Data-adaptive pipeline for filtering and normalizing metabolomics data.

Courtney Schiffman, Lauren Petrick, Kelsi Perttula, Yukiko Yano, Henrik Lars Valentin Carlsson, Todd Whitehead, Catherine Metayer, Josie Hayes, William M.B. Edmands, Stephen Rappaport, Sandrine Dudoit.

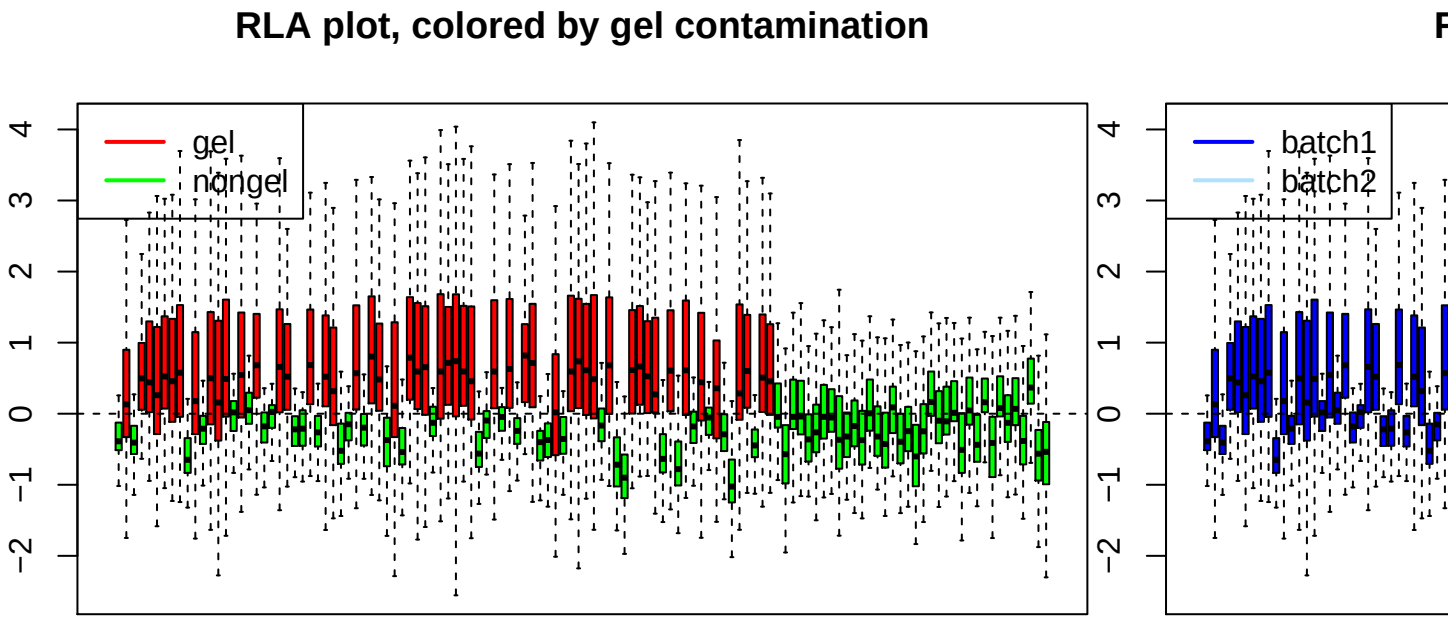
Below is code demonstrating the use of the data-adaptive filtering and normalization pipeline for untargeted metabolomic data described by Schiffman et al. (2018) in "Data-adaptive pipeline for filtering and normalizing metabolomics data.". The example metabolomic dataset contains 122 serum samples (61 incident CRC cases and 61 controls) from Perttula et al. (2016). The data-adaptive pipeline is run to filter samples and features and normalize the data, and several plots and metrics are used to evaluate the method.

Outlier Detection

The first step is to load the data, and to identify the biological samples, blank samples and QC samples. Next, outlier samples are identified using methods from the MetMSLine function pcaOutId. No sample outliers are identified. The data are also plotted with RLA plots to demonstrate the unwanted variation from batch and gel contamination.

```
library(edgeR)
library(EDASeq)
library(scone)
library(xcms)
library(RUVSeq)
library(foreach)
library(doParallel)
registerDoParallel(cores = 4)
library(nlme)
library(EnvStats)
library(MetMSLine)
library(car)

## load xcms object that has had retention time correction and grouping, but
## has not been filled
load("peakTable.RData")

# read in covariates
load("covars.RData")

# observation names (CRC incident cases and controls)
obsNames <- colnames(peakTable)[grep("obsNames", colnames(peakTable))]
# QC sample names
qcNames <- colnames(peakTable)[grep("LocalQC", colnames(peakTable))]
# Blank sample names
blankNames <- colnames(peakTable)[grep("Blank.", colnames(peakTable))][-1]


# Sample names in each batch
obsNames1 <- obsNames[covars$batch == "Batch1"]
```

```r
obsNames2 <- obsNames[covars$batch == "Batch2"]

# Gel contamination status of each sample
gel <- covars$gelPca
gel[gel == FALSE] <- "green"
gel[gel == TRUE] <- "red"

batch <- as.numeric(covars$batch)
batch[batch == 1] <- "blue"
batch[batch == 2] <- "lightskyblue1"

# Plot RLA before normalization to visualize unwanted variation
plotRLE(as.matrix(peakTable[, obsNames]), outline = FALSE, col = gel, xaxt = "n",
    main = "RLA plot, colored by gel contamination")
legend("topleft", legend = c("gel", "nongel"), col = c("red", "green"), lwd = 2)
plotRLE(as.matrix(peakTable[, obsNames]), outline = FALSE, col = batch, xaxt = "n",
    main = "RLA plot, colored by batch")
legend("topleft", legend = c("batch1", "batch2"), col = c("blue", "lightskyblue1"),
    lwd = 2)

filled_peakTable <- MetMSLine::zeroFill(peakTable, obsNames1)
filled_peakTable <- MetMSLine::logTrans(filled_peakTable, obsNames1, base = 2)
pcaOutResults_samples <- MetMSLine::pcaOutId(filled_peakTable, obsNames1, outTol = 1.4,
    center = T, scale = "pareto")


filled_peakTable <- MetMSLine::zeroFill(peakTable, obsNames2)
filled_peakTable <- MetMSLine::logTrans(filled_peakTable, obsNames2, base = 2)
pcaOutResults_samples <- MetMSLine::pcaOutId(filled_peakTable, obsNames2, outTol = 1.4,
    center = T, scale = "pareto")
```



Filtering features with blank samples

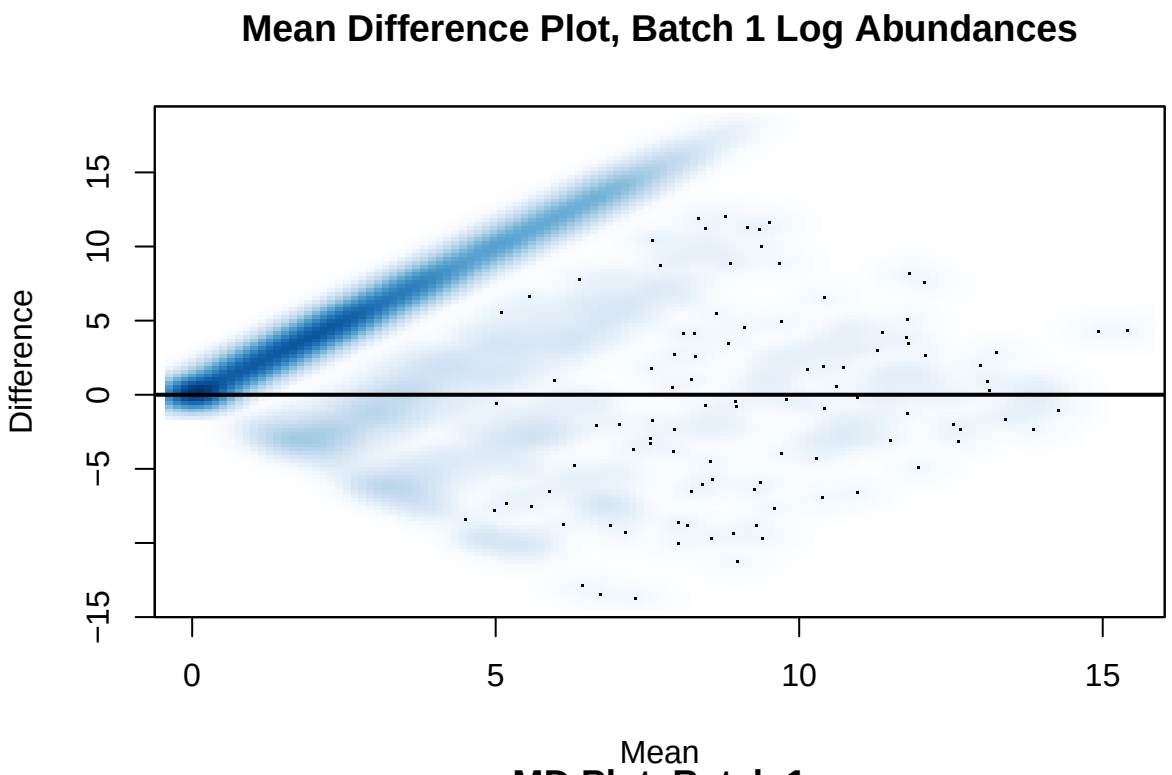
Next, features are filtered based on their average abundances in blank and biological samples, within each batch.

