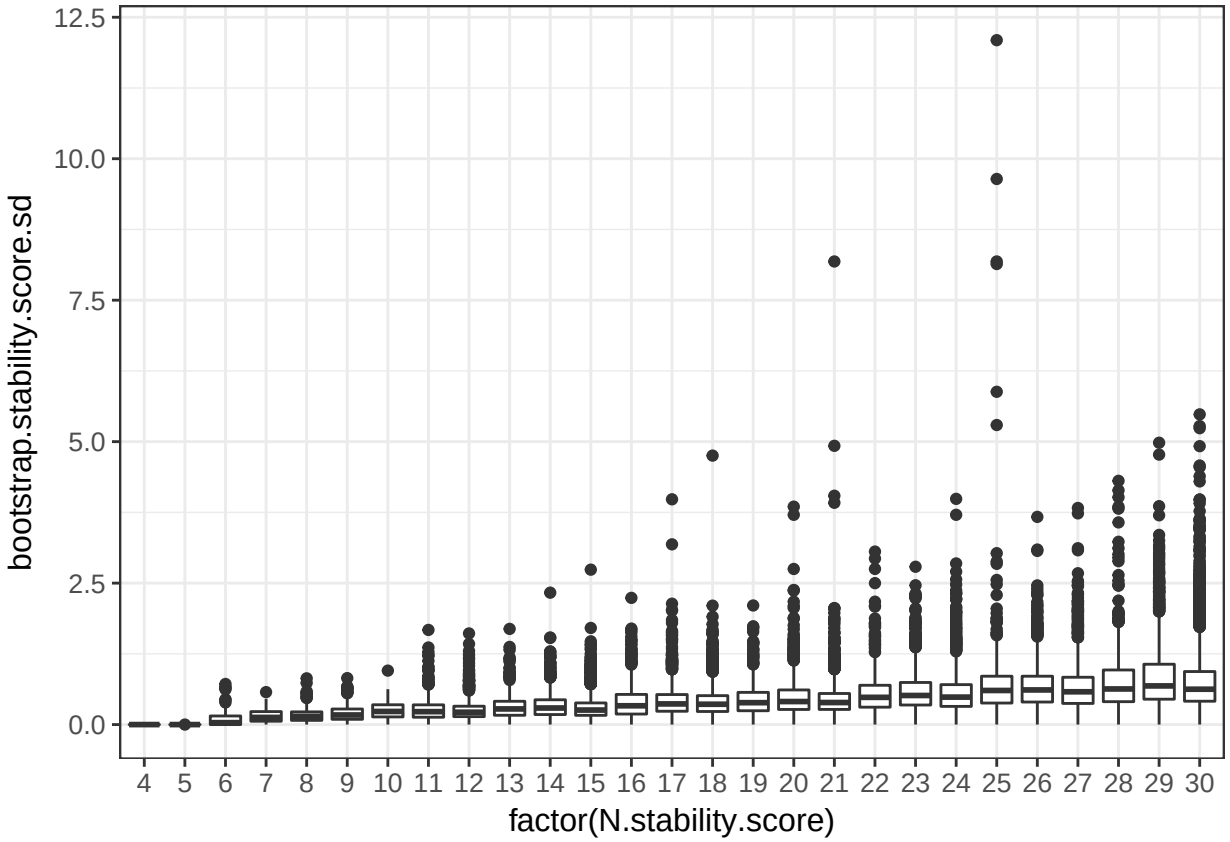
```
score.data$bootstrap.abundance.score.z.perprot<-
  score.data$bootstrap.abundance.score.mean/
  (score.data$bootstrap.abundance.score.sd/sqrt(score.data$N.abundance.score/2))
qplot(factor(N.abundance.score),bootstrap.abundance.score.sd,data=score.data,geom="boxplot")+
  theme_bw(base_size=12)
```



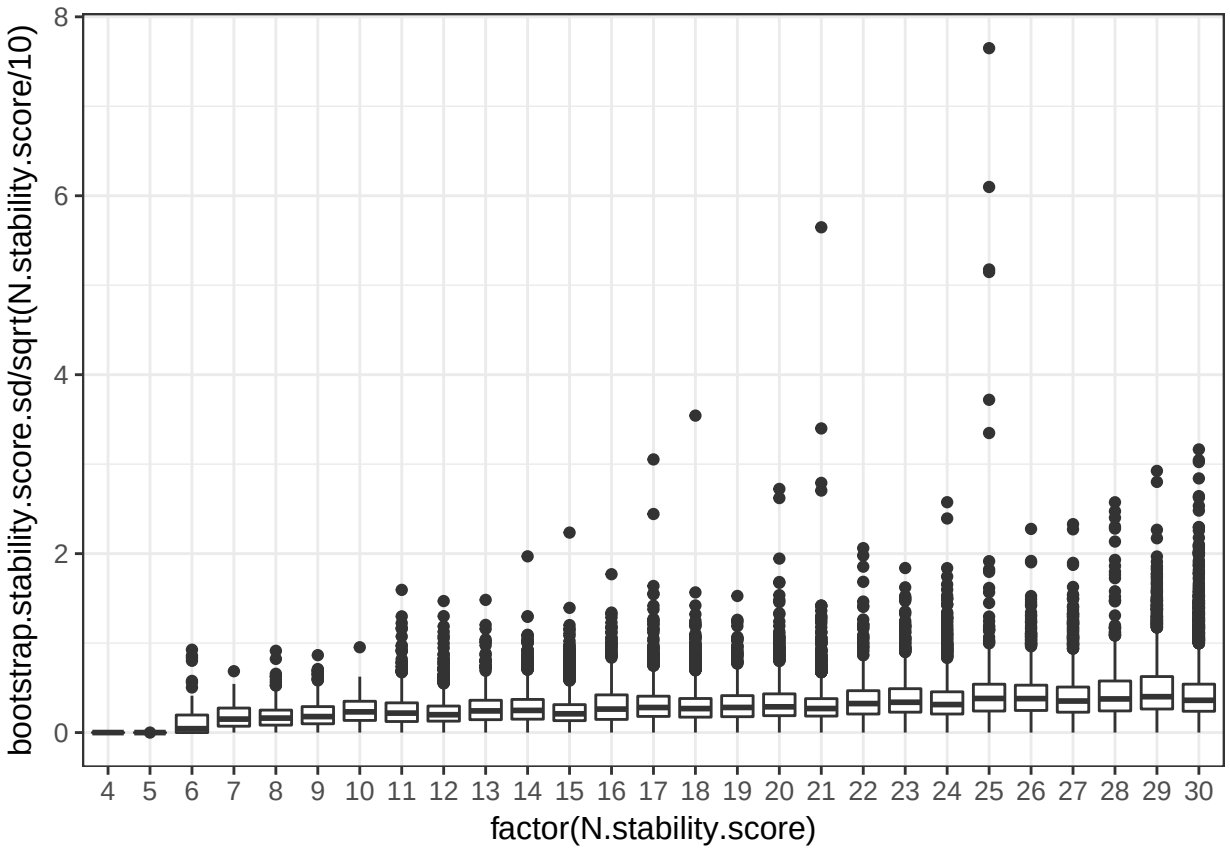

```
qplot(factor(N.abundance.score),bootstrap.abundance.score.sd/
        sqrt(N.abundance.score/2),
      data=score.data,geom="boxplot")+
  theme_bw(base_size=12)
```




```
score.data$bootstrap.stability.score.z.perprot<-
  score.data$bootstrap.stability.score.mean/(score.data$bootstrap.stability.score.sd/
                                               sqrt(score.data$N.stability.score/10))
qplot(factor(N.stability.score),bootstrap.stability.score.sd,
      data=score.data,geom="boxplot")+
  theme_bw(base_size=12)
```



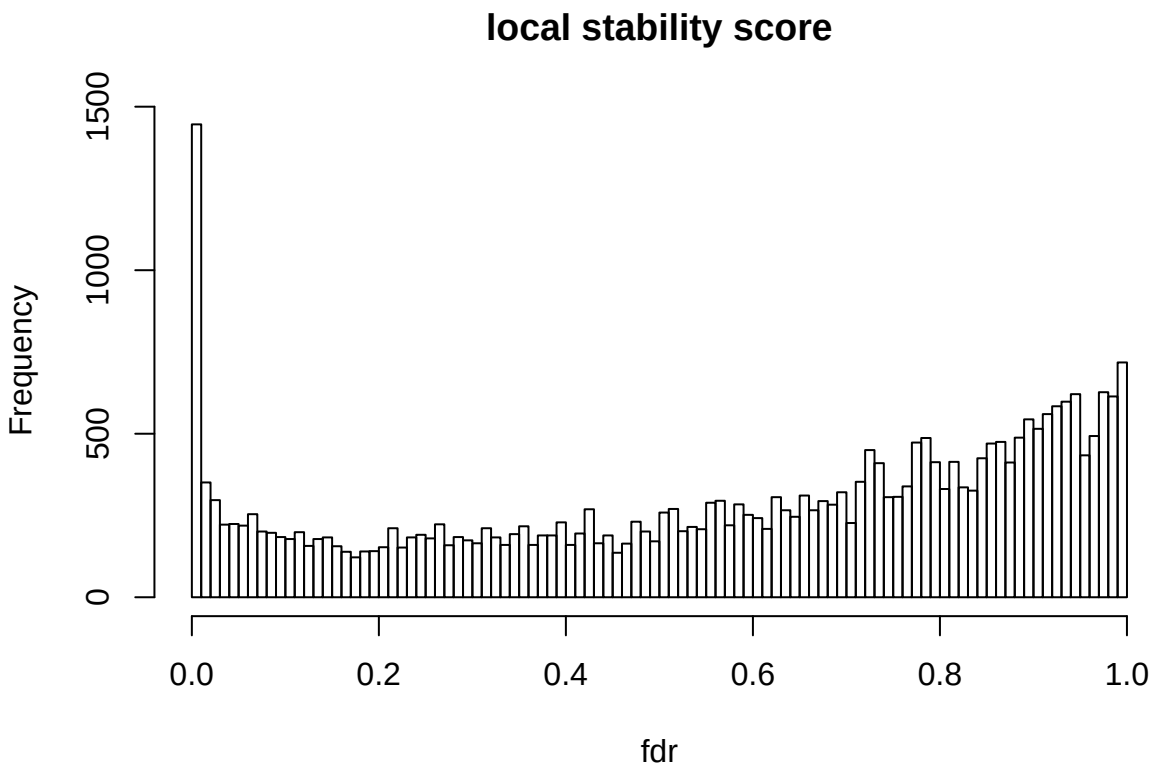

```
qplot(factor(N.stability.score),bootstrap.stability.score.sd/
        sqrt(N.stability.score/10),
      data=score.data,geom="boxplot")+
  theme_bw(base_size=12)
```



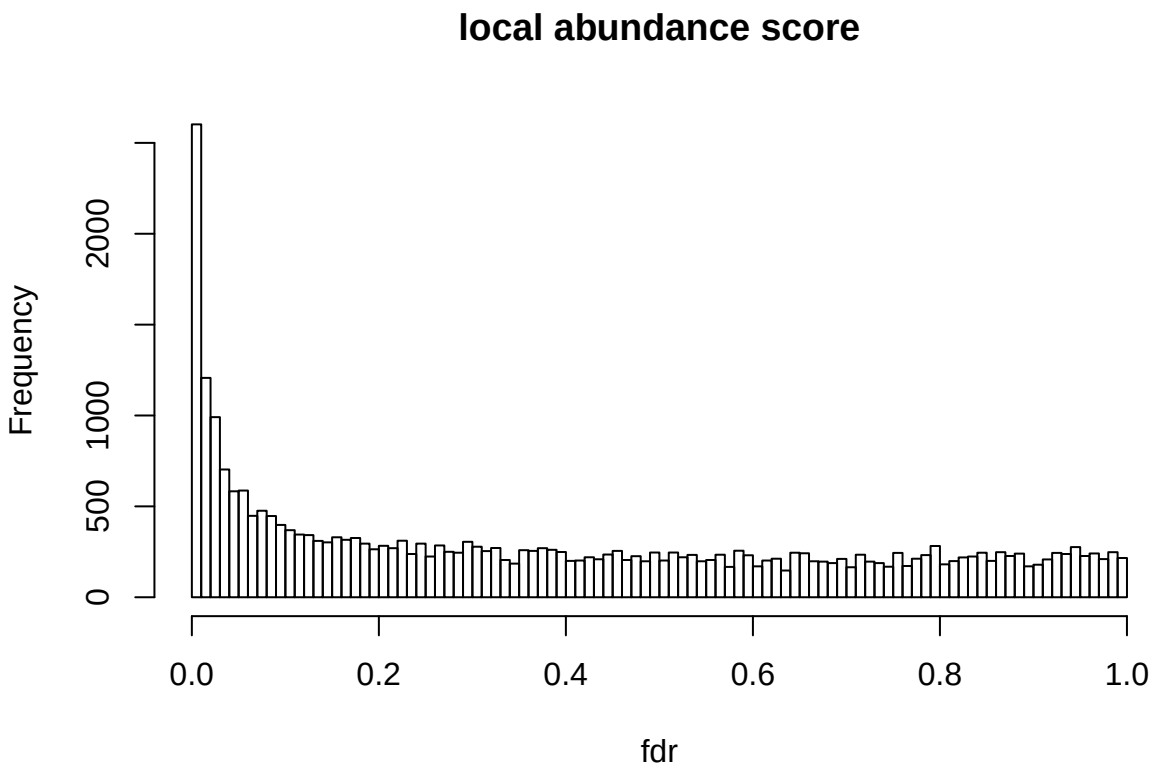
Assigning p-values and false discovery rates to different z-distributions of abundance and stability scores

Local p-values

```
score.data$fdr.bootstrap.stability.score.perprot<-NA
for(i in 1:nrow(score.data)){
  if(!is.na(score.data$bootstrap.stability.score.z.perprot[i])){
    score.data$fdr.bootstrap.stability.score.perprot[i]<-
      pt(abs(score.data$bootstrap.stability.score.z.perprot[i]),
         df = score.data$N.stability.score[i]-1,lower.tail = F)*2
  }
}
rm(i)
for(cycle in unique(score.data$cell.cycle)){
  score.data$fdr.bootstrap.stability.score.perprot[score.data$cell.cycle==cycle]<-
    p.adjust(score.data$fdr.bootstrap.stability.score.perprot[score.data$cell.cycle==cycle],
             method = "fdr")
}
rm(cycle)
hist(score.data$fdr.bootstrap.stability.score.perprot,breaks=100,
     main = "local stability score",
     xlab = "fdr")
```



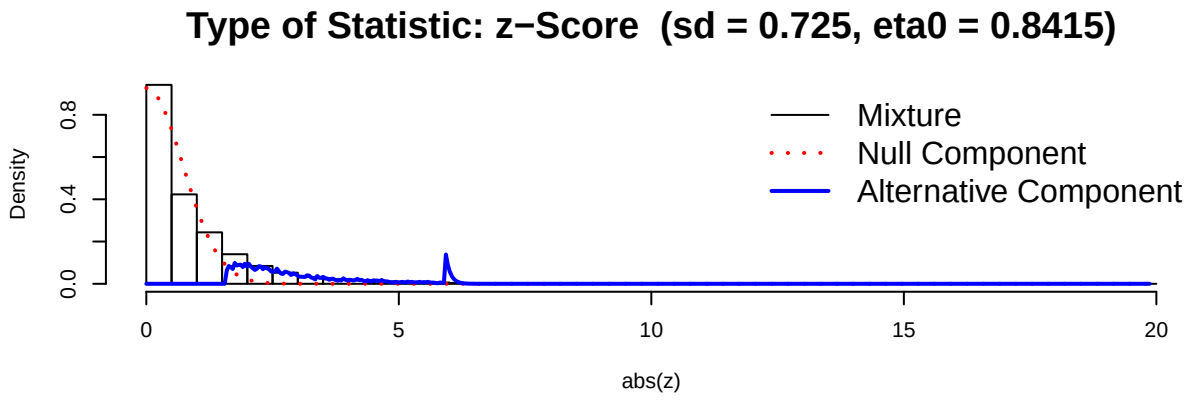

```
score.data$fdr.bootstrap.abundance.score.perprot<-NA
for(i in 1:nrow(score.data)){
  if(!is.na(score.data$bootstrap.abundance.score.z.perprot[i])){
    score.data$fdr.bootstrap.abundance.score.perprot[i]<-
      pt(abs(score.data$bootstrap.abundance.score.z.perprot[i]),
         df = score.data$N.abundance.score[i]-1,lower.tail = F)*2
  }
}
rm(i)
for(cycle in unique(score.data$cell.cycle)){
  score.data$fdr.bootstrap.abundance.score.perprot[score.data$cell.cycle==cycle]<-
    p.adjust(score.data$fdr.bootstrap.abundance.score.perprot[score.data$cell.cycle==cycle],
             method = "fdr")
}
rm(cycle)
hist(score.data$fdr.bootstrap.abundance.score.perprot,breaks=100,
     main = "local abundance score",xlab = "fdr")
```