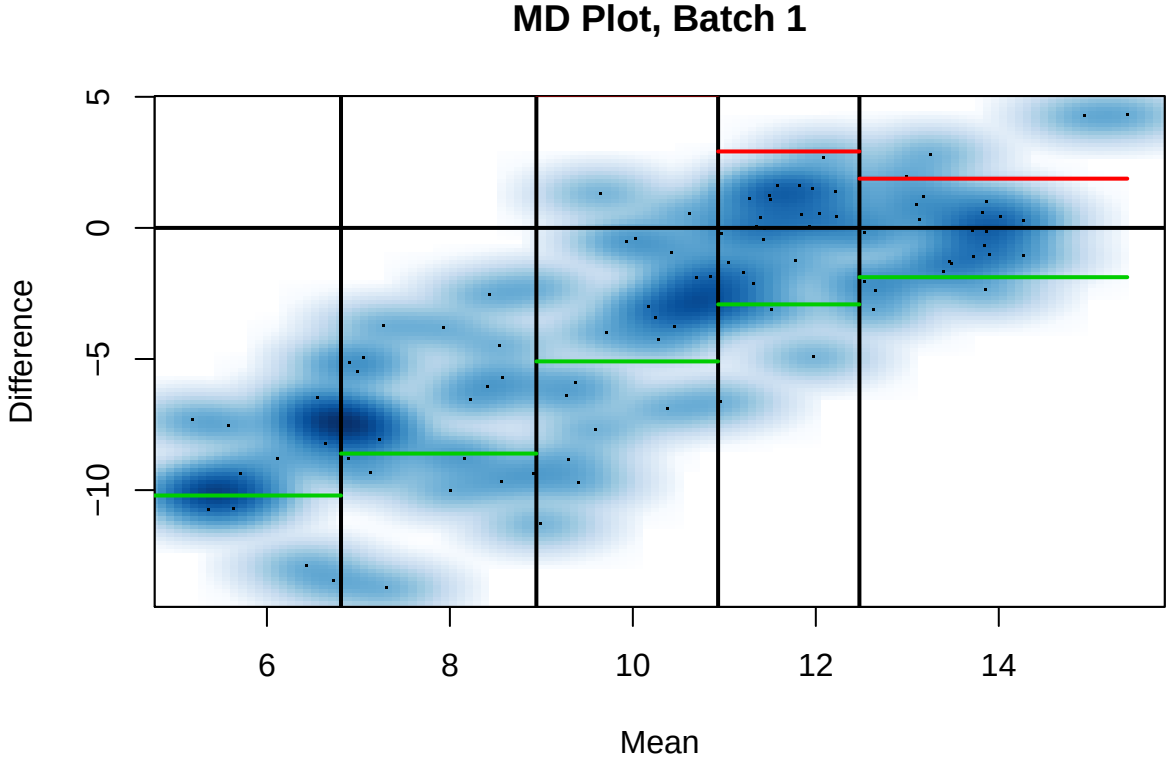
```
# Blank and QC sample names in each batch
blankNames1 <- blankNames[1:3]
blankNames2 <- blankNames[4:6]
qcNames1 <- qcNames[1:20]
qcNames2 <- qcNames[21:32]

# turn missing values to zero
tozero <- function(x){
  x[is.na(x)] <- 0
  return(x)
}

peakTable[,obsNames] <- as.data.frame(t(apply(peakTable[,obsNames],1,tozero)))

# natural log transform all values in the biological samples
peakTable[,obsNames] <- log1p(peakTable[,obsNames])

# Average log abundances across samples in each batch
obsMean1 <- as.vector(apply(peakTable[, obsNames1], 1, mean))
obsMean2 <- as.vector(apply(peakTable[, obsNames2], 1, mean))

peakTable[,blankNames] <- as.data.frame(t(apply(peakTable[,blankNames],1,tozero)))
peakTable[,blankNames] <- log1p(peakTable[,blankNames])

# Average log abundances across blank samples in each batch
blankMean1 <- apply(peakTable[, blankNames1], 1, mean)
blankMean2 <- apply(peakTable[, blankNames2], 1, mean)

# MD-Plots and filtering by blank sample abundances in Batch 1

# Plot all features in a MD-plot
smoothScatter(((blankMean1)+(obsMean1))/2,(obsMean1)-(blankMean1),xlab="Mean",ylab="Difference",main="M
abline(h=0,lwd=2,col="black")

zero.filt <- apply(peakTable[,blankNames1],1, function(x) 0%in%x)
line1 <- rownames(peakTable)[zero.filt] # feature names with >=1 zero in blanks

# define empirical quantiles
quantiles <- c(0.2,0.4,.6,.8,1)
breaks <- quantile(((blankMean1[!zero.filt])+(obsMean1[!zero.filt]))/2,quantiles)

# Begin filtering in the cluster where features are detected in all 3 blank samples
# In each partition, find the lower quartile of the differences below the zero-line and use the absolut

diff1 <- (obsMean1)-(blankMean1)
mean1 <- ((blankMean1)+(obsMean1))/2

# partition 1
less1 <- diff1[!rownames(peakTable)%in%line1 & diff1 <0 & mean1 <= breaks[1]]
```

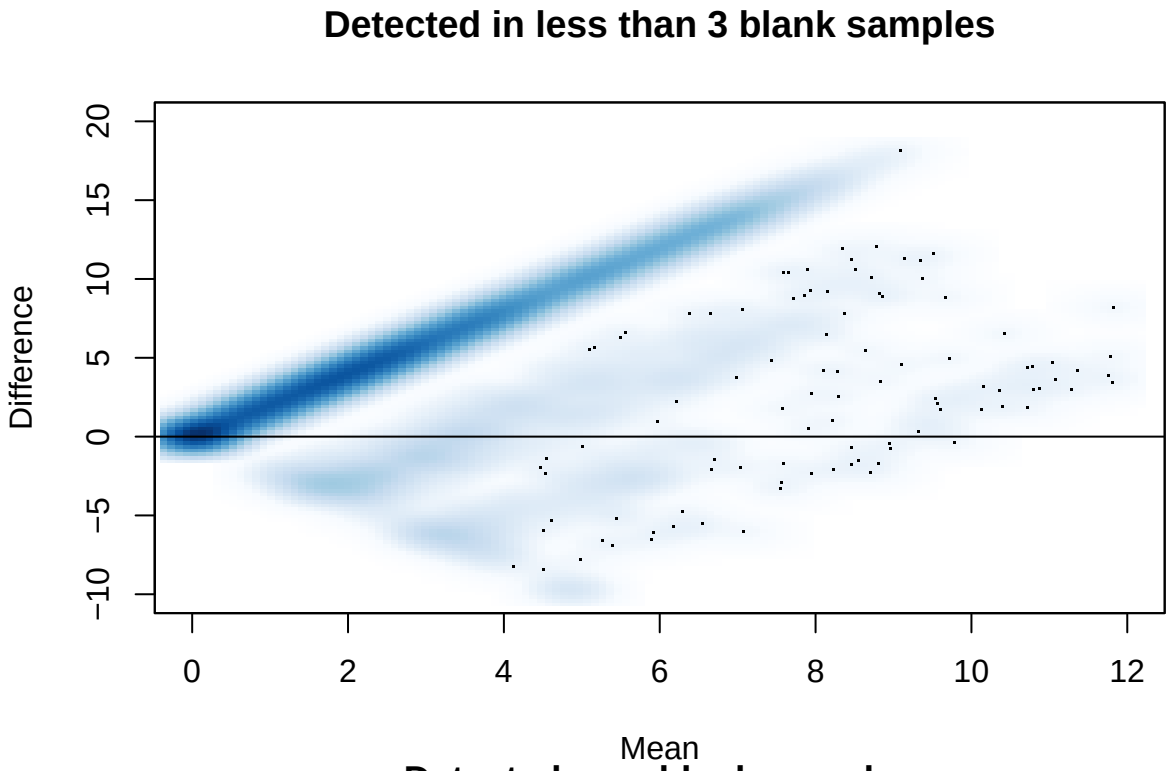
```r
bin1 <- rownames(peakTable)[diff1>0 & mean1 <=breaks[1] & !rownames(peakTable)%in%line1 & diff1> abs(sum

# partition 2
less2 <- diff1[!rownames(peakTable)%in%line1 & diff1 <0 & mean1 <=breaks[2] & mean1 > breaks[1]]
bin2 <- rownames(peakTable)[diff1>0 & mean1 <= breaks[2] & mean1> breaks[1] & !rownames(peakTable)%in%li

# partition 3
less3 <- diff1[!rownames(peakTable)%in%line1 & diff1 <0 & mean1 <= breaks[3] & mean1 >breaks[2]]
bin3 <- rownames(peakTable)[diff1>0 & mean1 <= breaks[3] & mean1>breaks[2] & !rownames(peakTable)%in%lin

# partition 4
less4 <- diff1[!rownames(peakTable)%in%line1 & diff1 <0 & mean1 <=breaks[4] & mean1 >breaks[3]]
bin4 <- rownames(peakTable)[diff1>0 & mean1 <=breaks[4] & mean1> breaks[3] & !rownames(peakTable)%in%lin

# partition 5
less5 <- diff1[!rownames(peakTable)%in%line1 & diff1 <0 & mean1 > breaks[4] ]
bin5 <- rownames(peakTable)[diff1>0 & mean1 > breaks[4]  & !rownames(peakTable)%in%line1 & diff1> abs(su

## Plot the data-adaptive cutoffs for each partition
smoothScatter(((blankMean1[!zero.filt])+(obsMean1[!zero.filt]))/2,(obsMean1[!zero.filt])-(blankMean1[!ze
abline(h=0,lwd=2,col="black")
abline(v=breaks[1],lwd=2)
abline(v=breaks[2],lwd=2)
abline(v=breaks[3],lwd=2)
abline(v=breaks[4],lwd=2)
segments(x0=0,y0=summary(less1)[2],x1=breaks[1],col="green3",lwd=2)
segments(x0=0,y0=abs(summary(less1)[2]),x1=breaks[1],col="red",lwd=2)
segments(x0=breaks[1],y0=summary(less2)[2],x1=breaks[2],col="green3",lwd=2)
segments(x0=breaks[1],y0=abs(summary(less2)[2]),x1=breaks[2],col="red",lwd=2)
segments(x0=breaks[2],y0=summary(less3)[2],x1=breaks[3],col="green3",lwd=2)
segments(x0=breaks[2],y0=abs(summary(less3)[2]),x1=breaks[3],col="red",lwd=2)
segments(x0=breaks[3],y0=summary(less4)[2],x1=breaks[4],col="green3",lwd=2)
segments(x0=breaks[3],y0=abs(summary(less4)[2]),x1=breaks[4],col="red",lwd=2)
segments(x0=breaks[4],y0=summary(less5)[2],x1=breaks[5],col="green3",lwd=2)
segments(x0=breaks[4],y0=abs(summary(less5)[2]),x1=breaks[5],col="red",lwd=2)


## Now filter features in the remaining three clusters (detected in zero, one or two blanks samples)

num.zero <- apply(peakTable[,blankNames1],1, function(x) sum(x==0))
smoothScatter(((blankMean1[zero.filt])+(obsMean1[zero.filt]))/2,(obsMean1[zero.filt])-(blankMean1[zero.f
abline(h=0,lwd=1)

## Not detected in any blank samples
smoothScatter(((blankMean1[num.zero==3])+(obsMean1[num.zero==3]))/2,(obsMean1[num.zero==3])-(blankMean1
abline(h=0,lwd=1)

## Not detected in two of the three blank samples
smoothScatter(((blankMean1[num.zero==2])+(obsMean1[num.zero==2]))/2,(obsMean1[num.zero==2])-(blankMean1
abline(h=abs(summary(diff1[diff1<0 & num.zero==2])[2]),col='green3')
abline(h=c(summary(diff1[diff1<0 & num.zero==2])[2]),col='red')
abline(h=0,lwd=1)
```

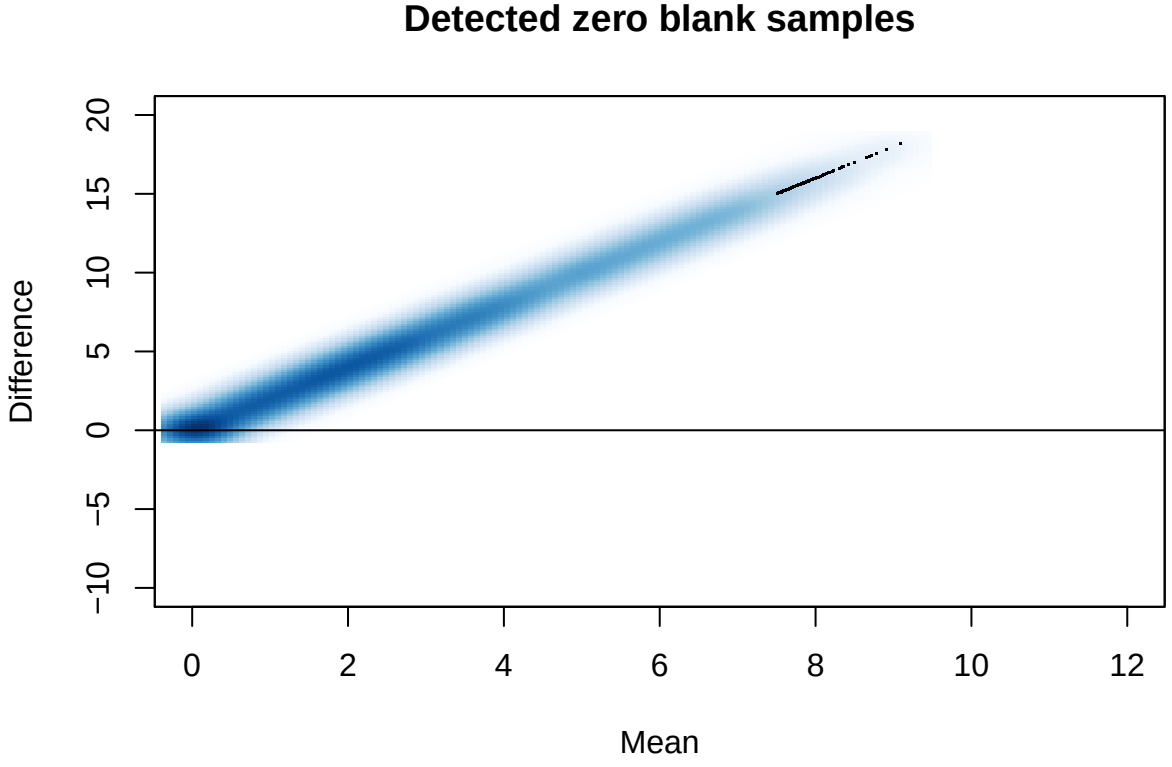
```
## Not detected in one of the blank samples
smoothScatter(((blankMean1[num.zero==1])+(obsMean1[num.zero==1]))/2,(obsMean1[num.zero==1])-(blankMean1
abline(h=abs(summary(diff1[diff1<0 & num.zero==1])[2]),col='green3')
abline(h=c(summary(diff1[diff1<0 & num.zero==1])[2]),col='red')
abline(h=0,lwd=1)


line1 <- rownames(peakTable)[num.zero==3 | (num.zero==2 & diff1>abs(summary(diff1[diff1<0 & num.zero==2

# features remaining in the first batch
batch1.features <- c(line1,bin1,bin2,bin3,bin4,bin5) # 20147


# Repeat filtering for batch 2
smoothScatter(((blankMean2)+(obsMean2))/2,(obsMean2)-(blankMean2),xlab="Mean",ylab="Difference",main="M
abline(h=0,col="black")

zero.filt <- apply(peakTable[,blankNames2],1, function(x) 0%in%x)

line1 <- rownames(peakTable)[zero.filt] # feature names with zeros

quantiles <- c(0.2,0.4,.6,.8,1)

breaks <- quantile(((blankMean2[!zero.filt])+(obsMean2[!zero.filt]))/2,quantiles)

diff1 <- (obsMean2)-(blankMean2)
mean1 <- ((blankMean2)+(obsMean2))/2
less1 <- diff1[!rownames(peakTable)%in%line1 & diff1 <0 & mean1 <= breaks[1]]
bin1 <- rownames(peakTable)[diff1>0 & mean1 <=breaks[1] & !rownames(peakTable)%in%line1 & diff1> abs(su

less2 <- diff1[!rownames(peakTable)%in%line1 & diff1 <0 & mean1 <=breaks[2] & mean1 > breaks[1]]
bin2 <- rownames(peakTable)[diff1>0 & mean1 <= breaks[2] & mean1> breaks[1] & !rownames(peakTable)%in%li

less3 <- diff1[!rownames(peakTable)%in%line1 & diff1 <0 & mean1 <= breaks[3] & mean1 >breaks[2]]
bin3 <- rownames(peakTable)[diff1>0 & mean1 <= breaks[3] & mean1>breaks[2] & !rownames(peakTable)%in%lir

less4 <- diff1[!rownames(peakTable)%in%line1 & diff1 <0 & mean1 <=breaks[4] & mean1 >breaks[3]]
bin4 <- rownames(peakTable)[diff1>0 & mean1 <=breaks[4] & mean1> breaks[3] & !rownames(peakTable)%in%lir

less5 <- diff1[!rownames(peakTable)%in%line1 & diff1 <0 & mean1 > breaks[4] ]
bin5 <- rownames(peakTable)[diff1>0 & mean1 > breaks[4]  & !rownames(peakTable)%in%line1 & diff1> abs(su


smoothScatter(((blankMean2[!zero.filt])+(obsMean2[!zero.filt]))/2,(obsMean2[!zero.filt])-(blankMean2[!z
abline(h=0,lwd=2,col="black")
abline(v=breaks[1],lwd=2)
abline(v=breaks[2],lwd=2)
abline(v=breaks[3],lwd=2)
abline(v=breaks[4],lwd=2)
segments(x0=-5,y0=summary(less1)[2],x1=breaks[1],col="red",lwd=2)
segments(x0=-5,y0=abs(summary(less1)[2]),x1=breaks[1],col="green3",lwd=2)
segments(x0=breaks[1],y0=summary(less2)[2],x1=breaks[2],col="red",lwd=2)
segments(x0=breaks[1],y0=abs(summary(less2)[2]),x1=breaks[2],col="green3",lwd=2)
```

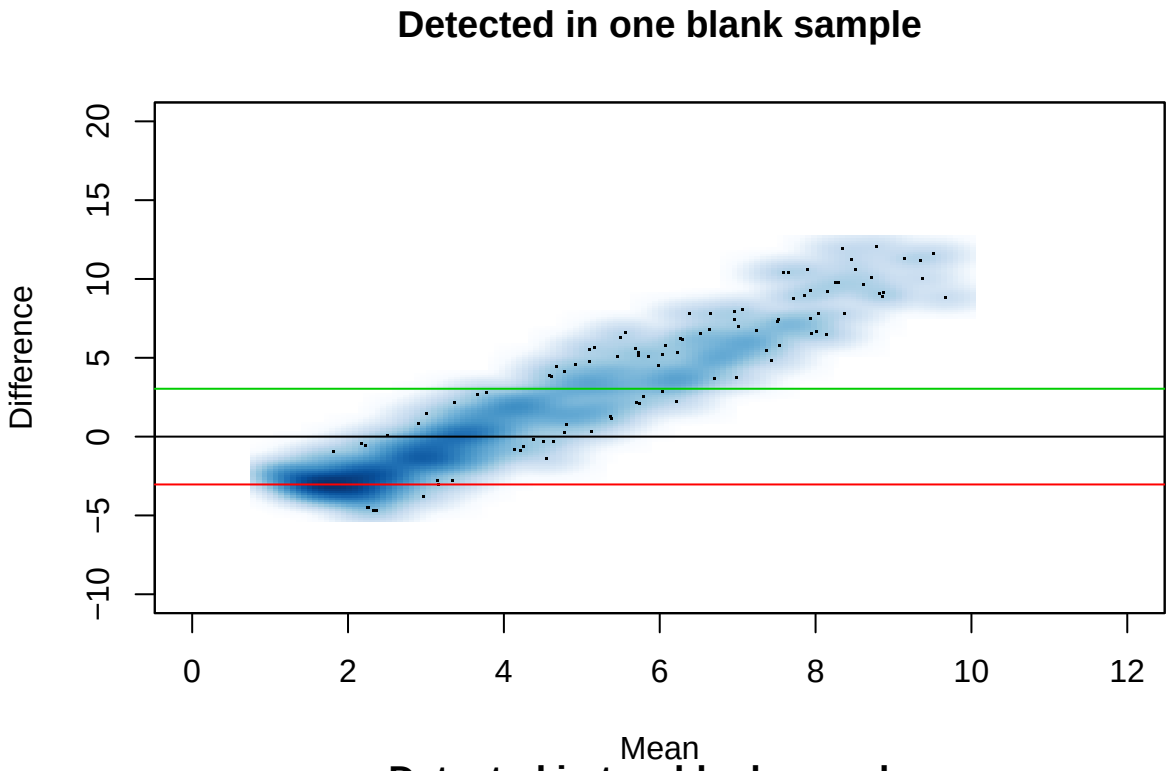
```
segments(x0=breaks[2],y0=summary(less3)[2],x1=breaks[3],col="red",lwd=2)
segments(x0=breaks[2],y0=abs(summary(less3)[2]),x1=breaks[3],col="green3",lwd=2)
segments(x0=breaks[3],y0=summary(less4)[2],x1=breaks[4],col="red",lwd=2)
segments(x0=breaks[3],y0=abs(summary(less4)[2]),x1=breaks[4],col="green3",lwd=2)
segments(x0=breaks[4],y0=summary(less5)[2],x1=20,col="red",lwd=2)
segments(x0=breaks[4],y0=abs(summary(less5)[2]),x1=20,col="green3",lwd=2)



num.zero <- apply(peakTable[,blankNames2],1, function(x) sum(x==0))
smoothScatter(((blankMean2[zero.filt])+(obsMean2[zero.filt]))/2,(obsMean2[zero.filt])-(blankMean2[zero.
abline(h=0,lwd=1)
smoothScatter(((blankMean2[num.zero==3])+(obsMean2[num.zero==3]))/2,(obsMean2[num.zero==3])-(blankMean2
abline(h=0,lwd=1)

smoothScatter(((blankMean2[num.zero==2])+(obsMean2[num.zero==2]))/2,(obsMean2[num.zero==2])-(blankMean2
abline(h=abs(summary(diff1[diff1<0 & num.zero==2])[2]),col='green3')
abline(h=c(summary(diff1[diff1<0 & num.zero==2])[2]),col='red')
abline(h=0,lwd=1)

smoothScatter(((blankMean2[num.zero==1])+(obsMean2[num.zero==1]))/2,(obsMean2[num.zero==1])-(blankMean2
abline(h=abs(summary(diff1[diff1<0 & num.zero==1])[2]),col='green3')
abline(h=c(summary(diff1[diff1<0 & num.zero==1])[2]),col='red')
abline(h=0,lwd=1)

line1 <- rownames(peakTable)[num.zero==3 | (num.zero==2 & diff1>abs(summary(diff1[diff1<0 & num.zero==2

batch2.features <- c(line1,bin1,bin2,bin3,bin4,bin5)

# take the intersection of the features remaining in batch 1 and batch 2
features <- intersect(batch1.features,batch2.features)

peakTable2 <- peakTable[rownames(peakTable)%in%features,]
```