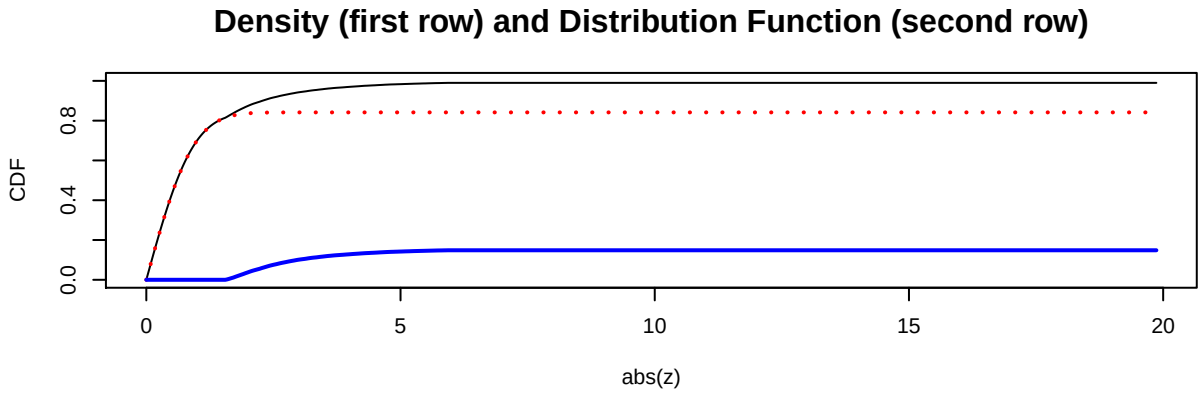


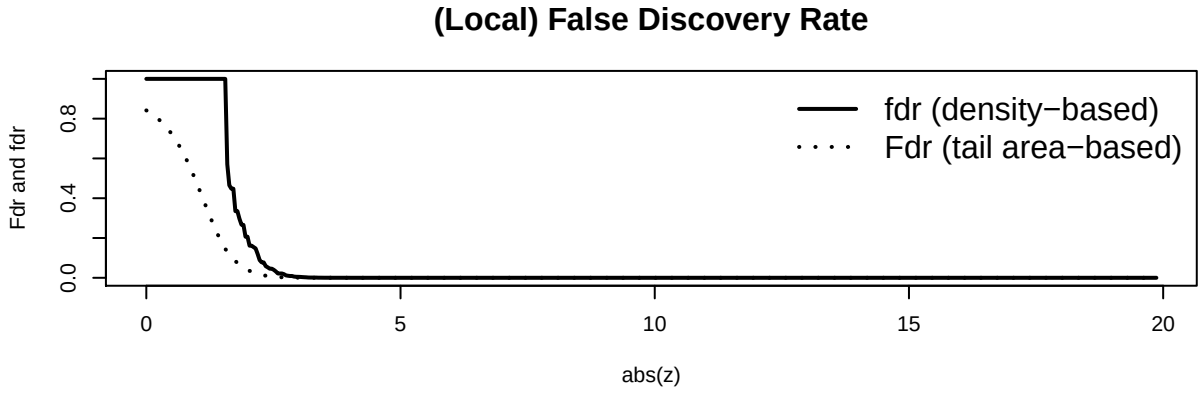
Global p-values

```
score.data$qval.bootstrap.stability.score.perpop<-NA
score.data$qval.bootstrap.stability.score.perpop[!is.na(
  score.data$bootstrap.stability.score.mean.z.perpop)]<-
  with(score.data[!is.na(score.data$bootstrap.stability.score.mean.z.perpop),],
       fdrtool(bootstrap.stability.score.mean.z.perpop))$qval
```

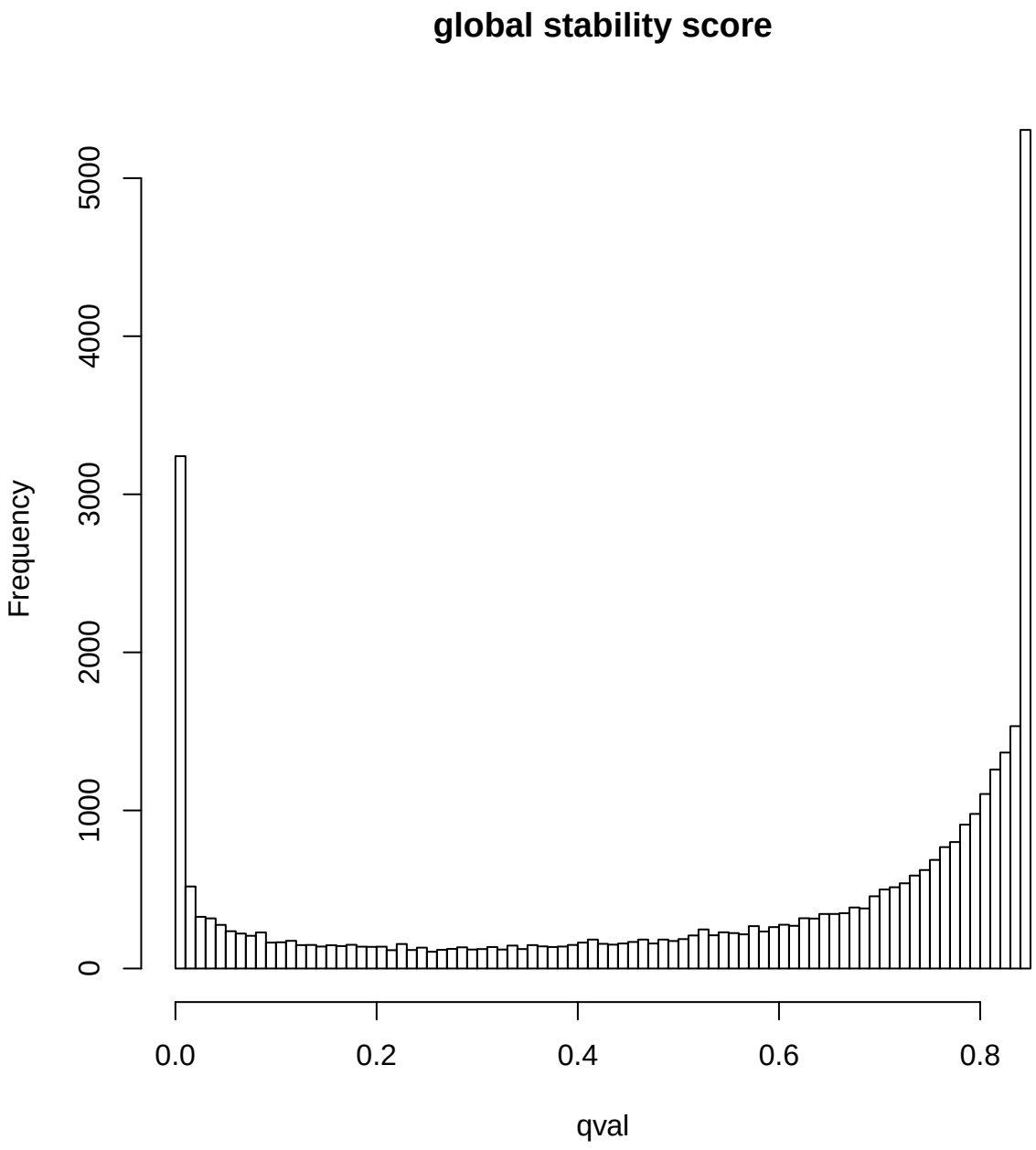
```
## Step 1... determine cutoff point
## Step 2... estimate parameters of null distribution and eta0
## Step 3... compute p-values and estimate empirical PDF/CDF
## Step 4... compute q-values and local fdr
## Step 5... prepare for plotting
```





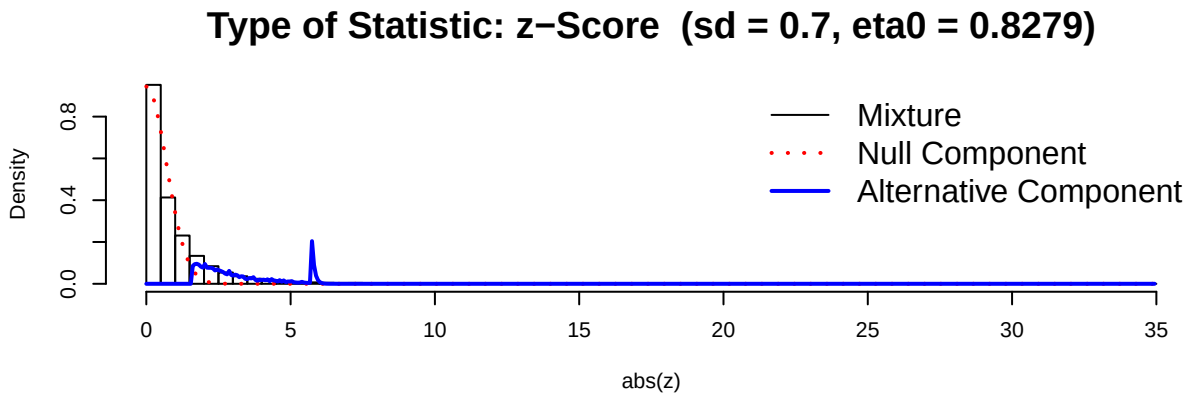


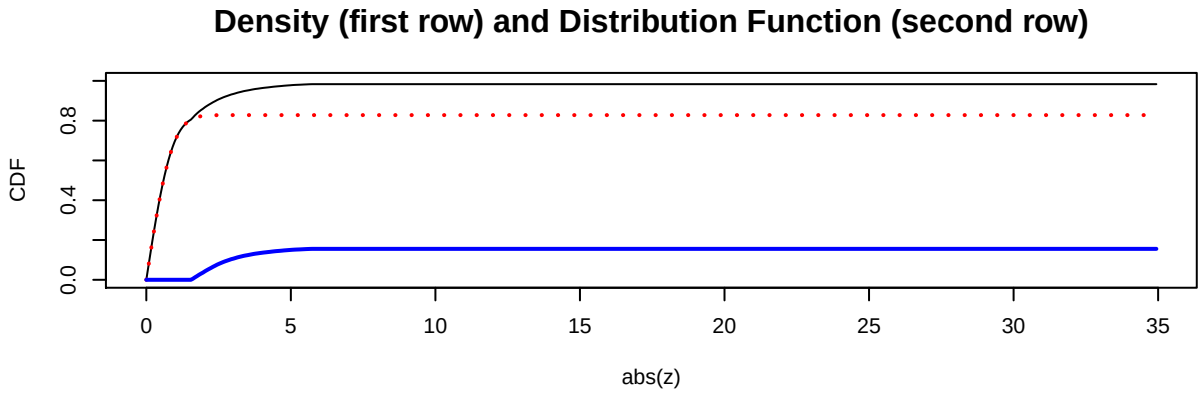

```
hist(score.data$qval.bootstrap.stability.score.perpop,breaks=100,
     main = "global stability score",xlab="qval")
```

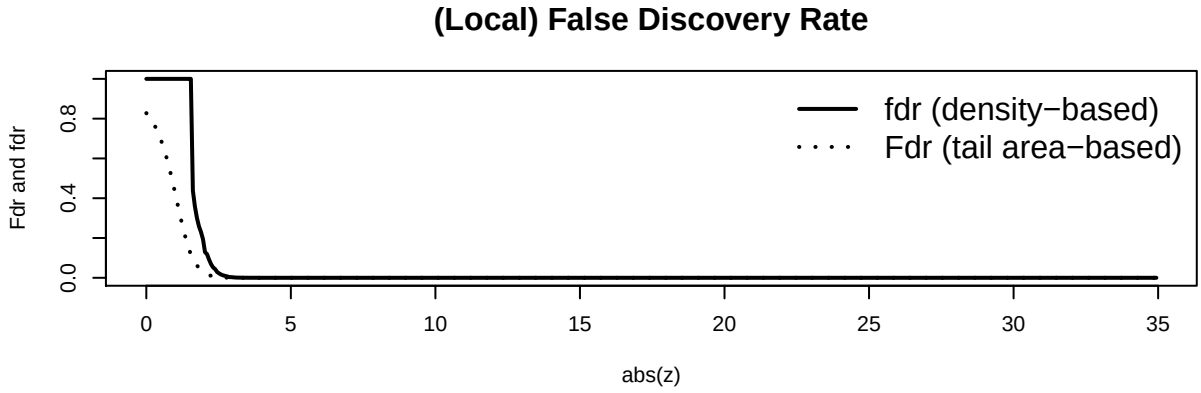



```
score.data$qval.bootstrap.abundance.score.perpop<-NA
score.data$qval.bootstrap.abundance.score.perpop[!is.na(
  score.data$bootstrap.abundance.score.mean.z.perpop)]<-
  with(score.data[!is.na(score.data$bootstrap.abundance.score.mean.z.perpop),],
       fdrtool(bootstrap.abundance.score.mean.z.perpop))$qval
```

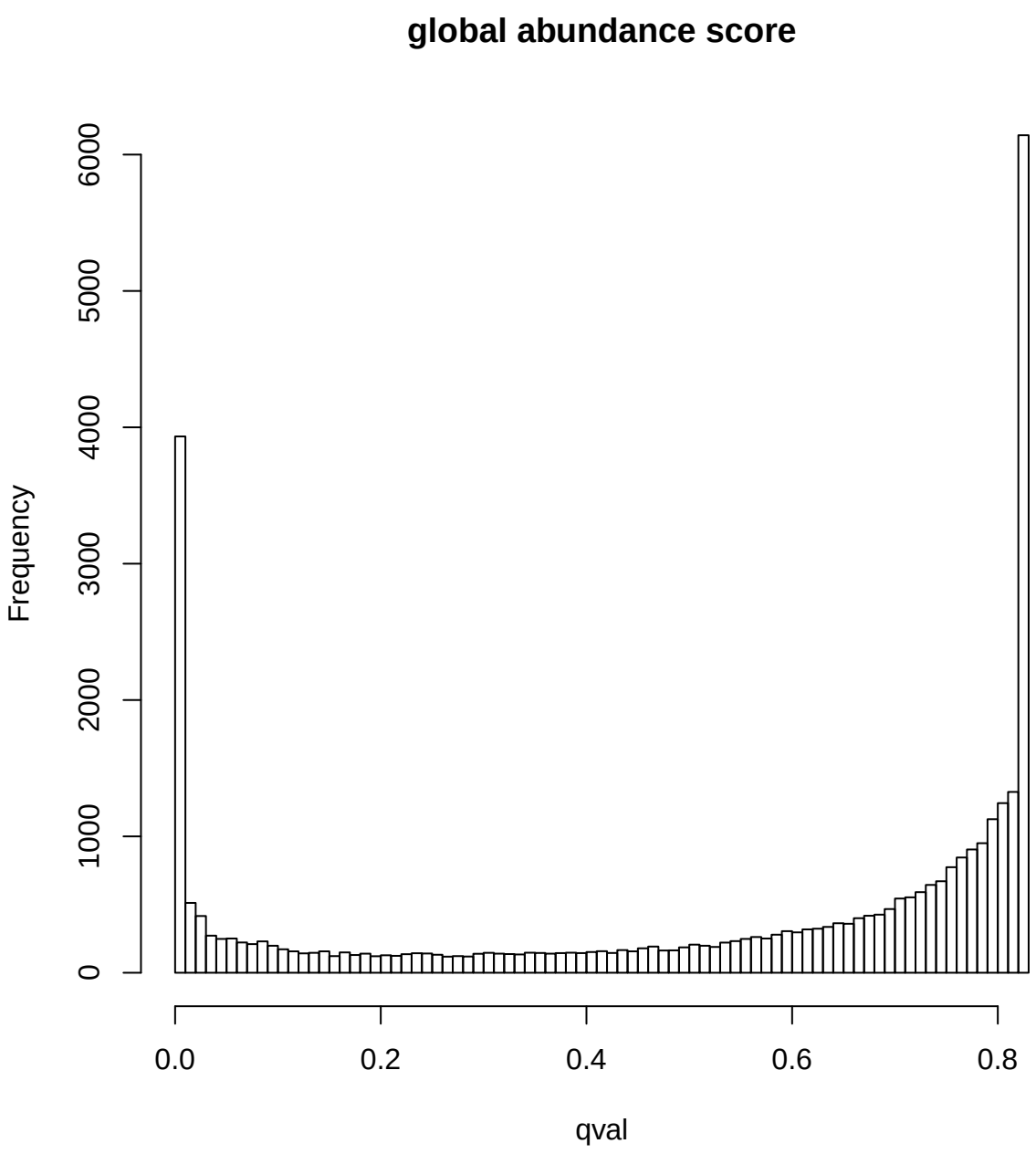
```
## Step 1... determine cutoff point
## Step 2... estimate parameters of null distribution and eta0
## Step 3... compute p-values and estimate empirical PDF/CDF
## Step 4... compute q-values and local fdr
## Step 5... prepare for plotting
```







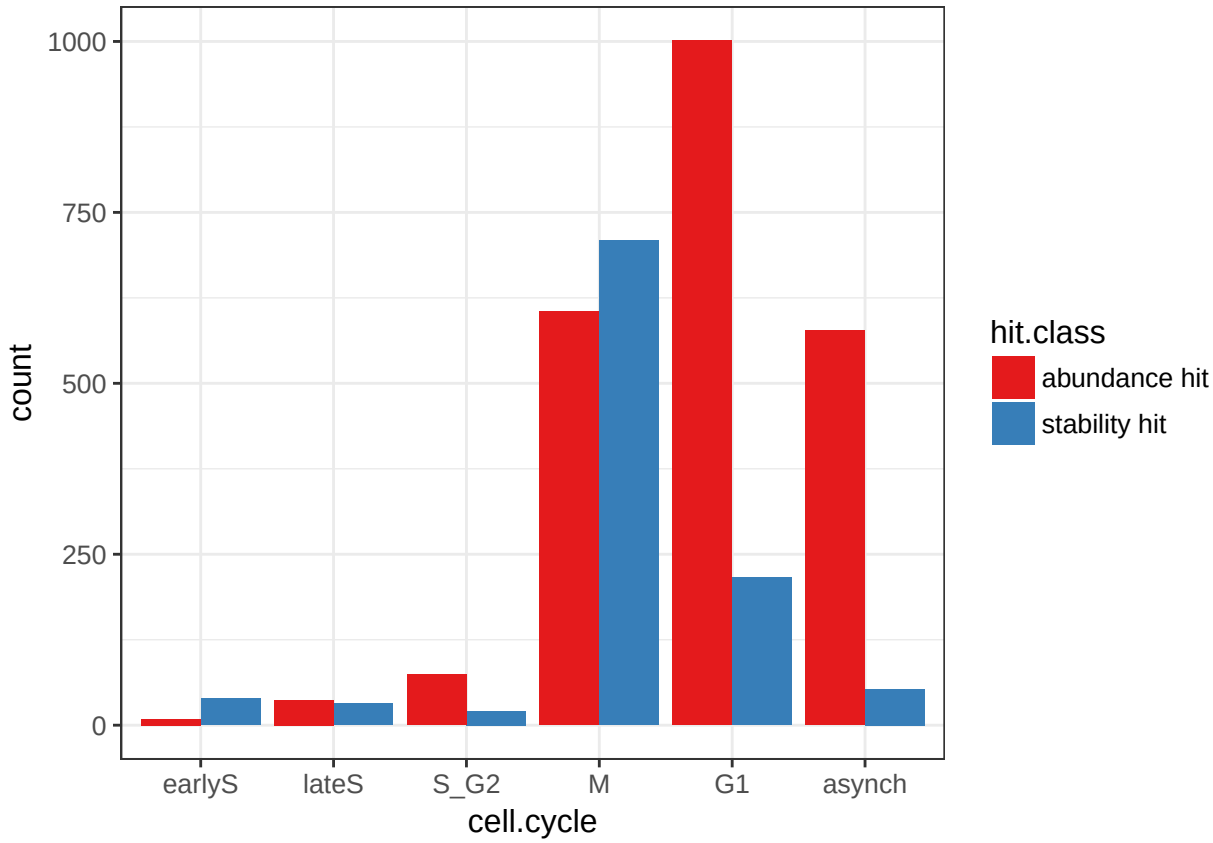

```
hist(score.data$qval.bootstrap.abundance.score.perpop,breaks=100,
     main = "global abundance score",xlab="qval")
```



Assigning hits with a 0.01 cutoff for both local and global fdr

```
score.data$abundance.hit.perpop<-ifelse(
  score.data$qval.bootstrap.abundance.score.perpop<=0.01,T,F)
score.data$abundance.hit.perprot<-ifelse(
  score.data$fdr.bootstrap.abundance.score.perprot<=0.01,T,F)
score.data$abundance.hit<-with(score.data,ifelse(
  abundance.hit.perpop&abundance.hit.perprot,T,F))
```

```
score.data$stability.hit.perpop<-ifelse(
  score.data$qval.bootstrap.stability.score.perpop<=0.01,T,F)
score.data$stability.hit.perprot<-ifelse(
  score.data$fdr.bootstrap.stability.score.perprot<=0.01,T,F)
score.data$stability.hit<-with(score.data,ifelse(
  stability.hit.perpop&stability.hit.perprot,T,F))

score.data$hit.class<-with(score.data,ifelse(
  abundance.hit&stability.hit,"abundance and stability",
  ifelse(abundance.hit&!stability.hit,"abundance hit",
  ifelse(!abundance.hit&stability.hit,"stability hit","no hit"))))
score.data$cell.cycle<-factor(score.data$cell.cycle,ordered=T,
                              levels=c("earlyS","lateS","S_G2","M","G1","asynch","G1_S"))
ggplot(data=subset(score.data,hit.class%in%c("abundance and stability hit",
                                             "abundance hit","stability hit")),
       aes(cell.cycle,fill=hit.class))+geom_bar(position="dodge")+
  scale_fill_brewer(palette="Set1")+theme_bw(base_size=12)
```



Reshaping score data

```
# score.data<-read.csv(file.path("processed_data","score.data_orig_V3.csv"))
mscore.data<-melt(as.data.frame(score.data)[,c("gene_name","cell.cycle",
"bootstrap.abundance.score.mean.z.perpop","bootstrap.stability.score.mean.z.perpop",
"fdr.bootstrap.stability.score.perprot","fdr.bootstrap.abundance.score.perprot",
```

```
"qval.bootstrap.stability.score.perpop","qval.bootstrap.abundance.score.perpop","hit.class")],
                  id.vars = c("gene_name","cell.cycle"))
rm(score.data)
mscore.data$variable<-revalue(mscore.data$variable,
replace = c("bootstrap.abundance.score.mean.z.perpop"="abundance.score",
            "bootstrap.stability.score.mean.z.perpop"="stability.score",
            "qval.bootstrap.stability.score.perpop"="fdr.global.stability.score",
            "qval.bootstrap.abundance.score.perpop"="fdr.global.abundance.score",
            "fdr.bootstrap.stability.score.perprot"="fdr.local.stability.score",
            "fdr.bootstrap.abundance.score.perprot"="fdr.local.abundance.score"))
mscore.data$variable<-factor(mscore.data$variable,ordered=T,
                             levels=c("abundance.score","stability.score","hit.class",
                                      "fdr.global.abundance.score","fdr.global.stability.score",
                                      "fdr.local.abundance.score","fdr.local.stability.score"))
```

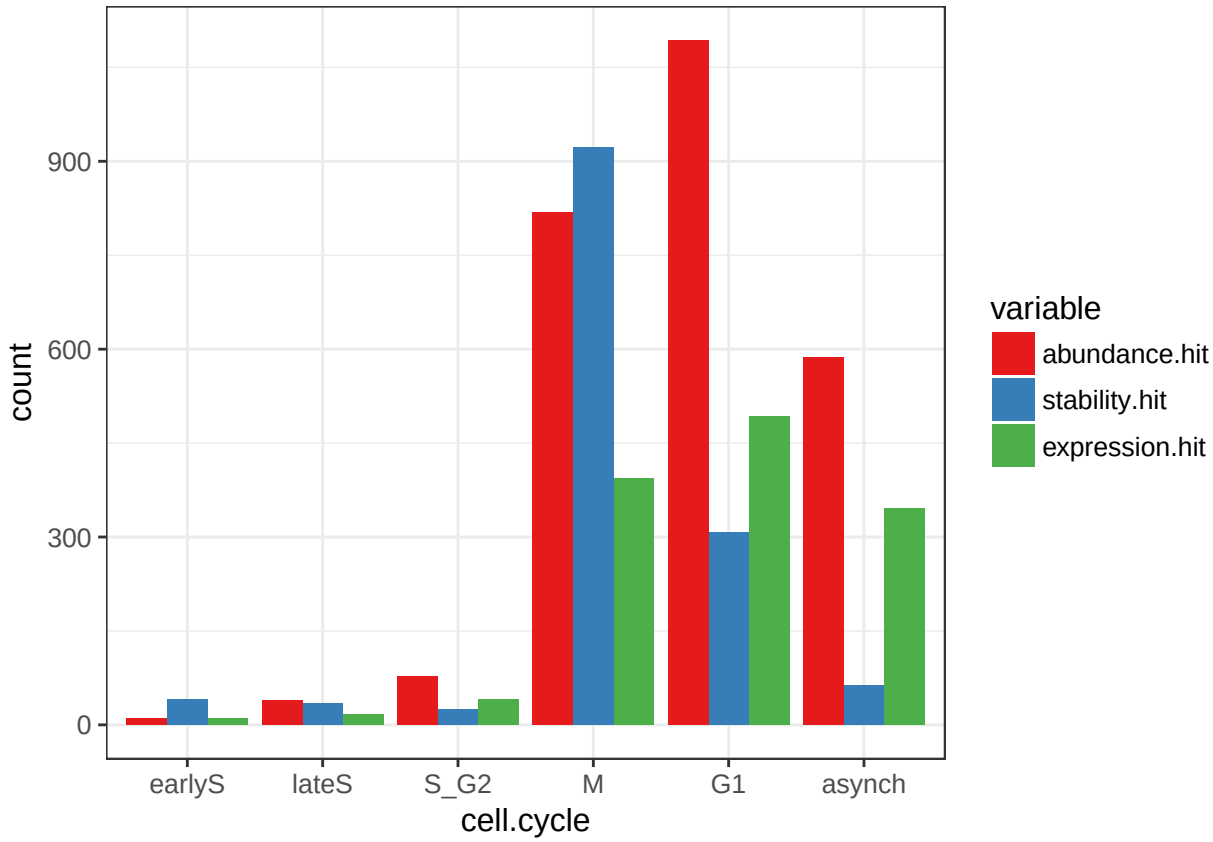
Merging abundance and stability scores with expression scores

```
sds.scores<-read.csv(file.path("processed_data","SDS_expression_score_data_V1.csv"))
sds.scores$min.global.fdr<-NULL
sds.scores$min.local.fdr<-NULL
sds.scores$X<-NULL
sds.scores<-subset(sds.scores,gene_name%in%unique(mscore.data$gene_name))
mscore.data<-subset(mscore.data,variable!="hit.class")
mscore.data$variable<-as.character(mscore.data$variable)
mscore.data$value<-as.numeric(as.character(mscore.data$value))
sds.scores$variable<-as.character(sds.scores$variable)
sds.scores$value<-as.numeric(as.character(sds.scores$value))
mscore.data<-rbind(mscore.data,sds.scores)
rm(sds.scores)
```

Hit classification

```
hit_data<-cast(data = subset(mscore.data,grepl("^fdr.",variable)),
               formula=gene_name+cell.cycle~variable)
hit_data$stability.hit<-with(hit_data,
                             ifelse(fdr.global.stability.score<=0.01&
                                      fdr.local.stability.score<=0.01,T,F))
hit_data$abundance.hit<-with(hit_data,
                             ifelse(fdr.global.abundance.score<=0.01&
                                      fdr.local.abundance.score<=0.01,T,F))
hit_data$expression.hit<-with(hit_data,
                              ifelse(fdr.global.expression.score<=0.01&
                                       fdr.local.expression.score<=0.01,T,F))
hit_data<-hit_data[,c("gene_name","cell.cycle","abundance.hit",
                      "expression.hit","stability.hit")]
hit_data<-melt(data=as.data.frame(hit_data),
               id.vars=c("gene_name","cell.cycle"))
hit_data$cell.cycle<-factor(hit_data$cell.cycle,ordered=T,
                            levels=c("earlyS","lateS","S_G2","M","G1","asynch","G1_S"))
hit_data$variable<-factor(hit_data$variable,ordered=T,
```

```
                          levels=c("abundance.hit","stability.hit","expression.hit"))
ggplot(data=subset(hit_data,value),aes(cell.cycle,fill=variable))+
  geom_bar(position="dodge")+scale_fill_brewer(palette="Set1")+theme_bw(base_size=12)
```




```
mscore.data<-rbind(mscore.data,hit_data)
rm(hit_data)
```

Calculating minimum local and global fdr for each protein

```
sum_score.data1<-ddply(.data = subset(mscore.data,grepl("^fdr.global",variable)),
                       .variables = c("gene_name"),
                       summarise,min.global.fdr=min(value,na.rm=T))
sum_score.data2<-ddply(.data = subset(mscore.data,grepl("^fdr.local",variable)),
                       .variables = c("gene_name"),
                       summarise,min.local.fdr=min(value,na.rm=T))
sum_score.data<-merge(sum_score.data1,sum_score.data2)
rm(sum_score.data1,sum_score.data2)
mscore.data<-merge(mscore.data,sum_score.data)
rm(sum_score.data)
```

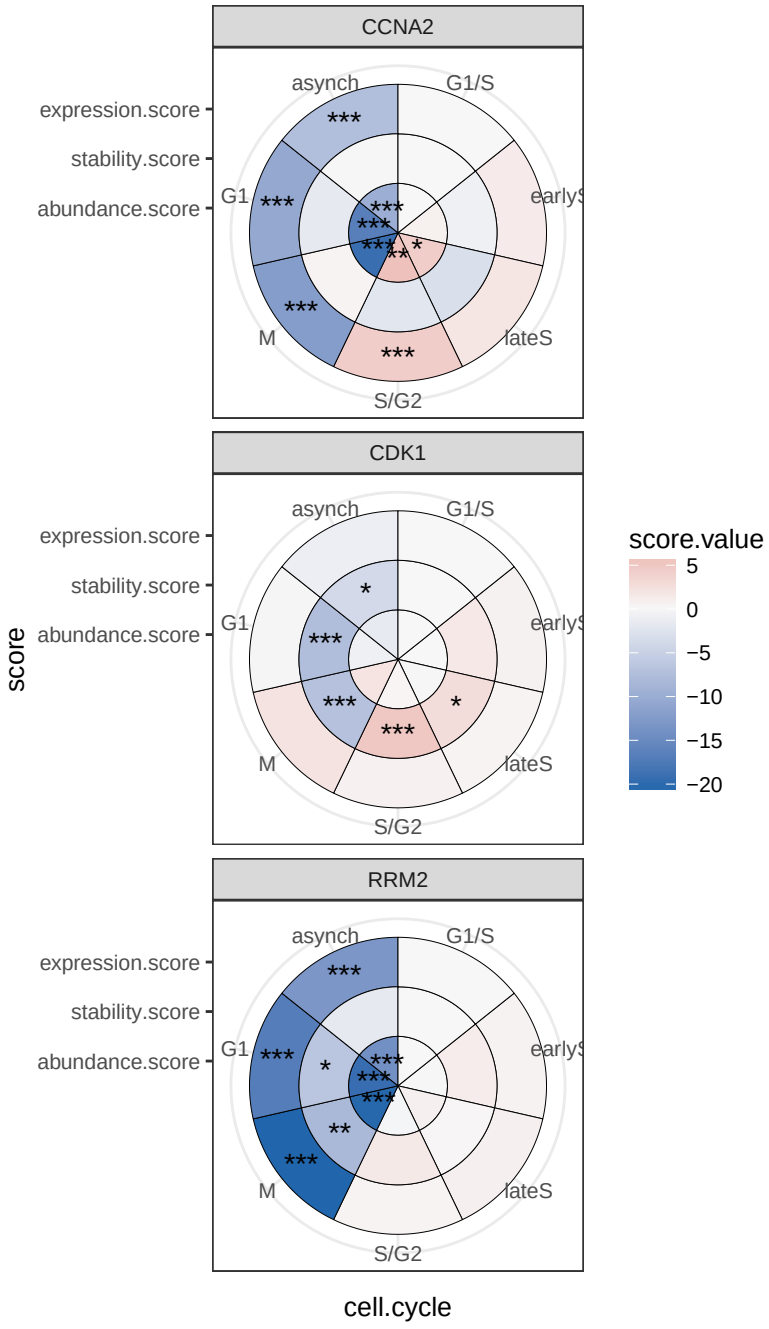
Saving score data

```
mscore.data$cell.cycle<-factor(mscore.data$cell.cycle,ordered=T,
                               levels=c("earlyS","lateS","S_G2","M","G1","asynch","G1_S"))
mscore.data$variable<-as.character(mscore.data$variable)
mscore.data$variable<-factor(mscore.data$variable,ordered=T,
                             levels=c("expression.score","abundance.score",
                                      "stability.score",
                                      "expression.hit","abundance.hit","stability.hit",
                                      "fdr.global.expression.score",
                                      "fdr.global.abundance.score","fdr.global.stability.score",
                                      "fdr.local.expression.score","fdr.local.abundance.score",
                                      "fdr.local.stability.score"))
cscore.data<-cast(mscore.data,gene_name+min.global.fdr+min.local.fdr~variable+cell.cycle)
cscore.data$path<-file.path(".","score_plots",
                            paste0(gsub("\\|","-",cscore.data$gene_name),".pdf"))
cscore.data$gene_name2<-paste0("'",cscore.data$gene_name)

write.csv(cscore.data,file.path("processed_data",
                                "CellCycle_data_scores_classification_V3.csv"),row.names = F)
write.csv(mscore.data,file.path("processed_data",
                                "CellCycle_data_scores_classification_longformat_V3.csv"),
          row.names = F)
```