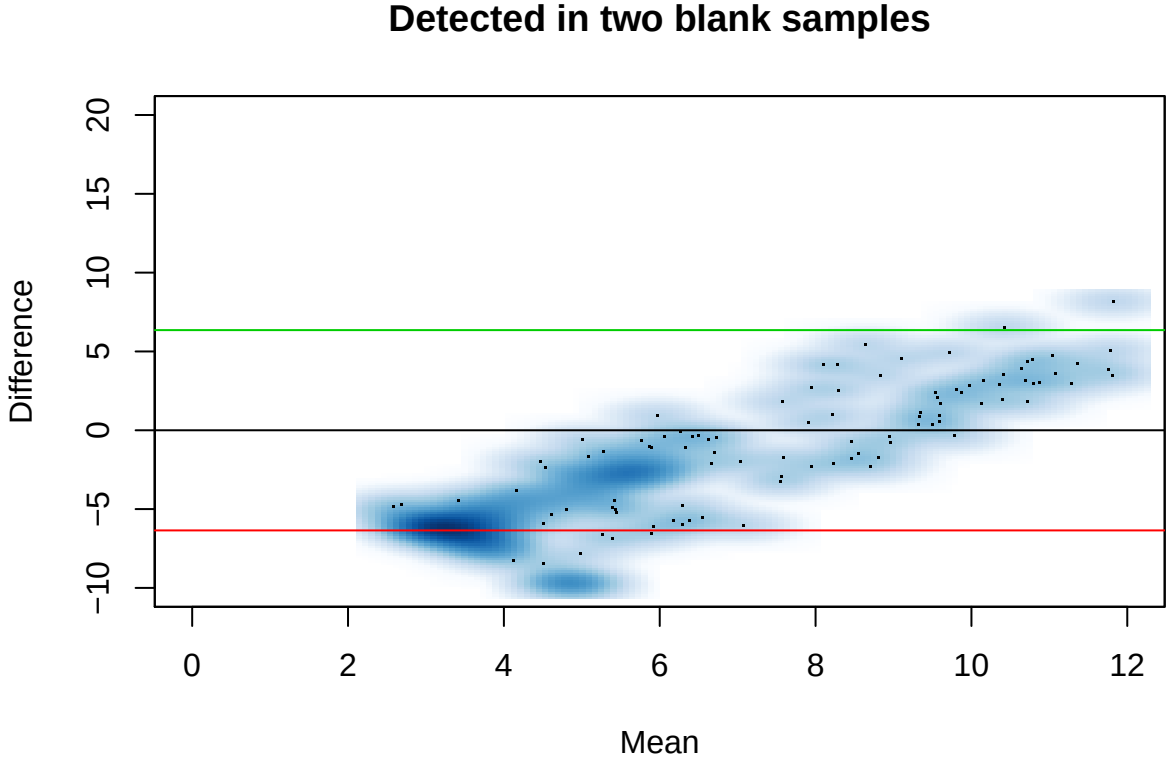

Mean Difference Plot, Batch 1 Log Abundances
Difference
15
10
5
0
−5
−15
0
5
10
15
Mean

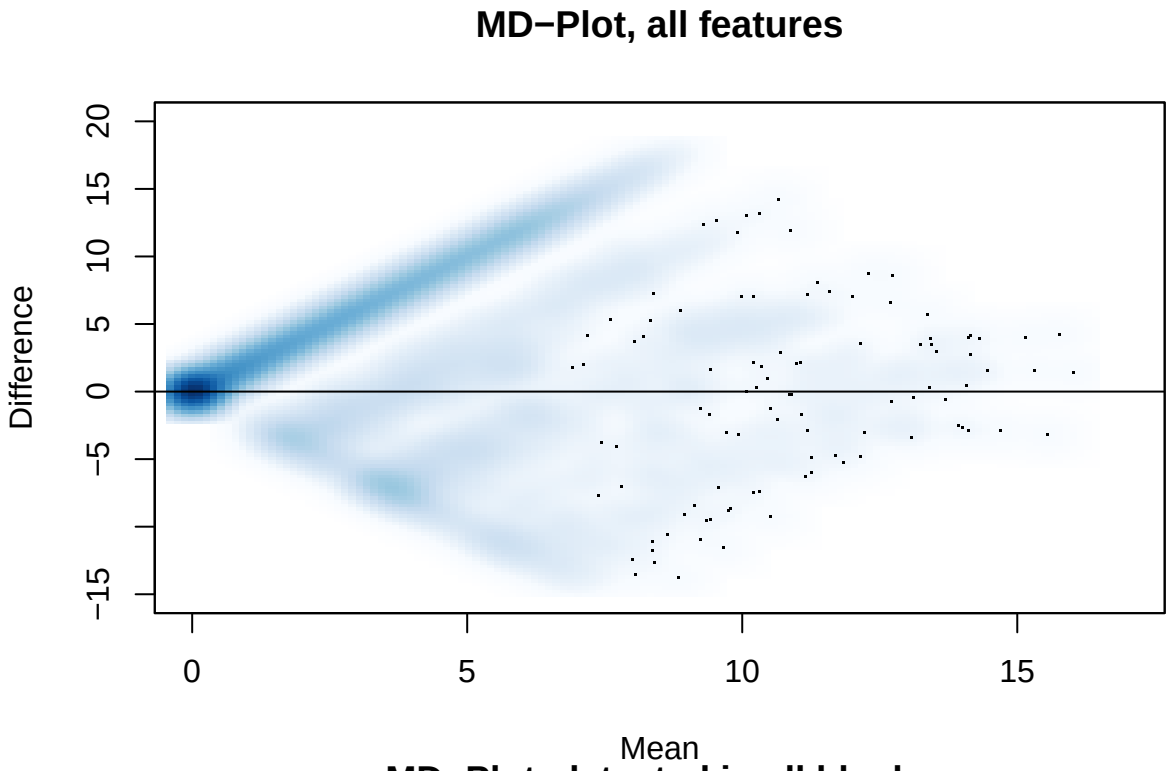

MD Plot, Batch 1
Difference
5
0
−5
−10
6
8
10
12
14
Mean

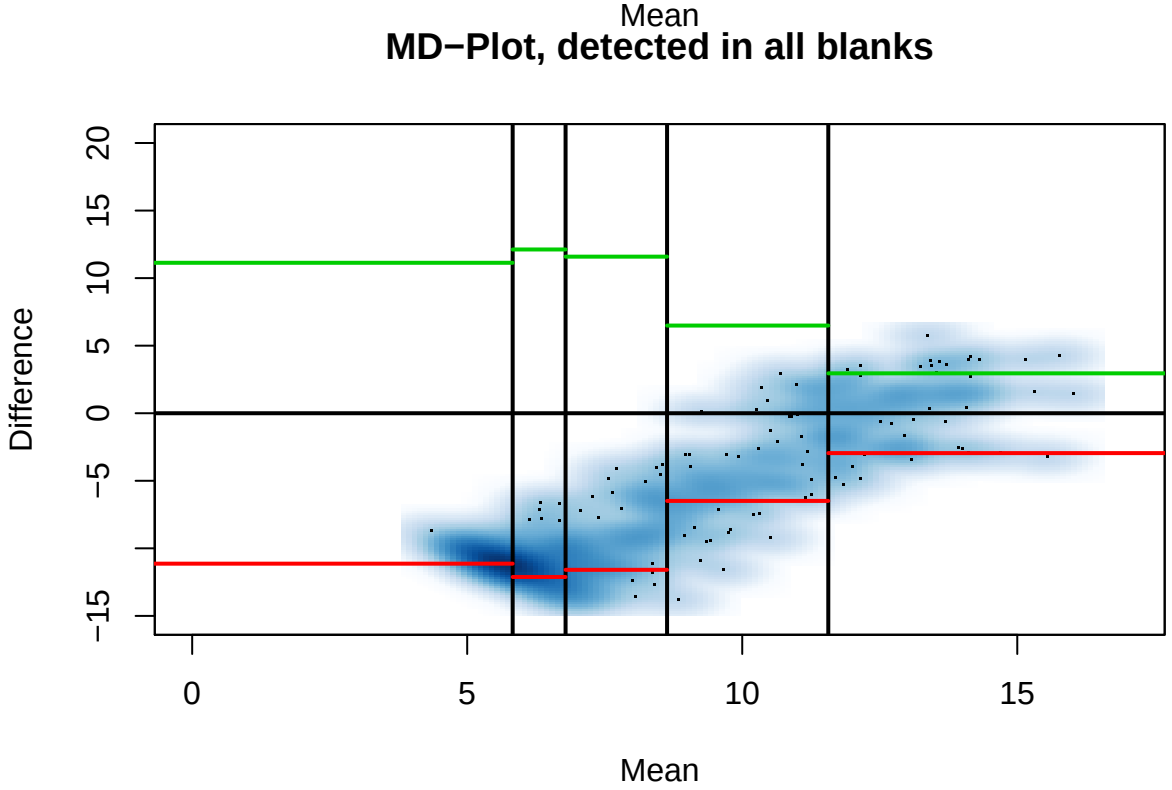

Detected in less than 3 blank samples
Difference
20
15
10
5
0
−5
−10
0
2
4
6
8
10
12
Mean