Reshaping score data for circle plots

```
cp.data<-subset(mscore.data,variable%in%c("abundance.score","expression.score","stability.score",
                                          "fdr.global.abundance.score",
                                          "fdr.global.expression.score",
                                          "fdr.global.stability.score",
                                          "fdr.local.stability.score","fdr.local.abundance.score","fdr.l
cp.data$measurement<-gsub("(.expression.+)|(.abundance.+)|(.stability.+)","",cp.data$variable)
cp.data$measurement<-gsub(".+score","score.value",cp.data$measurement)
cp.data$score<-gsub("(fdr.[a-z]+[.])","",cp.data$variable)
cp.data$X<-NULL
cp.data<-cast(cp.data,formula=gene_name+cell.cycle+score+min.global.fdr+
                min.local.fdr~measurement,value="value")
cp.data$score<-factor(cp.data$score,ordered=T,levels=c("abundance.score",
                                                       "stability.score",
                                                       "expression.score"))
cp.data$score.value<-as.numeric(as.character(cp.data$score.value))
cp.data$cell.cycle<-factor(cp.data$cell.cycle,ordered=T,levels=c("G1_S","earlyS","lateS",
                                                                "S_G2","M","G1","asynch"))
cp.data$cell.cycle<-revalue(cp.data$cell.cycle,replace = c("S_G2"="S/G2","G1_S"="G1/S"))
cp.data$fdr.local.star<-as.character(cut(as.numeric(as.character(cp.data$fdr.local)),
                                         breaks=c(0,0.001,0.01,0.05,1),include.lowest = T))
cp.data$fdr.local.star[is.na(cp.data$fdr.local.star)]<-""
cp.data$fdr.local.star<-revalue(cp.data$fdr.local.star,replace = c("(0.05,1]"="",
                                                                   "(0.01,0.05]"="*",
                                                                   "(0.001,0.01]"="**",
                                                                   "[0,0.001]"="***"))
cp.data$fdr.local.star[!as.numeric(as.character(cp.data$fdr.global))<=0.01&
```

```
                          !is.na(cp.data$fdr.global)]<-""
cp.data.g1s<-cp.data
cp.data.g1s$score<-"expression.score"
cp.data.g1s$cell.cycle<-"G1/S"
cp.data.g1s$fdr.global<-NA
cp.data.g1s$fdr.local<-NA
cp.data.g1s$score.value<-0
cp.data.g1s$fdr.local.star<-""
cp.data.g1s<-unique(cp.data.g1s)
cp.data<-rbind(cp.data,cp.data.g1s)
rm(cp.data.g1s)
ggplot(data=subset(cp.data,gene_name%in%c("CCNA2","CDK1","RRM2")),
       aes(x=cell.cycle,score,fill=score.value))+
  geom_tile(color="black")+coord_polar()+
  scale_fill_gradient2(low="#2166ac",high="#b2182b",midpoint = 0,mid="#f7f7f7")+
  theme_bw(base_size=10)+facet_wrap(~gene_name,ncol=1)+
  geom_text(aes(label=fdr.local.star))
```

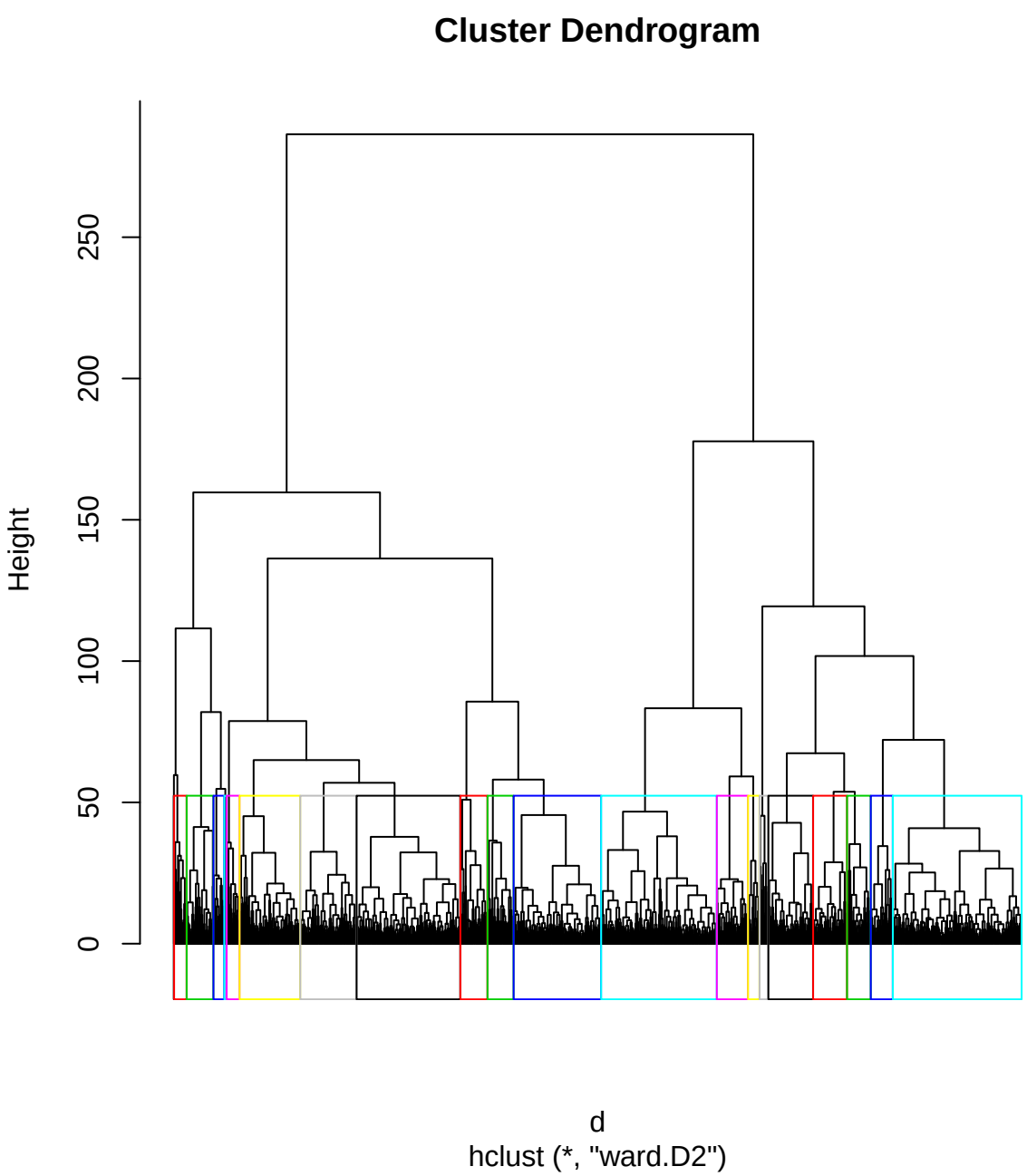



```
write.csv(cp.data,file.path("processed_data","circle-plot_data_V1.csv"),row.names = F)
```

Clustering of score data

```
clust_data<-cscore.data[!is.na(cscore.data$min.global.fdr)&cscore.data$min.global.fdr<=0.01&
                          cscore.data$min.local.fdr<=0.01,c("gene_name",
                                                            grep("^expression.score_",names(cscore.data)
                                                            grep("^abundance.score_",names(cscore.data)
                                                            grep("^stability.score_",names(cscore.data)
```

```
clust_data$abundance.score_G1_S<-NULL
clust_data$stability.score_G1_S<-NULL
clust_data<-na.omit(clust_data)
mclust_data<-clust_data
mclust_data$gene_name<-NULL

cnames<-names(mclust_data)
for(i in cnames){
  mclust_data[,i]<-as.numeric(as.character(mclust_data[,i]))
}
mclust_data<-matrix(as.matrix(mclust_data),nrow = nrow(mclust_data))
colnames(mclust_data)<-cnames
rm(cnames)
row.names(mclust_data)<-clust_data$gene_name
d<-dist(mclust_data, method = "euclidean")
fit <- hclust(d, method="ward.D2")
plot(fit, labels = FALSE, hang = -1)
rect.hclust(fit,k=21,border=1:21)
```

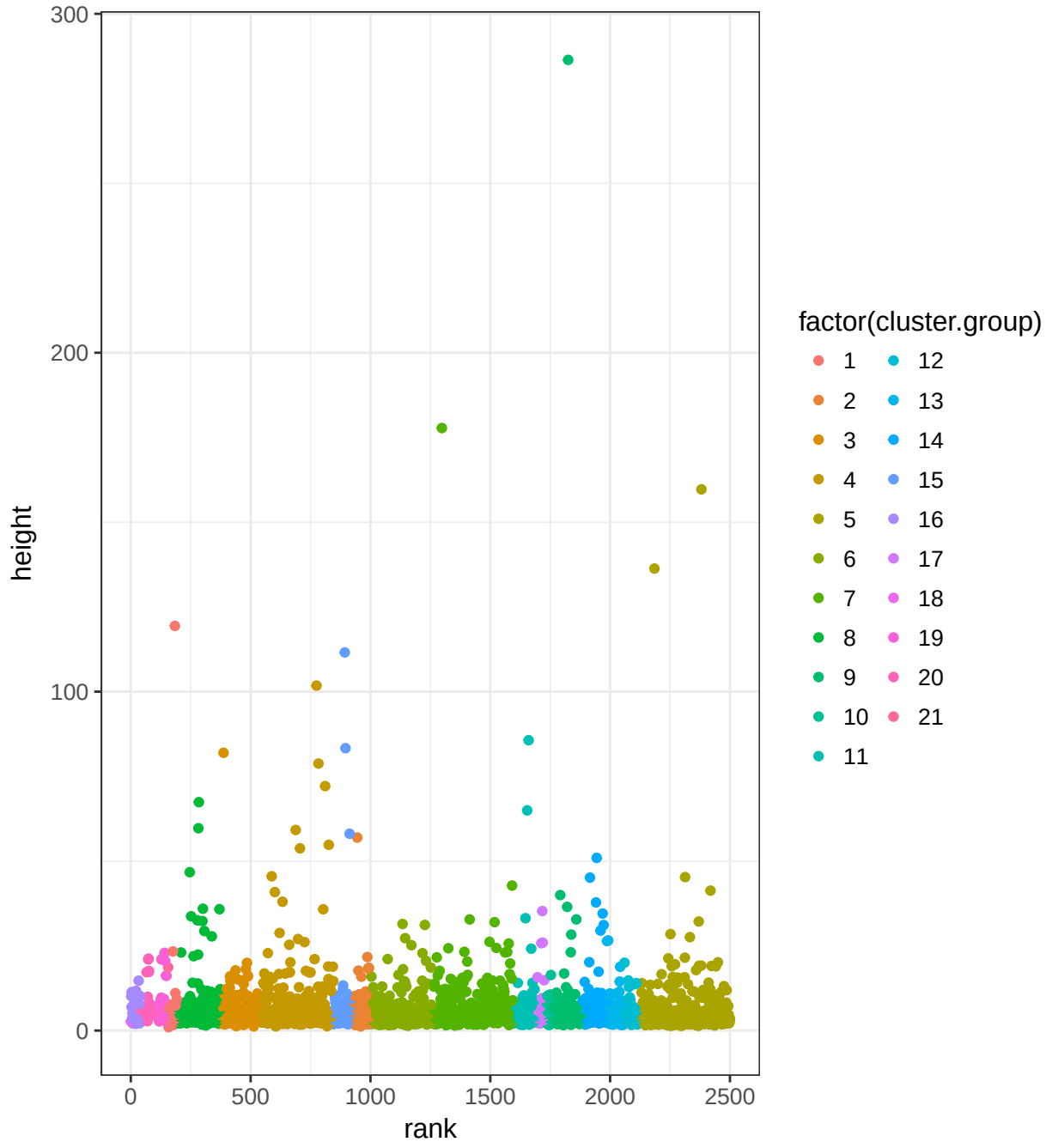



```
cluster_results<-data.frame(gene_name=fit$labels[fit$order],
                            height=c(1,fit$height)[fit$order],rank=1:length(fit$order))
groups <- as.data.frame(cutree(fit, k=21))
names(groups)<-"cluster.group"
groups$gene_name<-rownames(groups)
cluster_results<-merge(cluster_results,groups)
head(cluster_results)
```

```
##   gene_name   height rank cluster.group
## 1       A2M 1.000000  158             1
## 2      AAAS 1.234654  959             2
```

```
## 3      AAMP 1.238967  513             3
## 4      AAR2 1.267047  606             4
## 5      AARS 1.281083  819             4
## 6  AASDHPPT 1.359301  603             4
```

```
qplot(rank,height,data=cluster_results,colour=factor(cluster.group))+
  theme_bw(base_size=12)
```




```
cluster_results<-cluster_results[order(cluster_results$rank),]
write.csv(cluster_results,file.path("processed_data","clustering_results_21clusters_V1.csv"),
          row.names = F)
```

Normalize 1DTPP-TRR data prior to using TPP package

Loading data and defining experimental conditions vector

```
conditions<-read.csv2("metadata_TPP_TRR.csv")
conditions<-subset(conditions,Experiment!="X")
conditions$RefCol<-NULL
conditions$Temperature<-NULL
conditions<-melt(conditions,id.vars = c("Experiment","Compound","rep","Path"),
                 variable_name = "tmt.label")
names(conditions)[grep("value",names(conditions))]<-"Temperature"
names(conditions)[grep("Compound",names(conditions))]<-"cell.cycle"
conditions$tmt.label<-gsub("^X","",conditions$tmt.label)
conditions<-na.omit(conditions)
files<-unique(conditions$Path)
files<-files[file.exists(file.path("TR_data",files))]
data<-NULL
for(i in 1:length(files)){
  file=files[i]
  x<-read.delim(file.path("./TR_data/",file))
  x<-subset(x,!grepl("[K,k][R,r][T,t][0-9]",gene_name))
  x<-subset(x,!grepl("#+",protein_id))
  x$Path<-file
  x<-x[,c("protein_id","description","gene_name","top3","Path","qupm",
          grep("signal_sum",names(x),value = T))]
  data<-rbind(data,x)
  rm(x)
}
data$description<-gsub(" OS.+","",data$description)
data$description<-gsub(", isoform.+","",data$description)
data$description<-gsub(" [(]Fragment.+","",data$description)

sconditions<-unique(conditions[,c("cell.cycle","rep","Path")])
data<-merge(data,sconditions)

rm(files,i,file,sconditions)
names(data)
```

```
##  [1] "Path"            "protein_id"      "description"
##  [4] "gene_name"       "top3"            "qupm"
##  [7] "signal_sum_126"  "signal_sum_127L" "signal_sum_127H"
## [10] "signal_sum_128L" "signal_sum_128H" "signal_sum_129L"
## [13] "signal_sum_129H" "signal_sum_130L" "signal_sum_130H"
## [16] "signal_sum_131L" "cell.cycle"      "rep"
```

Building an expression set

Restructuring data

```
data2<-data
data2$description<-NULL
data2$protein_id<-NULL
```

```
data2<-unique(data2)
mdata<-melt(data2,id.vars = c("gene_name","top3","Path","cell.cycle","rep","qupm"),
            variable_name = "tmt.label")
rm(data2)
mdata$measurement<-gsub("_[0-9]+.+","",mdata$tmt.label)
mdata$tmt.label<-gsub("([a-z,A-Z]+_)+","",mdata$tmt.label)
mdata$tmt.label<-as.character(mdata$tmt.label)
mdata<-merge(mdata,conditions)
mdata$Experiment<-NULL;mdata$Path<-NULL
mdata<-subset(mdata,!is.na(gene_name))
mdata$m.top3<-"m.top3"
mdata$found<-"found.in.reps"
cdata<-cast(mdata,formula=gene_name~measurement+cell.cycle+Temperature+rep+tmt.label,
            value = "value",fun.aggregate = mean,na.rm=T)
cdata2<-cast(subset(mdata,measurement=="signal_sum"),formula=gene_name~m.top3,
             value = "top3",fun.aggregate = mean,na.rm=T)
cdata<-merge(cdata,cdata2)
cdata2<-cast(subset(mdata,measurement=="signal_sum"),formula=gene_name~found,
             value = "top3",fun.aggregate = length)
cdata2$found.in.reps<-cdata2$found.in.reps/10
cdata<-merge(cdata,cdata2)
cdata2<-cast(subset(mdata,measurement=="signal_sum"),formula=gene_name~found+cell.cycle,
             value = "top3",fun.aggregate = length)
cols<-names(cdata2)[names(cdata2)!="gene_name"]
for(i in cols){
  cdata2[,i]<-cdata2[,i]/10
}
rm(i)
cdata<-merge(cdata,cdata2)

mdata$max.qupm<-"max.qupm"
cdata2<-cast(subset(mdata,measurement=="signal_sum"),formula=gene_name~max.qupm,
             value = "qupm",fun.aggregate = max,na.rm=T)
cdata<-merge(cdata,cdata2)
#max.qupm for each cell.cycle
cdata2<-cast(subset(mdata,measurement=="signal_sum"),formula=gene_name~max.qupm+cell.cycle,
             value = "qupm",fun.aggregate = max,na.rm=T)
cdata<-merge(cdata,cdata2)
#min.qupm for each cell.cycle
mdata$max.qupm<-"min.qupm"
cdata2<-cast(subset(mdata,measurement=="signal_sum"),formula=gene_name~max.qupm+cell.cycle,
             value = "qupm",fun.aggregate = min,na.rm=T)
cdata<-merge(cdata,cdata2)

cdata2<-cast(subset(mdata,rep=="rep2"&measurement=="signal_sum"),formula=gene_name~found,
             value = "top3",fun.aggregate = length)
cdata2$found.in.reps<-cdata2$found.in.reps/10
names(cdata2)[2]<-"found.in.rep2"
cdata<-merge(cdata,cdata2,all=T)

cdata2<-cast(subset(mdata,rep=="rep3"&measurement=="signal_sum"),formula=gene_name~found,
             value = "top3",fun.aggregate = length)
cdata2$found.in.reps<-cdata2$found.in.reps/10
```

```
names(cdata2)[2]<-"found.in.rep3"
cdata<-merge(cdata,cdata2,all=T)

cdata2<-cast(subset(mdata,rep=="rep4"&measurement=="signal_sum"),formula=gene_name~found,
             value = "top3",fun.aggregate = length)
cdata2$found.in.reps<-cdata2$found.in.reps/10
names(cdata2)[2]<-"found.in.rep4"
cdata<-merge(cdata,cdata2,all=T)
rm(cdata2)
```

Filtering data

```
dim(cdata)
```

```
## [1] 9004  139
```

```
cdata<-subset(cdata,max.qupm>=2&found.in.reps_asynch>=2&found.in.reps_G1>=2&
                found.in.reps_G1_S>=2&found.in.reps_M>=2)
dim(cdata)
```

```
## [1] 5297  139
```

Constructing assay data

```
raw_data<-cdata[,c("gene_name",grep("^signal_sum_",names(cdata),value=T))]
rownames(raw_data)<-raw_data$gene_name
raw_data$gene_name<-NULL
raw_data<-raw_data[,grep("signal_sum",names(raw_data),value=T)]
names(raw_data)<-gsub("signal_sum_","",names(raw_data))
raw_data<-as.data.frame(raw_data)
```

Constructing metadata

```
metadata<-data.frame(col.name=names(raw_data))
metadata<-add_col_to_df(metadata,data_col = "col.name",col_name = "tmt.label",
                        Vector = c("126","127L","127H","128L","128H","129L","129H",
                                   "130L","130H","131L","131H"))
metadata<-add_col_to_df(metadata,data_col = "col.name",col_name = "Temperature",
                        Vector = c("37", "40.4", "44", "46.9", "49.8", "52.9", "55.5",
                                   "58.6", "62","66.3"))
metadata<-add_col_to_df(metadata,data_col = "col.name",col_name = "rep",
                        Vector = c("rep1","rep2","rep3","rep4"))
metadata<-add_col_to_df(metadata,data_col="col.name",col_name = "cell.cycle",
                        Vector = c("asynch_","G1_","G1_S_","M_"))
metadata$cell.cycle<-gsub("_$","",metadata$cell.cycle)
metadata<-merge(metadata,conditions,sort=F)
metadata$ID<-metadata$col.name
metadata$condition<-with(metadata,paste0(cell.cycle,"_",Temperature))
```

Constructing feature data

```
rownames(metadata)<-metadata$ID
colnames(raw_data)<-metadata$ID
```

Transformation of raw signal_sum data

The log2 is computed from the raw signal intensities. Furthermore, Infinite values are transformed into missing (NA) values.

```
raw_data_m<-log2(as.matrix(raw_data))
raw_data_m[is.infinite((raw_data_m))]<-NA
```

Constructing the Expression set

```
raw_dataE <- ExpressionSet(assayData = raw_data_m,
                           phenoData = AnnotatedDataFrame(metadata))
validObject(raw_dataE)
```

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

```
rm(raw_data_m,metadata,raw_data,mdata)
```

Remove Batcheffects

Batcheffects are removed by fitting a linear model to the data that try to explain the replicates. The batch effect is than subtracted from the data applying the limma package. Batcheffects are removed for each cell.cycle experiment separately.

```
exprs(raw_dataE)[is.na(exprs(raw_dataE))]<-NA
ccs<-unique(raw_dataE$cell.cycle)
batchcleaned_dataM<-NULL
for(cc in ccs){
  im<-raw_dataE[,raw_dataE$cell.cycle==cc]
  exprs(im)<-removeBatchEffect(exprs(im),
                               batch=as.character(pData(im)$rep),
                               design=model.matrix(~as.character(pData(im)$condition)))
  batchcleaned_dataM<-cbind(batchcleaned_dataM,exprs(im))
  rm(im,norm_im)
}
rm(ccs,cc)
```

Reconstruct Expression set

```
metadata<-data.frame(col.name=colnames(batchcleaned_dataM))
metadata<-add_col_to_df(metadata,data_col = "col.name",col_name = "tmt.label",
                        Vector = c("126","127L","127H","128L","128H","129L","129H",
                                   "130L","130H","131L","131H"))
metadata<-add_col_to_df(metadata,data_col = "col.name",col_name = "Temperature",
                        Vector = c("37", "40.4", "44", "46.9", "49.8", "52.9",
```

```
                                    "55.5", "58.6", "62","66.3"))
metadata<-add_col_to_df(metadata,data_col = "col.name",col_name = "rep",
                        Vector = c("rep1","rep2","rep3","rep4"))
metadata<-add_col_to_df(metadata,data_col="col.name",col_name = "cell.cycle",
                        Vector = c("asynch_","G1_","G1_S_","M_"))
metadata$cell.cycle<-gsub("_$","",metadata$cell.cycle)
metadata<-merge(metadata,conditions,sort=F)
metadata$ID<-metadata$col.name
metadata$condition<-with(metadata,paste0(cell.cycle,"_",Temperature))
rownames(metadata)<-metadata$ID
colnames(batchcleaned_dataM)<-metadata$ID
batchcleaned_dataE <- ExpressionSet(assayData = batchcleaned_dataM,
                            phenoData = AnnotatedDataFrame(metadata))
rm(batchcleaned_dataM,metadata,vsn.fit)
```

Normalization

The vsn package from Wolfgang Huber is used to apply a variance stabilization normalization method on the log2 raw data. Since it is a TPP experiment, each temperature is normalized separately. Therefore, the expression set has to be reconstructed again. Normalization is done for each Temperature separately.

```
Ts<-unique(batchcleaned_dataE$Temperature)
norm_dataM<-NULL
for(temp in Ts){
  im<-exprs(batchcleaned_dataE[,batchcleaned_dataE$Temperature==temp])
  vsn.fit<-vsn2(2^im)
  norm_im<-predict(vsn.fit,2^im)
  norm_dataM<-cbind(norm_dataM,norm_im)
  rm(im,norm_im)
}
rm(Ts,temp)
```

Reconstruct Expression set

```
metadata<-data.frame(col.name=colnames(norm_dataM))
metadata<-add_col_to_df(metadata,data_col = "col.name",col_name = "tmt.label",
                        Vector = c("126","127L","127H","128L","128H","129L","129H",
                                   "130L","130H","131L","131H"))
metadata<-add_col_to_df(metadata,data_col = "col.name",col_name = "Temperature",
                        Vector = c("37", "40.4", "44", "46.9", "49.8", "52.9",
                                   "55.5", "58.6", "62","66.3"))
metadata<-add_col_to_df(metadata,data_col = "col.name",col_name = "rep",
                        Vector = c("rep1","rep2","rep3","rep4"))
metadata<-add_col_to_df(metadata,data_col="col.name",col_name = "cell.cycle",
                        Vector = c("asynch_","G1_","G1_S_","M_"))
metadata$cell.cycle<-gsub("_$","",metadata$cell.cycle)
metadata<-merge(metadata,conditions,sort=F)
metadata$ID<-metadata$col.name
metadata$condition<-with(metadata,paste0(cell.cycle,"_",Temperature))
rownames(metadata)<-metadata$ID
colnames(norm_dataM)<-metadata$ID
```

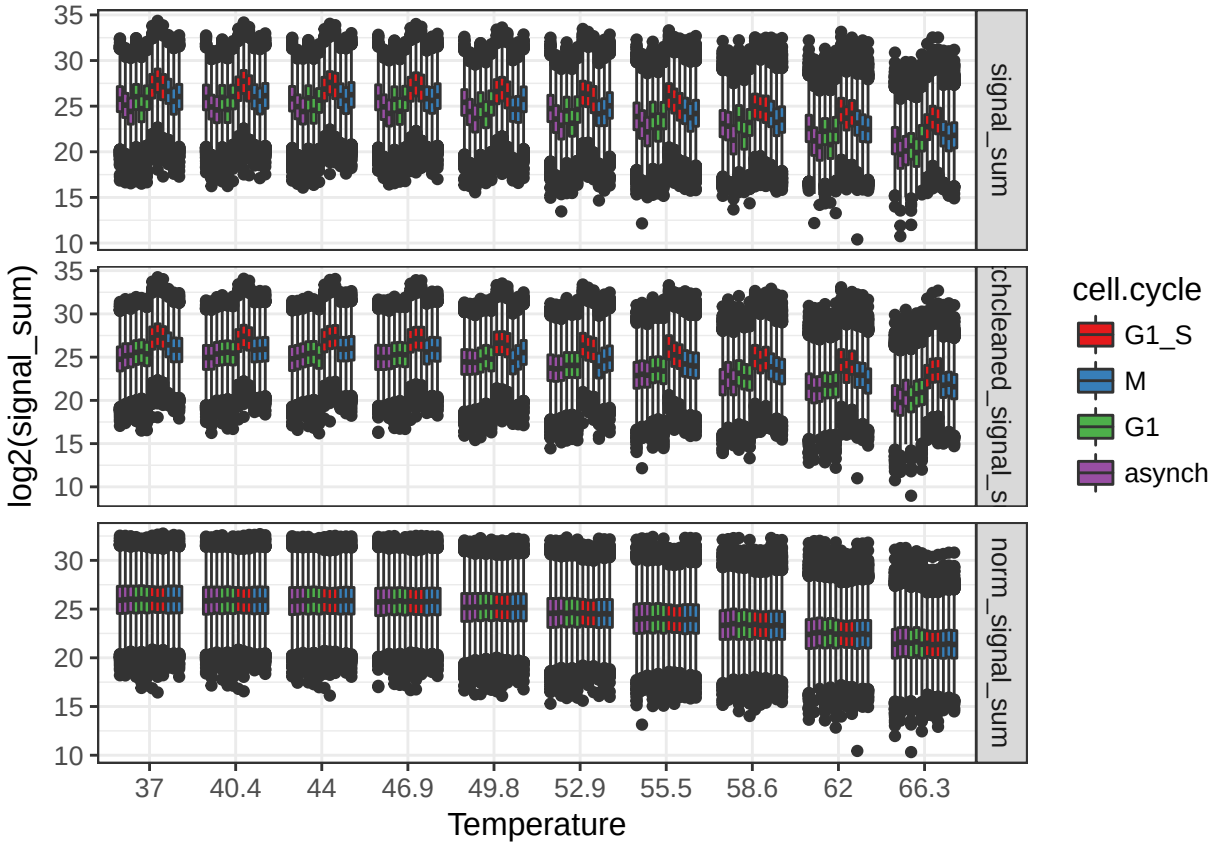
```
norm_batchcl_dataE <- ExpressionSet(assayData = norm_dataM,
                                    phenoData = AnnotatedDataFrame(metadata))
rm(norm_dataM,metadata,vsn.fit)
```

Merge normalized data back into 'data'

```
cd<-as.data.frame(2^exprs(norm_batchcl_dataE))
names(cd)<-paste0("norm_signal_sum_",names(cd))
cd$gene_name<-rownames(cd)
cdata<-merge(cdata,cd)
rm(cd)
cd<-as.data.frame(2^exprs(batchcleaned_dataE))
names(cd)<-paste0("batchcleaned_signal_sum_",names(cd))
cd$gene_name<-rownames(cd)
cdata<-merge(cdata,cd)
rm(cd)
rm(raw_dataE,batchcleaned_dataE,norm_batchcl_dataE)
```

Normalization QC

```
mfc<-melt(cdata[,c("gene_name",grep("^signal_sum",names(cdata),value=T),
                   grep("norm_signal_sum",names(cdata),value=T),
                   grep("batchcleaned_signal_sum",names(cdata),value=T))],
          id.vars = c("gene_name"))
mfc<-na.omit(mfc)
mfc<-add_col_to_df(mfc,col_name = "measurement",data_col = "variable",
                   Vector=c("signal_sum","batchcleaned_signal_sum","norm_signal_sum"))
mfc$variable<-gsub(".+ignal_sum_","",mfc$variable)
mfc$variable<-gsub("_[0-9]+$","",mfc$variable)
mfc$variable<-gsub("_[0-9]+[H,L]$","",mfc$variable)
mfc$rep<-mfc$variable
mfc$variable<-gsub("_rep[0-9]+$","",mfc$variable)
names(mfc)[grep("variable",names(mfc))]<-c("condition")
mfc$experiment<-mfc$condition
mfc$rep<-gsub(".+_","",mfc$rep)

conditions$condition<-with(conditions,paste0(cell.cycle,"_",Temperature))
mfc<-merge(mfc,conditions)
mfc$Temperature<-factor(mfc$Temperature)
mfc$cell.cycle<-factor(mfc$cell.cycle,ordered=T,
                       levels=c("G1_S","earlyS","lateS","S_G2","M","G1","asynch"))

ggplot(data=mfc,aes(Temperature,log2(value),fill=cell.cycle))+
  facet_grid(measurement~.,scale="free")+geom_boxplot(aes(group=paste(condition,rep)))+
  theme_bw(base_size=12)+scale_fill_brewer(palette="Set1")+ylab("log2(signal_sum)")
```




```
ggsave(file.path("QC_plots","TRR_normalization_result.pdf"),width=15,height=15)
rm(mfc)
```

Calcuate new fold changes

```
fc<-cdata[,c("gene_name",grep("norm_signal_sum",names(cdata),value=T),
             grep("max.qupm_",names(cdata),value=T))]
names(fc)[grepl("norm_signal_sum",names(fc))]<-gsub("_[0-9]+.$","",
                                                    names(fc)[grepl("norm_signal_sum",names(fc))])
names(fc)<-gsub("norm_signal_sum_","",names(fc))
cols<-ncol(fc)

temps<-as.character(unique(conditions$Temperature))
cycles<-as.character(na.omit(unique(conditions$cell.cycle)))
reps<-as.character(unique(conditions$rep))
for(cycle in cycles){
  for(temp in temps){
    for(rep in reps){
      name<-paste0(cycle,"_",temp,"_",rep,".37.tempr")
      fc[,name]<-fc[,paste0(cycle,"_",temp,"_",rep)]/fc[,paste0(cycle,"_37","_",rep)]
      fc[,name][fc[,paste0("max.qupm_",cycle)]<2]<-NA
    }
  }
}
fc<-fc[,-c(2:cols)]
```

```
rm(temps,cycles,temp,cycle,rep,reps,name)
cdata<-merge(cdata,fc)
rm(fc)
```