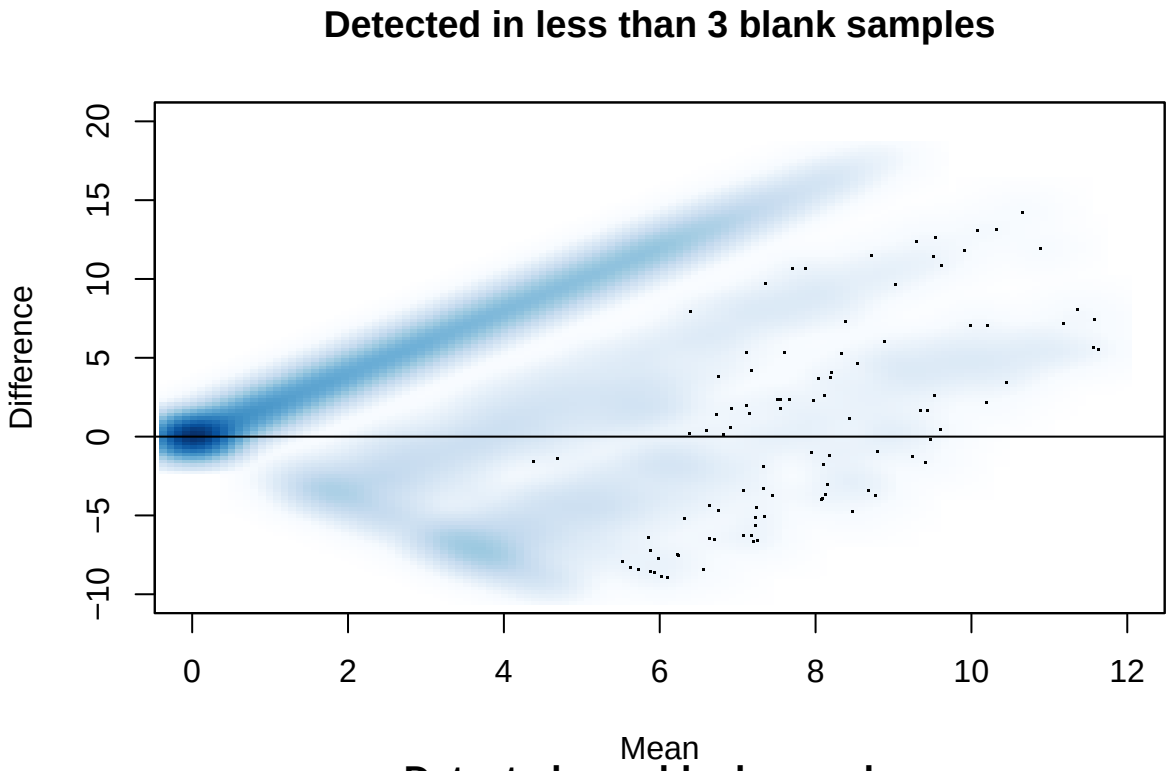

Detected zero blank samples
Difference
20
15
10
5
0
−5
−10
0
2
4
6
8
10
12
Mean

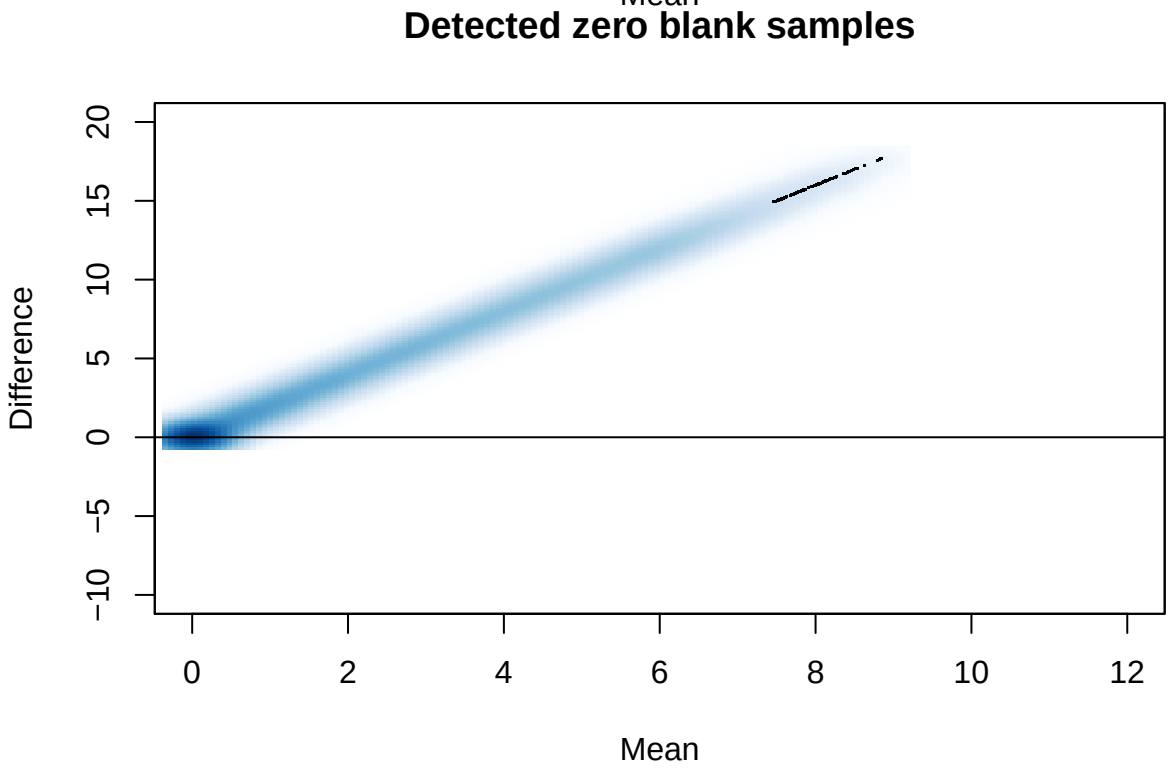

Detected in one blank sample
Difference
20
15
10
5
0
−5
−10
0
2
4
6
8
10
12
Mean

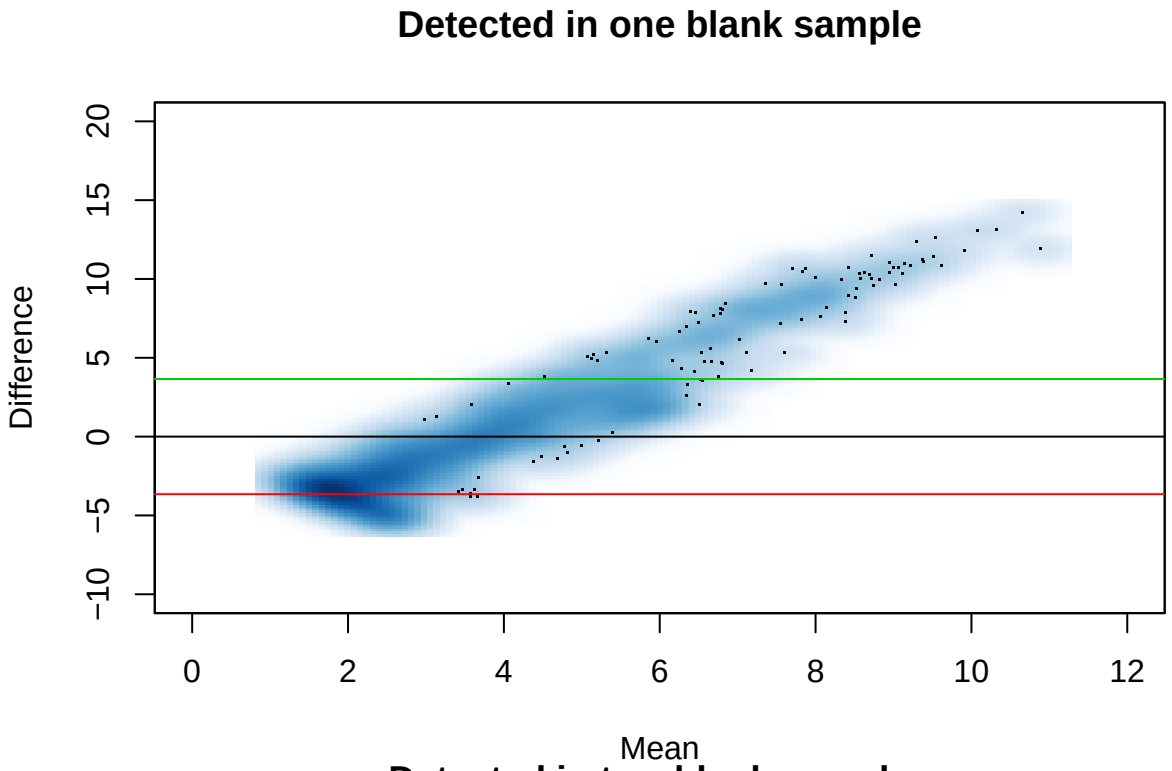

Detected in two blank samples
Difference
20
15
10
5
0
−5
−10
0
2
4
6
8
10
12
Mean

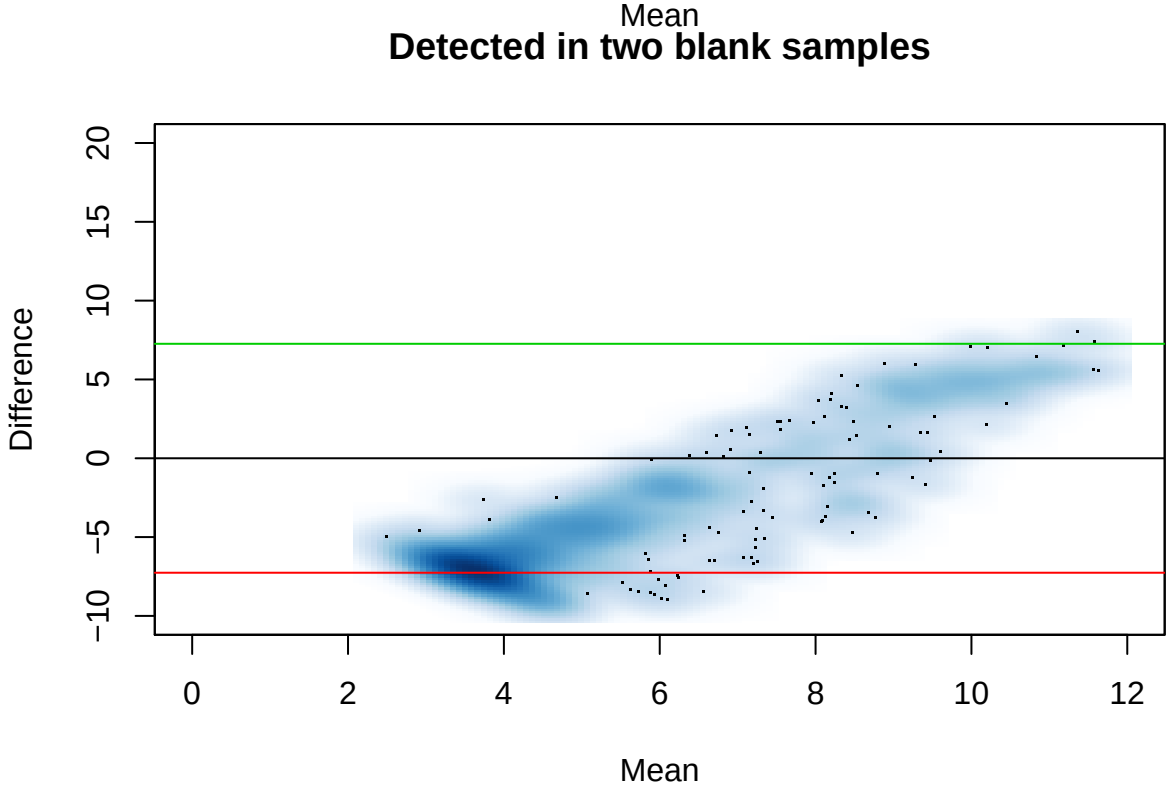

MD−Plot, all features
Difference
20
15
10
5
0
−5
−15
0
5
10
15
Mean


MD−Plot, detected in all blanks
Difference
20
15
10
5
0
−5
−15
0
5
10
15
Mean


Detected in less than 3 blank samples
Difference
20
15
10
5
0
−5
−10
0
2
4
6
8
10
12
Mean