Create modified output tables

```
data<-cdata[,c("gene_name",grep("norm_signal_sum",names(cdata),value=T),
               grep(".tempr$",names(cdata),value=T))]
metadata<-data.frame(col.name=names(data))
metadata<-add_col_to_df(metadata,data_col = "col.name",col_name = "Temperature",
                        Vector = c("37", "40.4", "44", "46.9", "49.8", "52.9",
                                   "55.5", "58.6", "62","66.3"))
metadata<-add_col_to_df(metadata,data_col = "col.name",col_name = "rep",
                        Vector = c("rep1","rep2","rep3","rep4"))
metadata<-add_col_to_df(metadata,data_col="col.name",col_name = "cell.cycle",
                        Vector = c("asynch_","G1_","G1_S_","M_"))
metadata$cell.cycle<-gsub("_$","",metadata$cell.cycle)
metadata<-add_col_to_df(metadata,data_col = "col.name",col_name = "measurement",
                        Vector = c("norm_signal_sum","tempr"))

metadata$condition<-gsub("(^norm_signal_sum_)|(.37.tempr$)","",metadata$col.name)
metadata$condition<-gsub("_rep.+","",metadata$condition)
metadata<-na.omit(metadata)
mmetadata<-melt(metadata,id.vars = c("Temperature","rep","cell.cycle","measurement",
                                     "condition"))
mmetadata<-merge(mmetadata,conditions)
cmetadata<-cast(mmetadata,
                Experiment+cell.cycle+Temperature+rep+condition+tmt.label+Path~measurement)

files<-unique(cmetadata$Path)
files<-files[file.exists(file.path("TR_data",files))]
for(i in 1:length(files)){
  file=files[i]
  x<-read.delim(file.path("./TR_data/",file))
  mmeta<-subset(mmetadata,Path==file)
  cmeta<-subset(cmetadata,Path==file)
  sub<-data[,c("gene_name",as.character(mmeta$value))]
  sub<-na.omit(sub)
  duplicates<-names(which(table(x$gene_name)>1))
  x<-subset(x,!gene_name%in%duplicates)
  sub<-subset(sub,gene_name%in%x$gene_name)
  x<-subset(x,gene_name%in%sub$gene_name)
  columns<-names(x)
  new.data<-data.frame(gene_name=sub$gene_name)
  for(j in 1:nrow(cmeta)){
    new.data[,paste0("signal_sum_",cmeta$tmt.label[j])]<-
      sub[,as.character(cmeta$norm_signal_sum[j])]
    new.data[,paste0("rel_fc_",cmeta$tmt.label[j])]<-
      sub[,as.character(cmeta$tempr[j])]
  }
  x<-x[,-grep("^signal_sum_",names(x))]
  x<-x[,-grep("^rel_fc_",names(x))]
```

```
  x<-merge(x,new.data)
  x<-x[order(x$protein_group_no),columns]
  write.table(x,file = file.path("TR_data",paste0("normalized_",file)),
              quote = F,row.names = F,sep = "\t")
  rm(file,x,mmeta,cmeta,sub,duplicates,columns,new.data,j)
}
rm(files,metadata,mmetadata,cmetadata)
```

TPP-TR post TPP package data reshaping and analysis

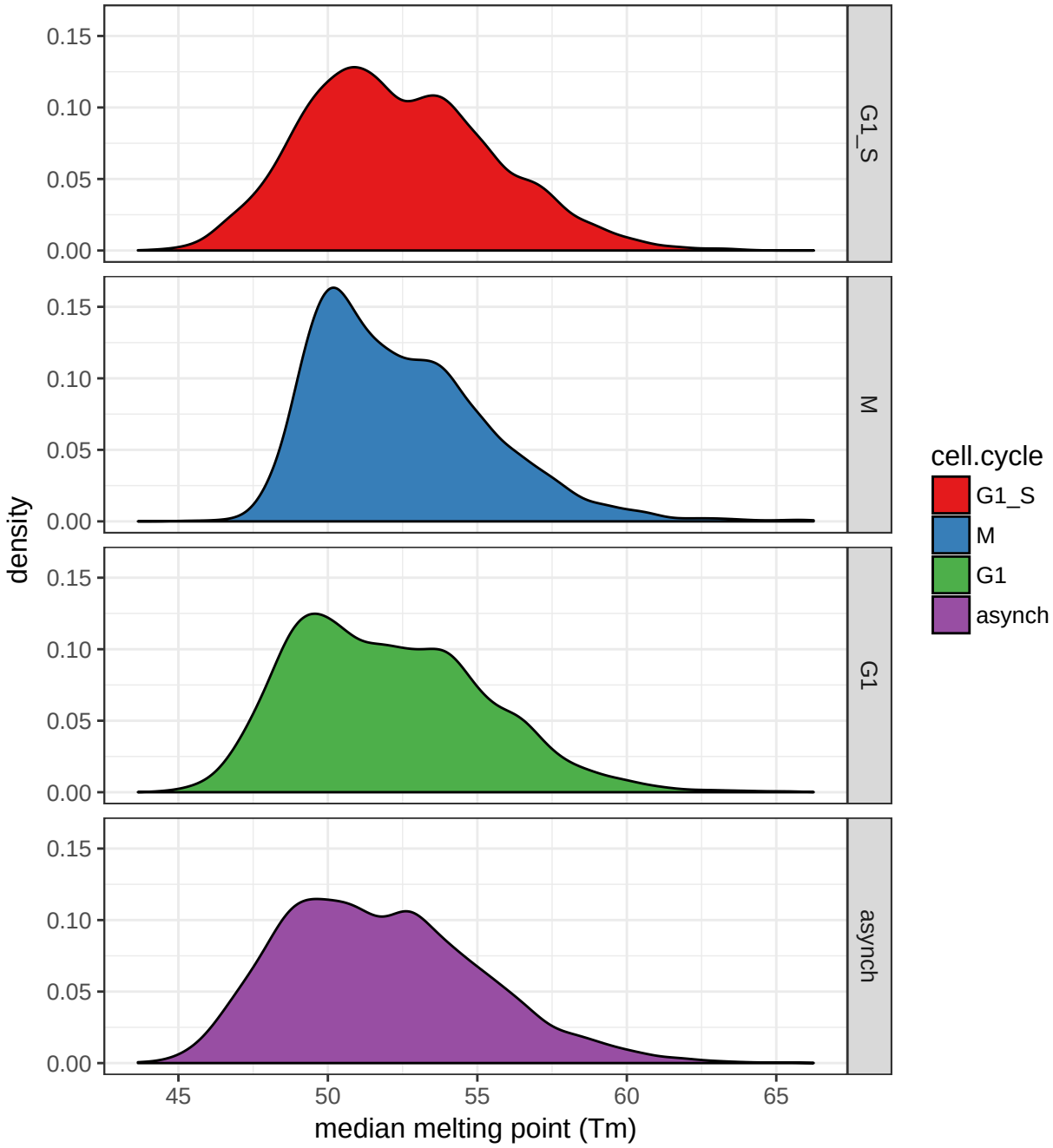
Extract melting points from TPP-TR data analyzed by TPP package

```
TPP.data<-NULL
files<-c("cc_results_TPP_TR_G1.txt","cc_results_TPP_TR_mitosis.txt",
         "cc_results_TPP_TR_refasynch.txt")
for(file in files){
  TPP.sub<-read.delim(file.path("TR_data_post_analysis",file))
  conditions<-gsub("meltPoint_","",grep("^meltPoint",names(TPP.sub),value=T))
  for(cond in conditions){
    cond.sub<-TPP.sub[which(TPP.sub[,paste0("qupm_",cond)]>=2&
                              TPP.sub[,paste0("R_sq_",cond)]>=0.8),c("Protein_ID",
                              paste0("meltPoint_",cond),"p_adj_NPARC")]
    names(cond.sub)<-c("gene_name","Tm","p_adj_NPARC")
    cond.sub$condition<-cond
    cond.sub$file<-file
    TPP.data<-rbind(TPP.data,cond.sub)
    rm(cond.sub)
  }
}
rm(cond)
rm(files,file,TPP.sub)
TPP.data$cell.cycle<-gsub("HeLa_","",TPP.data$condition)
TPP.data$cell.cycle<-gsub("_rep.+","",TPP.data$cell.cycle)
TPP.data$cell.cycle<-gsub("sis[0-9]$","sis",TPP.data$cell.cycle)
TPP.data$cell.cycle<-revalue(TPP.data$cell.cycle,replace = c("G1S"="G1_S","mitosis"="M"))
TPP.data.length<-TPP.data%>%
  group_by(gene_name,cell.cycle)%>%
  summarise(N=length(Tm))
TPP.data<-left_join(TPP.data,TPP.data.length)
rm(TPP.data.length)
TPP.data.sum<-TPP.data%>%
  filter(N>=2)%>%
  group_by(gene_name,cell.cycle)%>%
  summarise(Tm.median=median(Tm,na.rm=T),Tm.range=max(Tm)-min(Tm))
TPP.data<-left_join(TPP.data,TPP.data.sum)
rm(TPP.data.sum)
write.csv(TPP.data,file.path("processed_data","joined_TPP_meltingPoints_full_V1.csv"))

TPP.data$Tm<-NULL;TPP.data$condition<-NULL;TPP.data$p_adj_NPARC<-NULL;TPP.data$file<-NULL
TPP.data<-unique(TPP.data)
write.csv(TPP.data,file.path("processed_data","joined_TPP_median-meltingPoints_long_V1.csv"))
```

```
head(TPP.data)
```

```
##    gene_name cell.cycle N Tm.median  Tm.range
## 1 A0A0C4DFX4       G1_S 9  48.79067 0.6546650
## 2       AAAS       G1_S 9  53.64381 0.6802612
## 3       AACS       G1_S 9  48.46813 0.8663911
## 4       AAK1       G1_S 9  50.86921 0.6066478
## 5      AAMDC       G1_S 9  53.44491 3.2809329
## 6       AAMP       G1_S 9  51.33680 2.0244896
```

```
mTPP.data<-melt(subset(TPP.data),id.vars = c("gene_name","cell.cycle"))
cTPP.data<-cast(mTPP.data,formula = gene_name~variable+cell.cycle)
write.csv(cTPP.data,file.path("processed_data","joined_TPP_median-meltingPoints_wide_V1.csv"))
TPP.data$cell.cycle<-factor(TPP.data$cell.cycle,ordered=T,
                            levels=c("G1_S","earlyS","lateS","S_G2","M","G1","asynch"))
ggplot(data=subset(TPP.data),aes(Tm.median))+geom_density(aes(fill=cell.cycle))+
  facet_grid(cell.cycle~.)+xlab("median melting point (Tm)")+
  scale_fill_brewer(palette="Set1")+theme_bw(base_size=12)
```

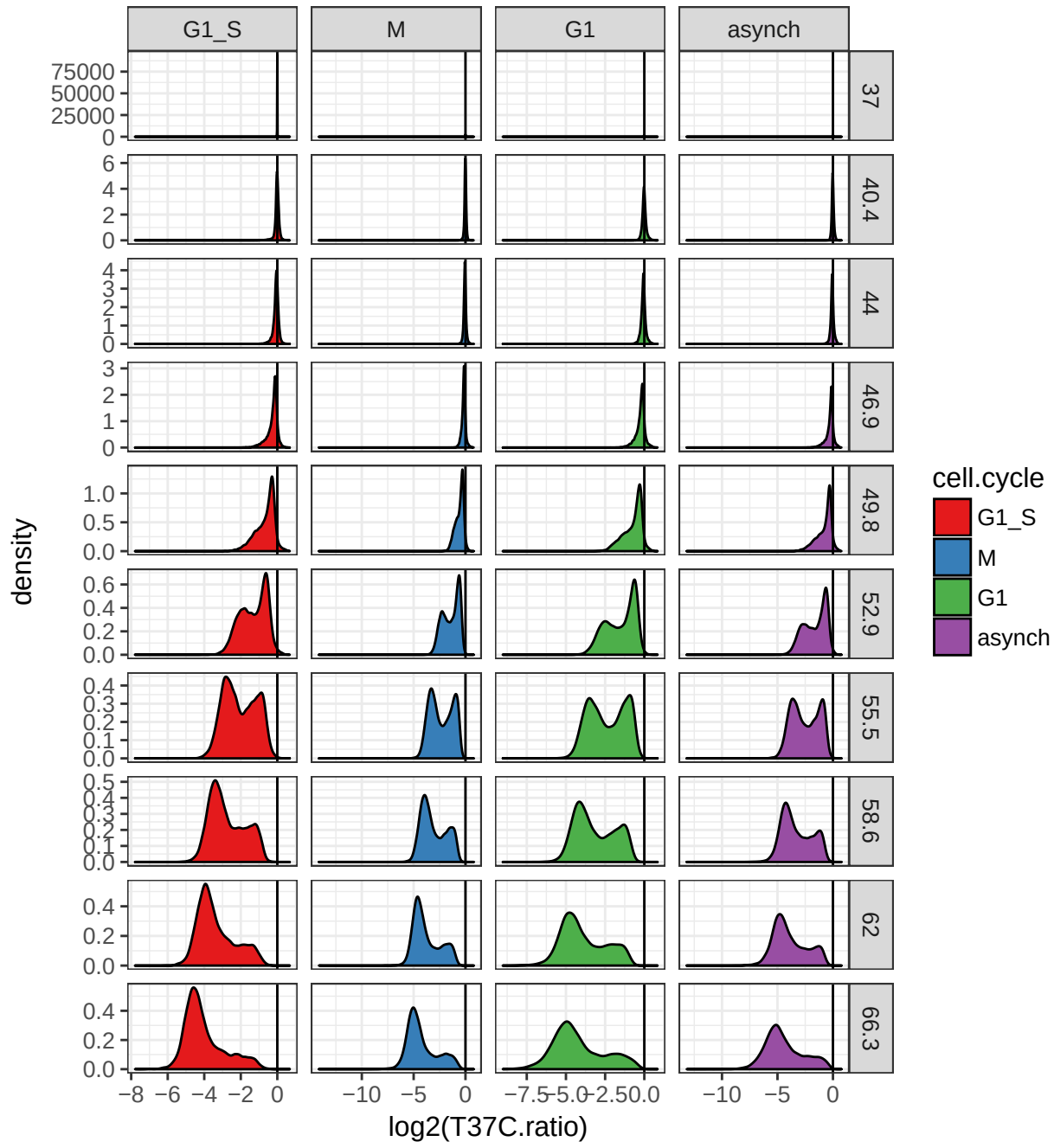



```
ggsave(file.path("QC_plots","TPP_Tm_distributions_vs_cell-cycle_V1.pdf"),width=8,height=10)
```

Extract melting curves from TPP-TR data analyzed by TPP package

```
TPP.data<-NULL
files<-c("cc_results_TPP_TR_G1.txt","cc_results_TPP_TR_mitosis.txt","cc_results_TPP_TR_refasynch.txt")
TPPTR_metadata<-data.frame(tmt.label=c("126","127L","127H","128L","128H",
                                       "129L","129H","130L","130H","131L"),
Temperature=c(37,40.4,44,46.9,49.8,52.9,    55.5,   58.6,   62, 66.3))
```

```
for(file in files){
  TPP.sub<-read.delim(file.path("TR_data_post_analysis",file))
  conditions<-gsub("meltPoint_","",grep("^meltPoint",names(TPP.sub),value=T))
  for(cond in conditions){
    cond.sub<-TPP.sub[which(TPP.sub[,paste0("qupm_",cond)]>=2&
                              TPP.sub[,paste0("R_sq_",cond)]>=0.8),c("Protein_ID",
                              grep(paste0("^norm_rel_fc_.+",cond),names(TPP.sub),value=T))]
    names(cond.sub)[1]<-c("gene_name")
    mcond.sub<-melt(cond.sub,id.vars = c("gene_name"))
    mcond.sub$condition<-cond
    mcond.sub<-add_col_to_df(mcond.sub,data_col = "variable",col_name = "tmt.label",
                             Vector = c("126","127L","127H","128L","128H","129L","129H","130L","130H","
    mcond.sub<-merge(mcond.sub,TPPTR_metadata)
    mcond.sub$file<-file
    TPP.data<-rbind(TPP.data,mcond.sub)
    rm(cond.sub,mcond.sub)
  }
}
rm(cond)
rm(files,file,TPP.sub,TPPTR_metadata)
TPP.data$cell.cycle<-gsub("HeLa_","",TPP.data$condition)
TPP.data$cell.cycle<-gsub("_rep.+","",TPP.data$cell.cycle)
TPP.data$cell.cycle<-gsub("sis[0-9]$","sis",TPP.data$cell.cycle)
TPP.data$cell.cycle<-revalue(TPP.data$cell.cycle,replace = c("G1S"="G1_S",
                                                             "mitosis"="M"))
TPP.data$rep<-paste0("rep",gsub(".+([0-9]$)","\\1",TPP.data$condition))
TPP.data$file2<-as.numeric(as.factor(TPP.data$file))
TPP.data$Temp<-paste0("T",TPP.data$Temperature)

write.csv(TPP.data,file.path("processed_data","joined_TPP_melting_curves_long_V1.csv"))
TPP.data$cell.cycle<-factor(TPP.data$cell.cycle,ordered=T,
                            levels=c("G1_S","earlyS","lateS","S_G2","M","G1","asynch"))

ggplot(data=subset(TPP.data),aes(log2(value)))+geom_density(aes(fill=cell.cycle))+
  facet_grid(Temperature~cell.cycle,scale="free")+xlab("log2(T37C.ratio)")+
  geom_vline(aes(xintercept=0))+scale_fill_brewer(palette="Set1")+theme_bw(base_size=12)
```




```
ggsave(file.path("QC_plots","TPP_TR_ratio_distributions_vs_Temp_V1.pdf"),width=15,height=10)
```

Comparison between NP40 and SDS

Loading and annotating data

```
conditions<-read.csv("metadata_NP40_vs_SDS_Andre.csv")
conditions<-subset(conditions,rep!="")
```

```
head(conditions)
```

```
##   tmt.label  rep condition cell.cycle lysis
## 1       126 rep3  SDS_G1_S       G1_S   SDS
## 2      127L rep4  SDS_G1_S       G1_S   SDS
## 3      127H rep3 NP40_G1_S       G1_S  NP40
## 4      128L rep4 NP40_G1_S       G1_S  NP40
## 5      128H rep3     SDS_M          M   SDS
## 6      129L rep4     SDS_M          M   SDS
##                                           Path
## 1 rep1_merged_results_20171210_1355_proteins.txt
## 2 rep1_merged_results_20171210_1355_proteins.txt
## 3 rep1_merged_results_20171210_1355_proteins.txt
## 4 rep1_merged_results_20171210_1355_proteins.txt
## 5 rep1_merged_results_20171210_1355_proteins.txt
## 6 rep1_merged_results_20171210_1355_proteins.txt
```

```
files<-unique(conditions$Path)
files<-files[file.exists(file.path("SDS_NP40",files))]
data<-NULL
for(i in 1:length(files)){
  file=files[i]
  x<-read.delim(file.path("./SDS_NP40/",file))
  x<-subset(x,!grepl("[K,k][R,r][T,t][0-9]",gene_name))
  x<-subset(x,!grepl("#+",protein_id))
  x<-subset(x,qupm>=2)
  x$Path<-file
  x<-x[,c("protein_id","description","gene_name","top3","Path","qupm",
          grep("signal_sum",names(x),value = T))]
  data<-rbind(data,x)
  rm(x)
}

#Remove duplicates
data$gene_name=as.character(data$gene_name)
dups<-c()
for(filen in files){
  sub<-subset(data,Path==filen)
  dups<-c(dups,names(which(table(sub$gene_name)>1)))
}
dups<-unique(dups)
for(dup in dups){
  sub<-subset(data,gene_name%in%dup)
  no=1
  for(ID in unique(sub$protein_id)){
    if(no>1){
      data$gene_name[data$protein_id==ID&data$gene_name==dup]<-paste0(dup,"_",no)
    }
    no=no+1
  }
}
rm(files,file,sub,filen,no,ID,dups,dup)
names(data)
```

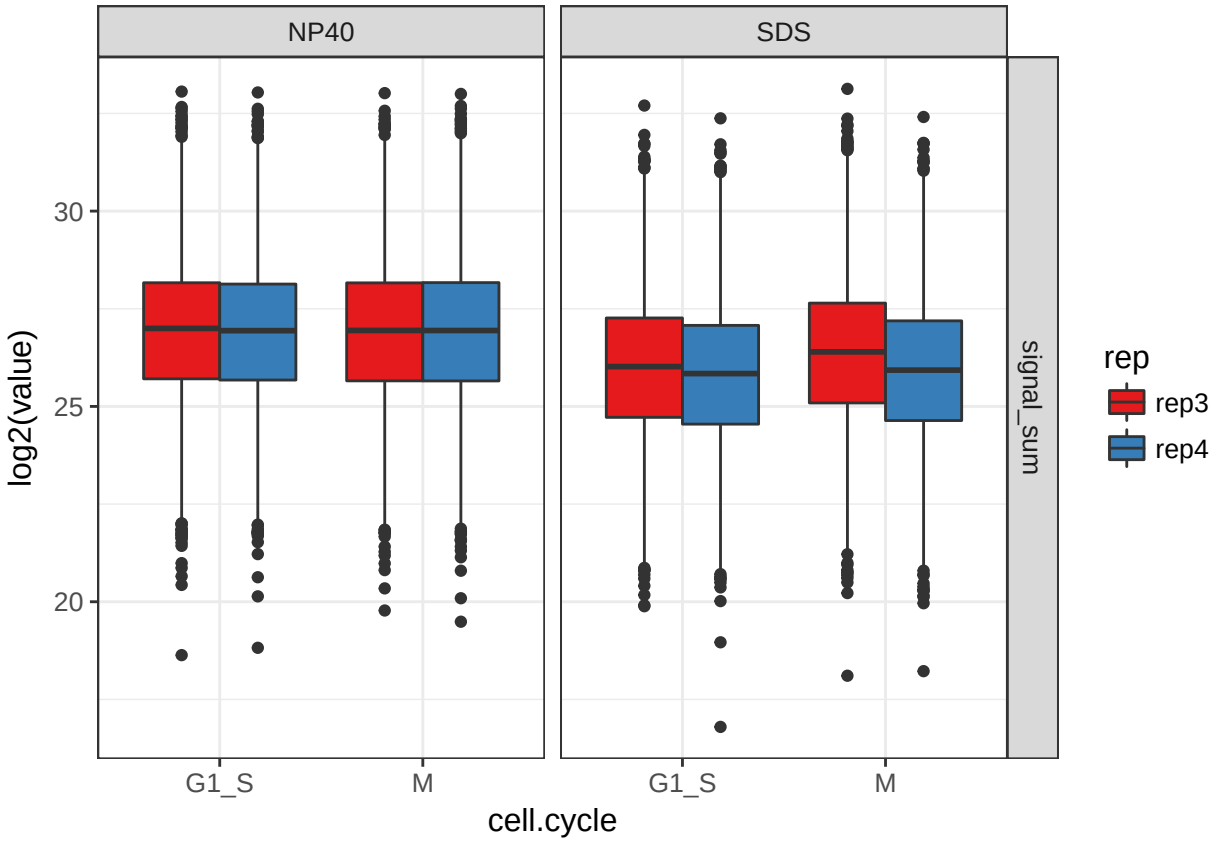
```
##  [1] "protein_id"      "description"     "gene_name"
```

```
##  [4] "top3"            "Path"            "qupm"
##  [7] "signal_sum_126"  "signal_sum_127L" "signal_sum_127H"
## [10] "signal_sum_128L" "signal_sum_128H" "signal_sum_129L"
## [13] "signal_sum_129H" "signal_sum_130L" "signal_sum_130H"
## [16] "signal_sum_131L"
```

Building an expression set

Restructuring data

```
data2<-data
data2$description<-NULL
data2$protein_id<-NULL
data2<-unique(data2)
mdata<-melt(data2,id.vars = c("gene_name","top3","Path","qupm"),
            variable_name = "tmt.label")
rm(data2)
mdata$measurement<-gsub("_[0-9]+.+","",mdata$tmt.label)
mdata$tmt.label<-gsub("([a-z,A-Z]+_)+","",mdata$tmt.label)
mdata$tmt.label<-as.character(mdata$tmt.label)
mdata<-merge(mdata,conditions)
mdata$Path<-NULL
mdata<-subset(mdata,!is.na(gene_name))
mdata$av.top3<-"av.top3"
mdata$found<-"found.in.reps"

ggplot(data=mdata,aes(cell.cycle,log2(value),fill=rep))+geom_boxplot()+facet_grid(measurement~lysis)+
  theme_bw(base_size=12)+scale_fill_brewer(palette="Set1")
```




```
cdata<-cast(mdata,formula=gene_name+top3+qupm~measurement+condition+rep+tmt.label,
            value = "value",fun.aggregate = mean,na.rm=T)
```

Constructing assay data

```
raw_data<-cdata
rownames(raw_data)<-raw_data$gene_name
raw_data$gene_name<-NULL
raw_data<-raw_data[,grep("signal_sum",names(raw_data),value=T)]
names(raw_data)<-gsub("signal_sum_","",names(raw_data))
raw_data<-as.data.frame(raw_data)
```

Constructing metadata

```
metadata<-data.frame(col.name=names(raw_data))
metadata<-add_col_to_df(metadata,data_col = "col.name",col_name = "tmt.label",
                        Vector = c("126","127L","127H","128L","128H","129L","129H",
                                   "130L","130H","131L","131H"))
metadata<-add_col_to_df(metadata,data_col="col.name",col_name="lysis",
                        Vector=c("SDS","NP40"))
metadata<-add_col_to_df(metadata,data_col = "col.name",col_name = "rep",
                        Vector = c("rep1","rep2","rep3","rep4"))
metadata<-merge(metadata,conditions,sort=F)
```

```
metadata$ID<-metadata$col.name
head(metadata)
```

```
##   tmt.label lysis  rep            col.name condition cell.cycle
## 1      127H  NP40 rep3 NP40_G1_S_rep3_127H NP40_G1_S       G1_S
## 2      128L  NP40 rep4 NP40_G1_S_rep4_128L NP40_G1_S       G1_S
## 3      129H  NP40 rep3    NP40_M_rep3_129H    NP40_M          M
## 4      130L  NP40 rep4    NP40_M_rep4_130L    NP40_M          M
## 5       126   SDS rep3   SDS_G1_S_rep3_126  SDS_G1_S       G1_S
## 6      127L   SDS rep4  SDS_G1_S_rep4_127L  SDS_G1_S       G1_S
##                                          Path                  ID
## 1 rep1_merged_results_20171210_1355_proteins.txt NP40_G1_S_rep3_127H
## 2 rep1_merged_results_20171210_1355_proteins.txt NP40_G1_S_rep4_128L
## 3 rep1_merged_results_20171210_1355_proteins.txt    NP40_M_rep3_129H
## 4 rep1_merged_results_20171210_1355_proteins.txt    NP40_M_rep4_130L
## 5 rep1_merged_results_20171210_1355_proteins.txt   SDS_G1_S_rep3_126
## 6 rep1_merged_results_20171210_1355_proteins.txt  SDS_G1_S_rep4_127L
```

Constructing feature data

```
feat_data<-as.data.frame(cdata[,c("gene_name",grep("top3",names(cdata),value=T),
                                  grep("qupm",names(cdata),value=T))])
rownames(feat_data)<-feat_data$gene_name
rownames(metadata)<-metadata$ID
colnames(raw_data)<-metadata$ID
```

Transformation of raw signal_sum data

The log2 is computed from the raw signal intensities. Furthermore, Infinite values are transformed into missing (NA) values.

```
raw_data_m<-log2(as.matrix(raw_data))
raw_data_m[is.infinite((raw_data_m))]<-NA
raw_data_m[is.nan((raw_data_m))]<-NA
```

Constructing the Expression set

```
raw_dataE <- ExpressionSet(assayData = raw_data_m,
                           phenoData = AnnotatedDataFrame(metadata),
                           featureData =AnnotatedDataFrame(feat_data))
validObject(raw_dataE)
```

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

```
rm(raw_data_m,metadata,raw_data,mdata)
```

Batch effect removal

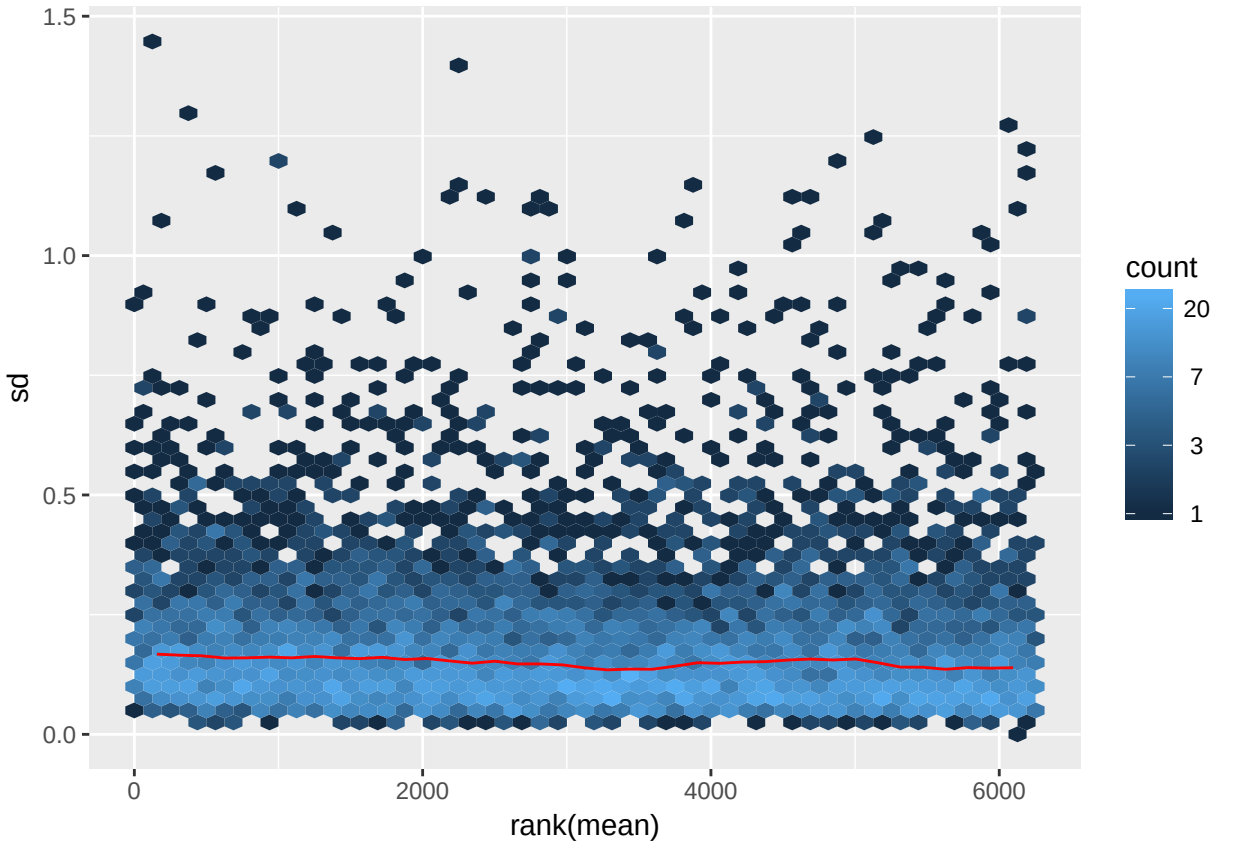
```
library(a4Base)
batchcl_raw_dataE<-raw_dataE
for(samplename in unique(pData(batchcl_raw_dataE)$lysis)){
  print(samplename)
  test<-batchcl_raw_dataE[,batchcl_raw_dataE$lysis==samplename]
  batchcl_raw_dataE<-batchcl_raw_dataE[,batchcl_raw_dataE$lysis!=samplename]
  exprs(test)<-removeBatchEffect(exprs(test),
                                 batch=as.character(pData(test)$rep),
                                 design=model.matrix(~as.character(pData(test)$condition)))
  batchcl_raw_dataE<-combineTwoExpressionSet(batchcl_raw_dataE,test)
}
```

```
## [1] "NP40"
## [1] "SDS"
```

Normalization

The vsn package from Wolfgang Huber is used to apply a variance stabilization normalization method on the log2 raw data.

```
norm_batchcl_raw_dataE<-batchcl_raw_dataE
vsn.fit<-vsn2(2^exprs(norm_batchcl_raw_dataE))
meanSdPlot(vsn.fit)
```




```
exprs(norm_batchcl_raw_dataE)<-predict(vsn.fit,2^exprs(norm_batchcl_raw_dataE))
```

Merge normalized data back into 'data'

```
cd<-as.data.frame(2^exprs(batchcl_raw_dataE))
names(cd)<-paste0("batchcleaned_signal_sum_",names(cd))
cd$gene_name<-rownames(cd)
cdata<-merge(cdata,cd)
rm(cd)
cd<-as.data.frame(2^exprs(norm_batchcl_raw_dataE))
names(cd)<-paste0("norm_signal_sum_",names(cd))
cd$gene_name<-rownames(cd)
cdata<-merge(cdata,cd)
rm(cd)

write.csv(cdata,file=file.path("SDS_vs_NP40_V2","Full_dataset_V1.csv"))

norm_batchcl_raw_dataE <- ExpressionSet(assayData = exprs(norm_batchcl_raw_dataE),
                           phenoData = AnnotatedDataFrame(pData(norm_batchcl_raw_dataE)),
                           featureData =AnnotatedDataFrame(fData(norm_batchcl_raw_dataE)))
```