Detected zero blank samples
Difference
20
15
10
5
0
−5
−10
0
2
4
6
8
10
12
Mean

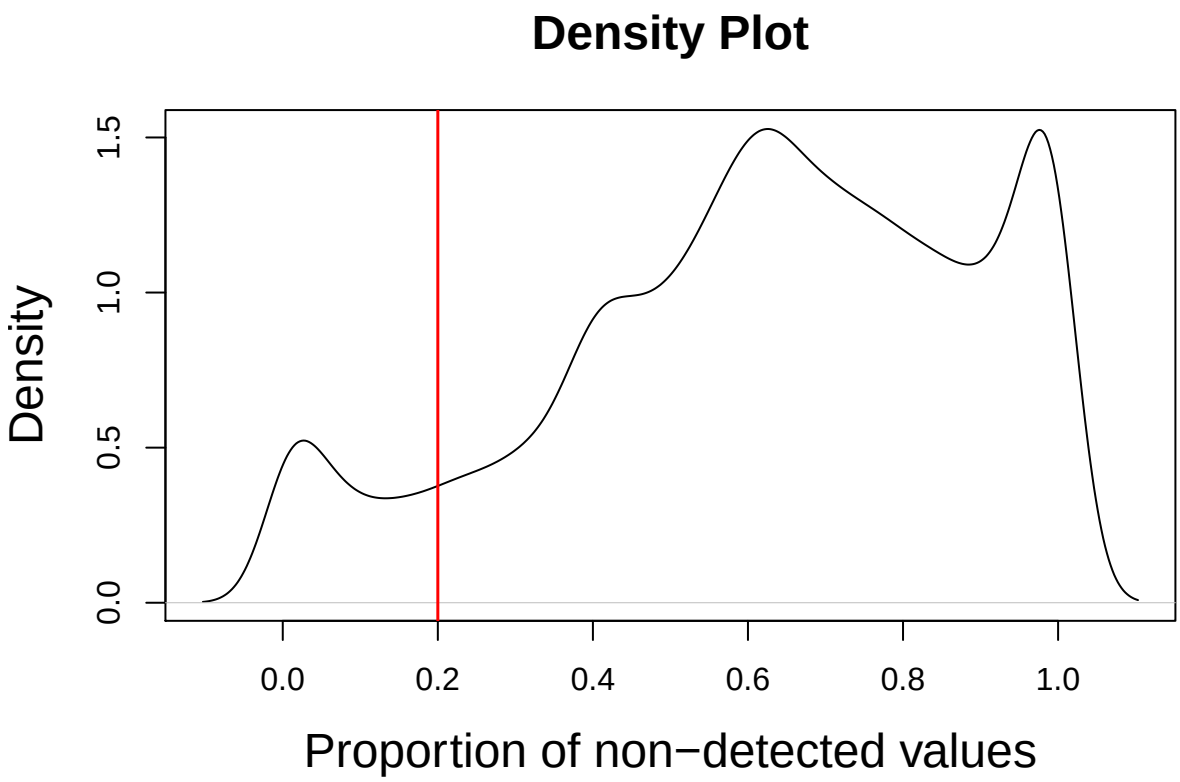


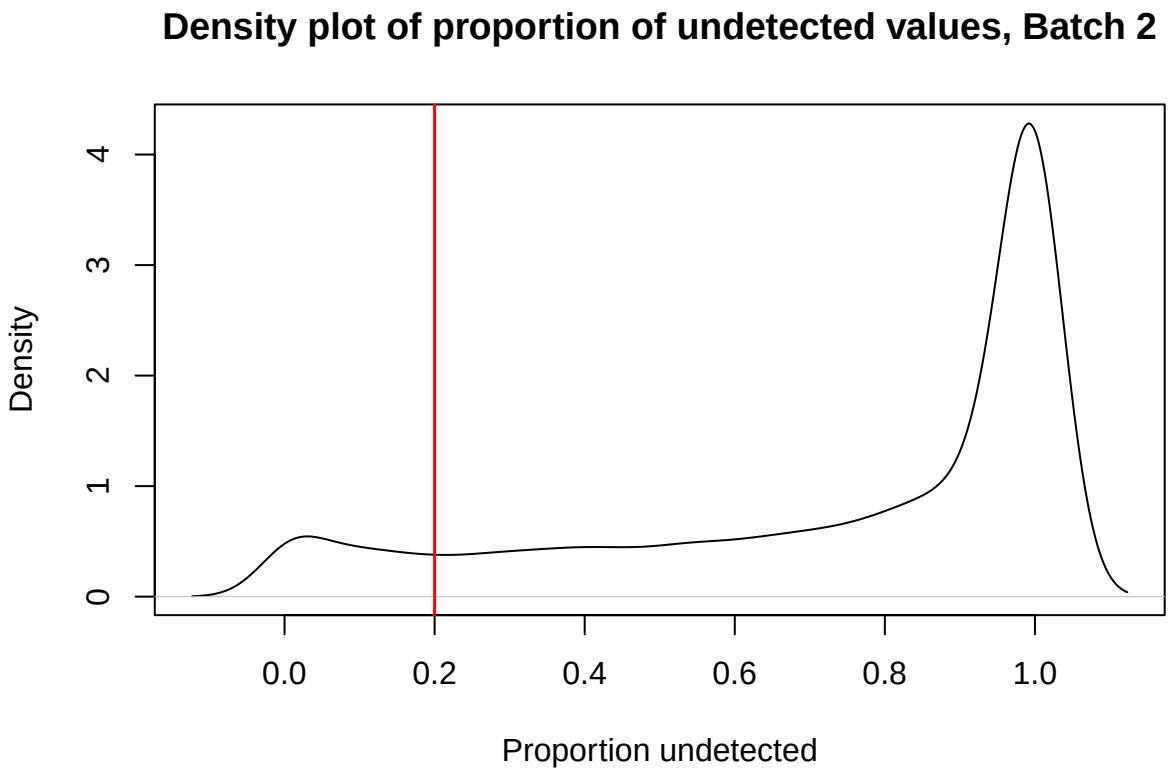


Filtering by percent missing

A density plot of percent missing values for each feature helps to determine a data-adaptive cutoff. Here, at most 20% missing in each batch is appropriate.

```r
# Density plot of percent missing in each batch

percent.zero1 <- apply(peakTable2[,obsNames1],1,function(x) sum(x==0)/length(obsNames1))
plot(density(percent.zero1),main="Density Plot",xlab="Proportion of non-detected values",cex.lab=1.5,ce
abline(v=0.2,lwd=1.5,col="red")

percent.zero2 <- apply(peakTable2[,obsNames2],1,function(x) sum(x==0)/length(obsNames2))
plot(density(percent.zero2),main="Density plot of proportion of undetected values, Batch 2",xlab="Propo
abline(v=0.2,lwd=1.5,col="red")

# Below 20% missing in each batch
percent.zeros <- percent.zero1<=0.2 & percent.zero2<=0.2
names(percent.zeros) <- rownames(peakTable2)

peakTable3 <- peakTable2[(percent.zero1<=0.2 & percent.zero2<=0.2),]
```





k-NN imputation

Missing values in the biological and QC samples are imputed with our variation on k-NN imputation, with 5 nearest feature neighbors.

```
peakTable3[,c(qcNames1,qcNames2)] <- as.data.frame(t(apply(peakTable3[,c(qcNames1,qcNames2)],1,tozero)))
peakTable3[,c(qcNames1,qcNames2)] <-log1p(peakTable3[,c(qcNames1,qcNames2)])


# FUNCTION TO PERFORM KNN IMPUTATION, DATA IS A metabolites BY SAMPLES MATRIX

myknn <- function(k,data){
  D <- dist(data) # compute distance between the rows of the data matrix
  D <- as.matrix(D)
  ranked <- t(apply(D,1,rank))
  norm <- data
  for (i in 1:nrow(data)){
    missing <- which(data[i,]==0) # for adduct i, index of which columns are missing/zero
    for (j in missing){
      n <- c()
      l=1 # start by finding the first nearest neighbor
      while (length(n)<k){
        neighbor <- which((ranked[i,]-1)==l) # which adduct is the lth nearest neighbor to adduct i? Mi
        if (!(data[neighbor,j]==0)){ # checking if the neighbor's value for the value j is zero
          n <- c(n,neighbor) # if not missing count it as a neighbor and move to the next one
          l=l+1
        } else {
          l=l+1 # if it is missing move to the next nearest
```

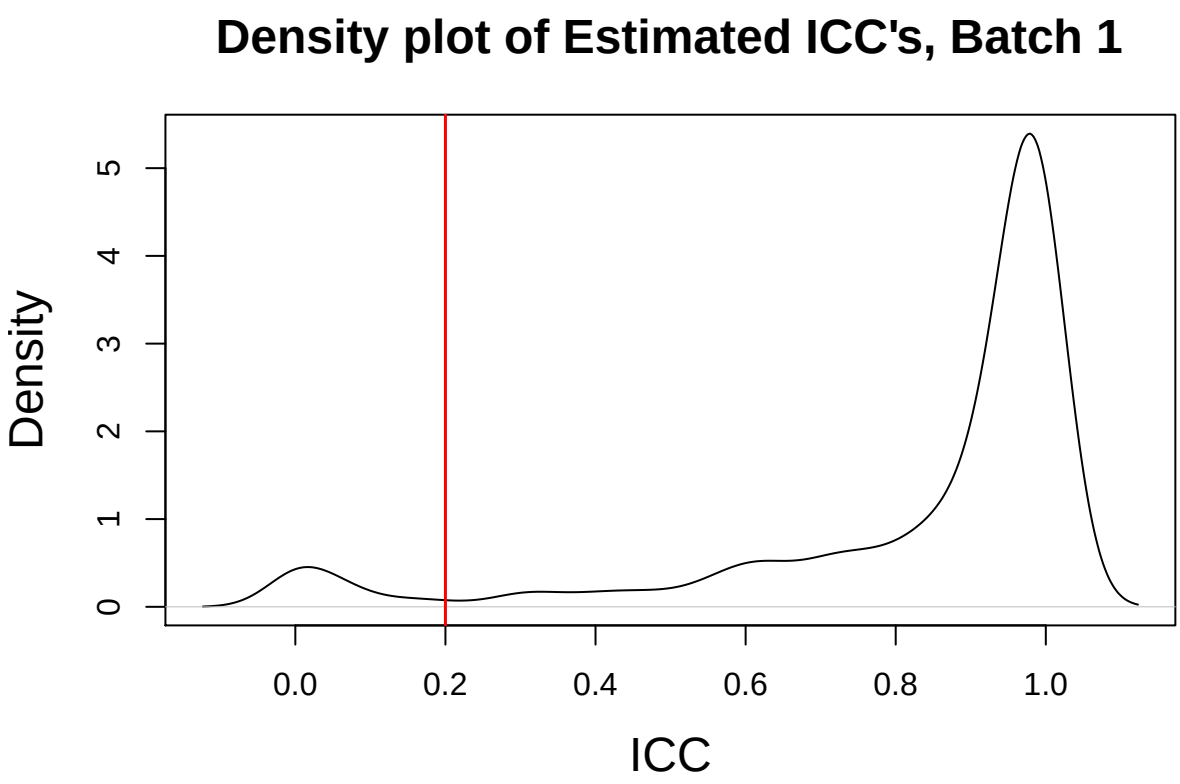
```
          n <- c(n)
        }
      }
      norm[i,j] <- mean(data[n,j])
    }
  }
  return(norm)
}

## Impute missing values in observed and QC samples
peakTable3[,c(obsNames,qcNames1,qcNames2)] <- as.data.frame(myknn(5,peakTable3[,c(obsNames,qcNames1,qcNa
```