Calculation of fold changes

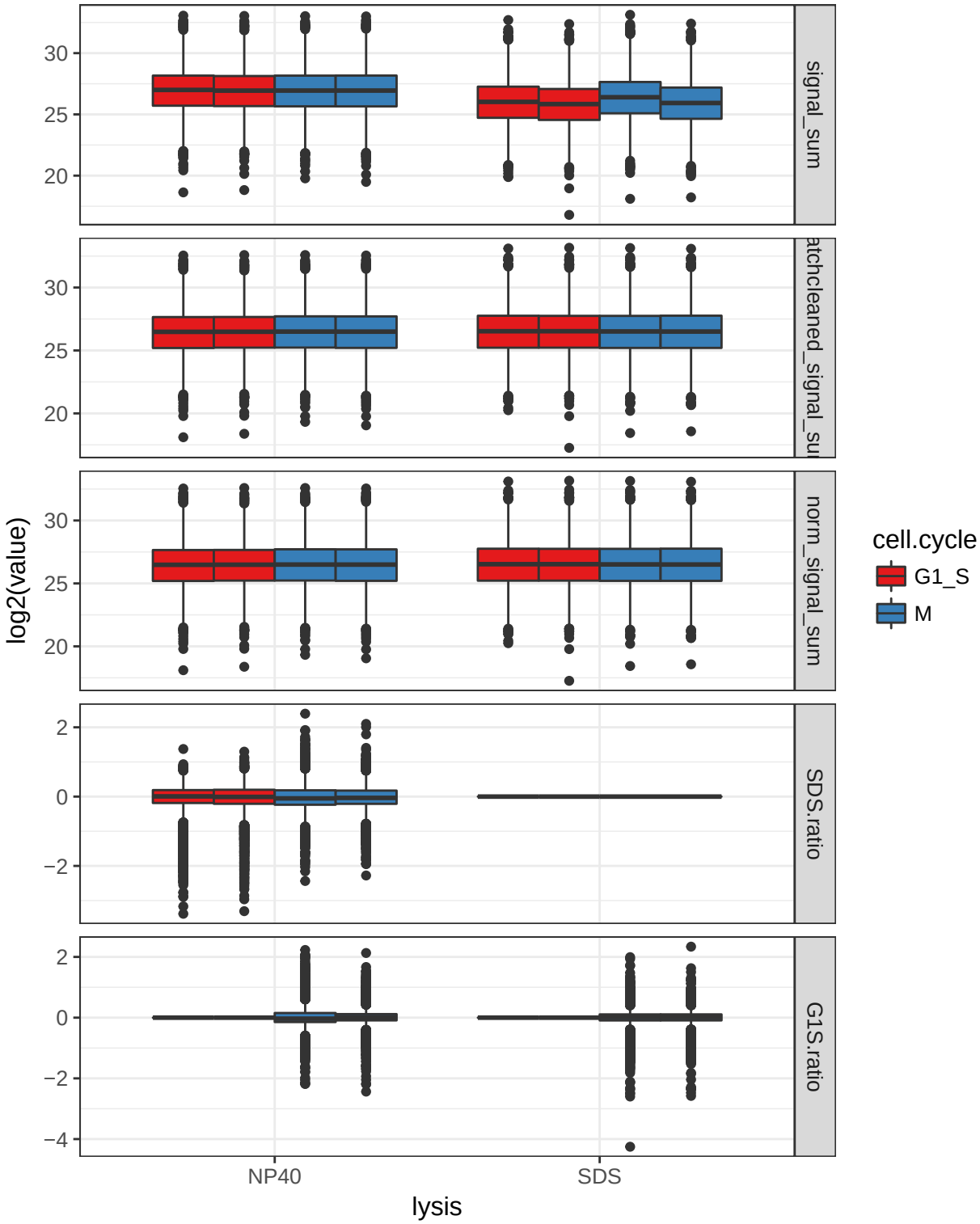
```
fc<-cdata[,c("gene_name","top3",grep("^norm_",names(cdata),value=T))]
names(fc)<-gsub("_[0-9]+.$","",names(fc))
names(fc)<-gsub("norm_signal_sum_","",names(fc))
ncols<-ncol(fc)

cc<-c("G1_S","M")
lysis<-c("NP40","SDS")
reps<-unique(conditions$rep)

for(cycle in cc){
  for(lys in lysis){
    for(rep in reps){
      name<-paste0(lys,"_",cycle,"_",rep,"_G1S.ratio")
      try(fc[,name]<-fc[,paste0(lys,"_",cycle,"_",rep)]/fc[,paste0(lys,"_G1_S_",rep)])
    }
  }
}
for(cycle in cc){
  for(lys in lysis){
    for(rep in reps){
      name<-paste0(lys,"_",cycle,"_",rep,"_SDS.ratio")
      try(fc[,name]<-fc[,paste0(lys,"_",cycle,"_",rep)]/fc[,paste0("SDS_",cycle,"_",rep)])
    }
  }
}
rm(cycle,lys,rep,cc,lysis,reps)

fc<-fc[,-c(3:ncols)]

cdata<-merge(cdata,fc)
write.csv(cdata,file=file.path("SDS_vs_NP40_V2","Full_dataset_V1.csv"))
```

Reshaping full data set

```
mfc<-melt(cdata[,c("gene_name",grep("top3",names(cdata),value=T),
                   grep("qupm",names(cdata),value=T),
                   grep("^signal_sum",names(cdata),value=T),
                   grep("^batchcleaned_signal_sum_",names(cdata),value=T),
                   grep("^norm_signal_sum",names(cdata),value=T),
                   grep("G1S.ratio",names(cdata),value=T),
                   grep("SDS.ratio",names(cdata),value=T)
                   )],
          id.vars = c("gene_name",grep("top3",names(cdata),value=T),
                   grep("qupm",names(cdata),value=T)))
mfc<-na.omit(mfc)
mfc<-add_col_to_df(mfc,col_name = "measurement",data_col = "variable",
                   Vector=c("signal_sum","batchcleaned_signal_sum","norm_signal_sum",
                            "SDS.ratio","G1S.ratio"))
mfc$variable<-gsub(".+ignal_sum_","",mfc$variable)
mfc$variable<-gsub("_SDS.ratio","",mfc$variable)
mfc$variable<-gsub("_G1S.ratio","",mfc$variable)
mfc$variable<-gsub("_[0-9]+$","",mfc$variable)
mfc$variable<-gsub("_[0-9]+[H,L]$","",mfc$variable)
mfc$rep<-mfc$variable
mfc$variable<-gsub("_rep[0-9]+$","",mfc$variable)
names(mfc)[grep("variable",names(mfc))]<-c("condition")
mfc$experiment<-mfc$condition
mfc$rep<-gsub(".+_","",mfc$rep)
mfc$lysis<-gsub("_.+","",mfc$condition)
mfc$cell.cycle<-gsub("(^NP40_)|(^SDS_)","",mfc$condition)
mfc$cell.cycle<-factor(mfc$cell.cycle,ordered=T,levels=c("G1_S","earlyS","lateS",
                                                         "S_G2","M","G1","asynch"))
```

Quality control of normalization

```
ggplot(data=mfc,
       aes(lysis,log2(value),fill=cell.cycle))+facet_grid(measurement~.,scale="free")+
  geom_boxplot(aes(group=paste(cell.cycle,lysis,rep)))+theme_bw(base_size=12)+
  scale_fill_brewer(palette="Set1")
```




```
ggsave(file.path("QC_plots","NP40_and_SDS_Andre_normalization_results_V1.pdf"),width=5,height=15)
```

LIMMA analysis

```
limma_results<-NULL
```

Construction of design matrix and defining conditions that will be compared

```
limma_data=norm_batchcl_raw_dataE
condition<-factor(pData(limma_data)$condition,ordered=F)
replicate<-factor(pData(limma_data)$rep,ordered=F)
condition_d<-model.matrix(~0+condition+replicate)
colnames(condition_d)<-gsub("condition","",colnames(condition_d))

to_condition<-c("NP40_G1_S - SDS_G1_S","NP40_M - SDS_M","(NP40_M - SDS_M) - (NP40_G1_S - SDS_G1_S)")
```

Fitting the model

```
mut_comparison<-eBayes(contrasts.fit(lmFit(limma_data,design=condition_d),
                                     makeContrasts(contrasts = to_condition,levels=condition_d)))
```

Identification of top hits

```
for(comp in to_condition){
  res <- limma::topTable(mut_comparison, sort.by = "t",  coef = comp, number = Inf)
  names(res)[grep("P.Value",names(res))]<-"pvalue.limma"
  names(res)[grep("adj.P.Val",names(res))]<-"fdr.limma"
  res<-na.omit(res)
  res$comparison <- comp
  res$ratio<-ifelse(grepl("[(]",comp),"G1_S.ratio","lysis.ratio")
  limma_results<-rbind(limma_results,res)
}
```

Hit annotation

```
limma_results$comparison<-factor(limma_results$comparison,ordered=T,levels=to_condition)
limma_results$hit_annotation_method<-NA
fdr_hit_threshold<-0.01
fdr_candidate_threshold=0.05
fc_hit_threshold<-1.5
fc_candidate_threshold<-1.25
limma_results$pvalue<-NA
limma_results$fdr<-NA
limma_results$hit_annotation_method<-"limma"

limma_results$pvalue <-limma_results$pvalue.limma
limma_results$fdr <-limma_results$fdr.limma

limma_results$hit<-with(limma_results,ifelse(fdr<=fdr_hit_threshold&abs(logFC)>=
                                               log2(fc_hit_threshold),T,F))
```

```
limma_results$hit_annotation<-with(limma_results,ifelse(fdr<=
                                                    fdr_hit_threshold&abs(logFC)>=log2(fc_hit_thr


limma_results$hit_annotation<-factor(limma_results$hit_annotation,ordered=T,levels=
                                       c("hit","candidate","no hit"))

with(limma_results,table(comparison,hit_annotation))
```

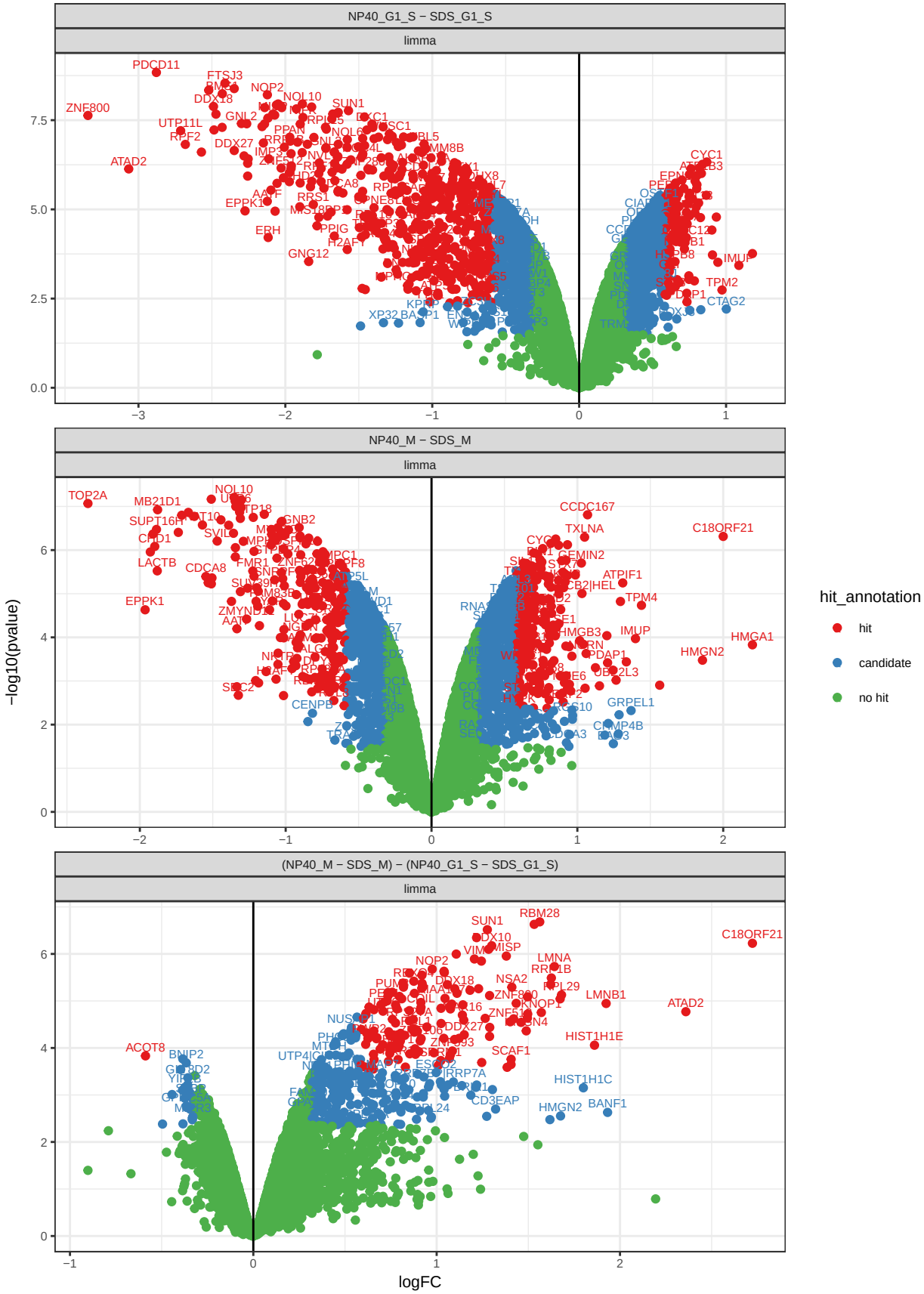
```
##                                             hit_annotation
## comparison                                    hit candidate no hit
##   NP40_G1_S - SDS_G1_S                        651      1130   4469
##   NP40_M - SDS_M                              497      1254   4499
##   (NP40_M - SDS_M) - (NP40_G1_S - SDS_G1_S)   147       252   5851
```

Save limma results

```
hit_list<-subset(limma_results,hit_annotation!="no hit")
write.csv(hit_list,file.path("SDS_vs_NP40_V2","Limma_hitlist_V2.csv"),row.names = F)
write.csv(limma_results,file.path("SDS_vs_NP40_V2","Limma_results_V1.csv"),row.names=F)
hits<-hit_list$gene_name
```

Volcano-Plot

```
qplot(logFC,-log10(pvalue),data=limma_results,colour=hit_annotation)+
  geom_vline(aes(xintercept=0))+facet_wrap(~comparison+
                                             hit_annotation_method,scale="free",ncol = 1)+
  geom_text(aes(label=gene_name),data=subset(limma_results,hit_annotation!="no hit"),
            vjust = 0, nudge_y = 0.1,size=2,check_overlap = T)+
  customPlot
```


NP40_G1_S − SDS_G1_S
limma
NP40_M − SDS_M
limma
(NP40_M − SDS_M) − (NP40_G1_S − SDS_G1_S)
limma
−log10(pvalue)
logFC
hit_annotation
hit
candidate
no hit

```
ggsave(file.path("SDS_vs_NP40_V2","Volcano_plot_no_overlap_V1.pdf"),width=15,height=15)
```

Session info

```
sessionInfo()
```

```
## R version 3.4.3 (2017-11-30)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 7 x64 (build 7601) Service Pack 1
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=German_Germany.1252  LC_CTYPE=German_Germany.1252
## [3] LC_MONETARY=German_Germany.1252 LC_NUMERIC=C
## [5] LC_TIME=German_Germany.1252
##
## attached base packages:
##  [1] grid      stats4    parallel  stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] a4Base_1.26.0        a4Core_1.26.0        a4Preproc_1.26.0
##  [4] glmnet_2.0-13        foreach_1.4.4        Matrix_1.2-12
##  [7] multtest_2.34.0      genefilter_1.60.0    mpm_1.0-22
## [10] KernSmooth_2.23-15   annaffy_1.50.0       KEGG.db_3.2.3
## [13] GO.db_3.5.0          AnnotationDbi_1.40.0 IRanges_2.12.0
## [16] S4Vectors_0.16.0     bindrcpp_0.2         dplyr_0.7.4
## [19] hexbin_1.27.2        gridExtra_2.3        LSD_3.0
## [22] purrr_0.2.4          fdrtool_1.2.15       gplots_3.0.1
## [25] matrixStats_0.52.2   MSnbase_2.4.1        ProtGenerics_1.10.0
## [28] BiocParallel_1.12.0  mzR_2.12.0           Rcpp_0.12.14
## [31] smoothmest_0.1-2     MASS_7.3-48          limma_3.34.5
## [34] vsn_3.46.0           Biobase_2.38.0       BiocGenerics_0.24.0
## [37] ggplot2_2.2.1        reshape_0.8.7        plyr_1.8.4
##
## loaded via a namespace (and not attached):
##  [1] bitops_1.0-6          bit64_0.9-7           RColorBrewer_1.1-2
##  [4] doParallel_1.0.11     rprojroot_1.3-2       tools_3.4.3
##  [7] backports_1.1.2       R6_2.2.2              affyio_1.48.0
## [10] DBI_0.7               lazyeval_0.2.1        colorspace_1.3-2
## [13] bit_1.1-12            compiler_3.4.3        preprocessCore_1.40.0
## [16] labeling_0.3          caTools_1.17.1        scales_0.5.0
## [19] affy_1.56.0           stringr_1.2.0         digest_0.6.14
## [22] rmarkdown_1.8         pkgconfig_2.0.1       htmltools_0.3.6
## [25] rlang_0.1.6           RSQLite_2.0           impute_1.52.0
## [28] BiocInstaller_1.28.0  bindr_0.1             mzID_1.16.0
## [31] gtools_3.5.0          RCurl_1.95-4.10       magrittr_1.5
## [34] MALDIquant_1.17       munsell_0.4.3         stringi_1.1.6
## [37] yaml_2.1.16           zlibbioc_1.24.0       blob_1.1.0
## [40] gdata_2.18.0          lattice_0.20-35       splines_3.4.3
```

```
## [43] annotate_1.56.1       knitr_1.18            pillar_1.1.0
## [46] codetools_0.2-15      XML_3.98-1.9          glue_1.2.0
## [49] evaluate_0.10.1       pcaMethods_1.70.0     gtable_0.2.0
## [52] assertthat_0.2.0      xtable_1.8-2          survival_2.41-3
## [55] tibble_1.4.1          iterators_1.0.9       memoise_1.1.0
```