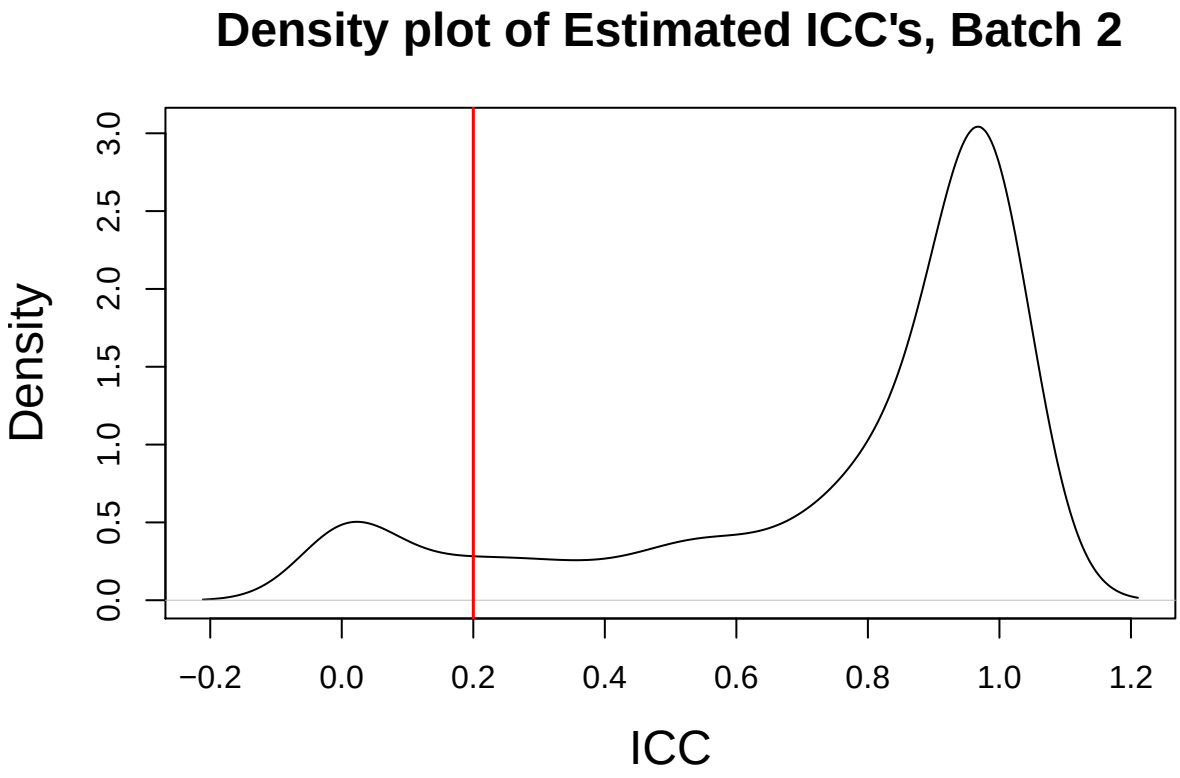
Filtering by ICC

The ICC for each feature is estimated in each batch using mixed effects models. A density plot of the estimated ICC's helps to determine a data-adaptive cutoff. Here, features with ICC's less than 0.2 are removed.

```
## Use mixed effects models to estimate ICC's within each batch

reps <- factor(c(1:length(obsNames1),rep(length(obsNames1)+1,length(qcNames1))))

vars1 <- foreach(i=(1:nrow(peakTable3)),.packages='nlme',.combine='rbind')%dopar%{
  reps <- factor(c(1:length(obsNames1),rep(length(obsNames1)+1,length(qcNames1))))
  data <- data.frame(y=as.numeric(peakTable3[i,c(obsNames1,qcNames1)]))
  mm <- lme(y~ 1, random = ~1|reps, data=data)
  return(as.numeric(VarCorr(mm)[1:2]))
}

rownames(vars1) <- rownames(peakTable3)
colnames(vars1) <- c("Intercept","Residual")

# Density plot of ICC values in batch 1
ICC1 <- apply(vars1,1,function(x) x[1]/sum(x))
plot(density(ICC1),main="Density plot of Estimated ICC's, Batch 1",xlab="ICC",cex.lab=1.5,cex.main=1.5)
abline(v=0.2,lwd=1.5,col="red")
ICC <- ICC1>.2


reps <- factor(c(1:length(obsNames2),rep(length(obsNames2)+1,length(qcNames2))))

vars2 <- foreach(i=(1:nrow(peakTable3)),.packages='nlme',.combine='rbind')%dopar%{
  data <- data.frame(y=as.numeric(peakTable3[i,c(obsNames2,qcNames2)]))
  mm <- lme(y~ 1, random = ~1|reps, data=data)
  return(as.numeric(VarCorr(mm)[1:2]))
}

rownames(vars2) <- rownames(peakTable3)
colnames(vars2) <- c("Intercept","Residual")

ICC2 <- apply(vars2,1,function(x) x[1]/sum(x))
plot(density(ICC2),main="Density plot of Estimated ICC's, Batch 2",xlab="ICC",cex.lab=1.5,cex.main=1.5)
```

```r
abline(v=0.2,lwd=1.5,col="red")

ICC <- ICC+(ICC2>.2)

## Keep features with ICC's above 0.2 in each batch
peakTable3 <- peakTable3[ICC==2,] # 378
```





Normalization with scone

In order to allow for estimated factors of unwanted variation in scone, empirical control features are estimate. Then, plotting estimate factors of unwanted variation shows that the first two factors are related to batch and gel contamination. Next, the biological factor of interest, scaling functions, batch, and QC matrix are defined. Here, we define a new batch factor that includes a cross between batch and gel contamination so that both factors are adjusted for. We can see that the top ranking scheme includes DESEQ scaling, adjustment for the batch/gel factor and adjustment for the QC matrix (run-order). An RLA plot of the normalized data shows an appropriate removal of unwanted variation. The scone metrics used by Schiffman et al. to evaluate the pipeline can be found by the first five columns of the 'get_scores' function.

```
# calculate empirical control features for use in scone
x <- factor(covars$case.control.status[covars$batch=='Batch2'])
x <- factor(x,levels=c('FALSE',"TRUE"),labels=c(0,1))
design <- model.matrix(~x, data=data.frame(x, row.names=obsNames2))
y <- DGEList(counts=round(peakTable3[,obsNames2]), group=x)
y <- calcNormFactors(y, method="upperquartile")
y <- estimateGLMCommonDisp(y, design)
y <- estimateGLMTagwiseDisp(y, design)
et <- exactTest(y)
top <- topTags(et, n=nrow(peakTable3))$table
empirical2 <- rownames(peakTable3)[which(!(rownames(peakTable3) %in% rownames(top)[1:(nrow(peakTable3)/

# New batch factor that captures unwanted variation from batch and gel contamination
scone.batch <- (covars$batch):as.factor(covars$gelPca)

# Calculate the first few estimated factors of unwanted variation to see what they correlate with
ruvgset <- RUVg(round(as.matrix(peakTable3[,obsNames])),empirical2,k=5,isLog = T)
```

```r
plot(ruvgset$W[,1],pch=19,col=as.numeric(scone.batch)+1,xlab=NULL,ylab="W1",main="First factor of unwan
legend('topright',legend=c('Batch 1','Batch 1 Gel','Batch 2'),col=c(2,3,4),lwd=2,cex=.8)

plot(ruvgset$W[,2],pch=19,col=as.numeric(scone.batch)+1,xlab=NULL,ylab="W2",main="Second factor of unwa
legend('bottomright',legend=c('Batch 1','Batch 1 Gel','Batch 2'),col=c(2,3,4),lwd=2,cex=.8)

# Scone normalization
# biology of interest
bio <- factor(covars$case.control.status)
bio <- factor(bio,levels=c('FALSE',"TRUE"),labels=c(0,1))
# New batch x gel factor
batch <- as.factor(covars$batch)
# sample scaling functions
scaling <- c(none=identity,deseq = DESEQ_FN,uq=UQ_FN)
# further source of unwanted variation is run-order
qc <- as.matrix(as.numeric(c(c(1:86),c(1:36))))

# allow for 2 estimated factors of unwanted variation
k_ruv <- 2
k_qc <- 1


my_scone <- SconeExperiment(exp(as.matrix(peakTable3[,obsNames]))-1, bio =bio,qc=qc,
                            negcon_ruv = rownames(peakTable3) %in% empirical2,
                            batch=scone.batch)

my_scone <- scone(my_scone,scaling = scaling,
                  k_ruv = k_ruv,adjust_bio = "yes", k_qc = k_qc,
                  adjust_batch = "yes",
                  run=FALSE,verbose=TRUE)


is_screened <- (get_params(my_scone)$uv_factors %in% c("ruv_k=1","ruv_k=2")) & (get_params(my_scone)$ad
my_scone = select_methods(my_scone,rownames(get_params(my_scone))[!is_screened])
#get_params(my_scone)[,-1]



my_scone <- scone(my_scone,scaling = scaling,k_ruv = k_ruv,adjust_bio = "yes", k_qc = k_qc,
                  adjust_batch = "yes",return_norm = "no",
                  run=TRUE,verbose=TRUE,eval_kclust = 2:6)



## View ranking of normalization schemes
head(get_score_ranks(my_scone),20)
```

```
##        none,deseq,qc_k=1,bio,batch    none,deseq,ruv_k=2,bio,no_batch
##                           24.14286                           23.14286
## none,deseq,ruv_k=2,no_bio,no_batch     none,deseq,qc_k=1,no_bio,batch
##                           23.00000                           22.92857
##      none,deseq,no_uv,no_bio,batch         none,deseq,no_uv,bio,batch
##                           19.71429                           19.71429
```

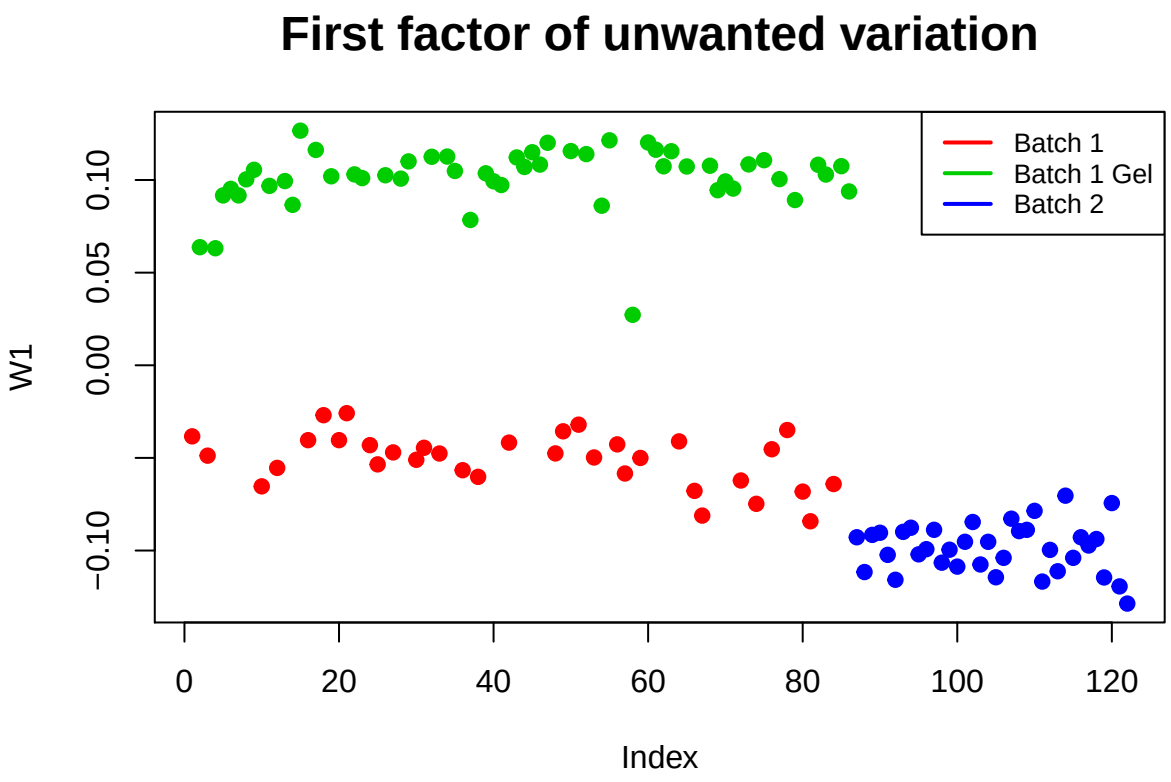
```
##       none,uq,ruv_k=2,bio,no_batch           none,uq,qc_k=1,bio,batch
##                           19.28571                           19.28571
##  none,deseq,qc_k=1,no_bio,no_batch       none,uq,qc_k=1,no_bio,batch
##                           19.07143                           19.07143
##     none,deseq,qc_k=1,bio,no_batch    none,uq,ruv_k=2,no_bio,no_batch
##                           18.71429                           18.14286
##      none,none,qc_k=1,no_bio,batch  none,none,ruv_k=2,no_bio,no_batch
##                           17.78571                           17.42857
##    none,deseq,ruv_k=1,bio,no_batch           none,uq,no_uv,bio,batch
##                           17.42857                           17.28571
##   none,deseq,no_uv,no_bio,no_batch none,deseq,ruv_k=1,no_bio,no_batch
##                           16.85714                           16.85714
##     none,none,ruv_k=2,bio,no_batch     none,uq,qc_k=1,no_bio,no_batch
##                           16.85714                           16.64286
```

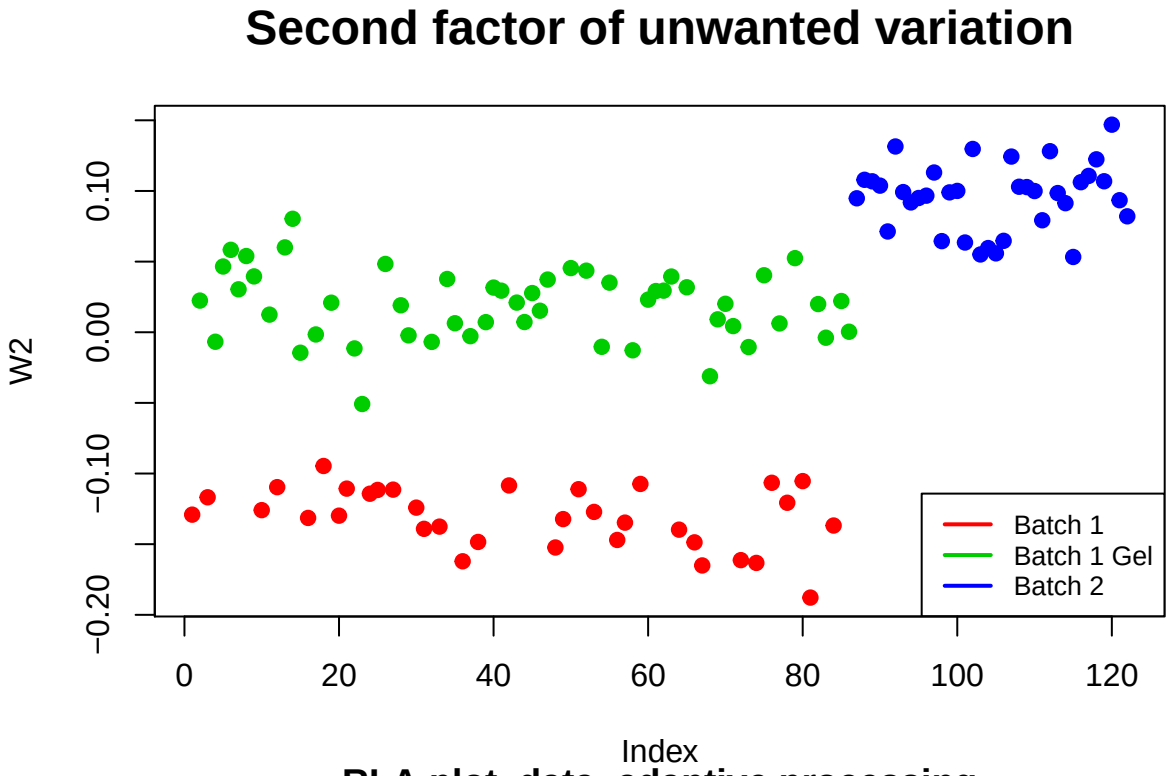
```
myscores <- get_scores(my_scone)
head(myscores)
```

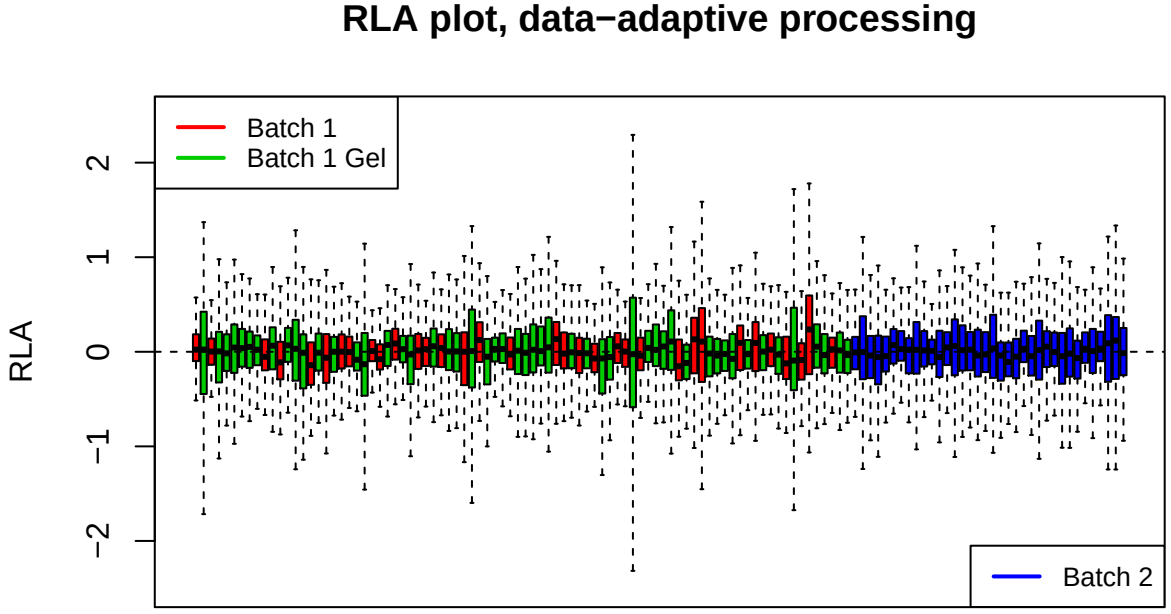
```
##                                      BIO_SIL  BATCH_SIL   PAM_SIL
## none,deseq,qc_k=1,bio,batch        0.01637112 0.06502668 0.2763547
## none,deseq,ruv_k=2,bio,no_batch    0.01797455 0.05353549 0.2644710
## none,deseq,ruv_k=2,no_bio,no_batch 0.01439908 0.05442142 0.2665655
## none,deseq,qc_k=1,no_bio,batch     0.01303499 0.06438318 0.2488139
## none,deseq,no_uv,no_bio,batch      0.01347112 0.06362549 0.2579243
## none,deseq,no_uv,bio,batch         0.01658801 0.06441018 0.2445258
##                                      EXP_QC_COR EXP_UV_COR      RLE_MED
## none,deseq,qc_k=1,bio,batch        -7.324949e-07 -0.2540555 -0.003726493
## none,deseq,ruv_k=2,bio,no_batch    -1.751028e-03 -0.2270815 -0.003936101
## none,deseq,ruv_k=2,no_bio,no_batch -1.730326e-03 -0.2269583 -0.003861331
## none,deseq,qc_k=1,no_bio,batch      2.220446e-16 -0.2517025 -0.003635965
## none,deseq,no_uv,no_bio,batch      -5.659256e-03 -0.2530492 -0.003630044
## none,deseq,no_uv,bio,batch         -5.670785e-03 -0.2555953 -0.003695859
##                                         RLE_IQR
## none,deseq,qc_k=1,bio,batch        -0.02136612
## none,deseq,ruv_k=2,bio,no_batch    -0.01553977
## none,deseq,ruv_k=2,no_bio,no_batch -0.01569106
## none,deseq,qc_k=1,no_bio,batch     -0.02168443
## none,deseq,no_uv,no_bio,batch      -0.02252552
## none,deseq,no_uv,bio,batch         -0.02247740
```

```
# Get the normalized data for the top ranked normalization
scaling <- c(deseq = DESEQ_FN)
qc <- as.matrix(as.numeric(c(c(1:86),c(1:36))))

k_ruv <- 0
k_qc <- 1


my_scone <- SconeExperiment(exp(as.matrix(peakTable3[,obsNames]))-1, bio =bio,qc=qc,
                            negcon_ruv = rownames(peakTable3) %in% empirical2,
                            batch=scone.batch)


my_scone <- scone(my_scone,scaling = scaling,
                  k_ruv = k_ruv,adjust_bio = "yes", k_qc = k_qc,
```

```
                  adjust_batch = "yes",
                  run=FALSE,verbose=TRUE)


is_screened <- (get_params(my_scone)$uv_factors %in% c('no_uv'))
my_scone = select_methods(my_scone,rownames(get_params(my_scone))[!is_screened])
#get_params(my_scone)[,-1]

is_screened <- (get_params(my_scone)$adjust_biology %in% c("no_bio"))
my_scone = select_methods(my_scone,rownames(get_params(my_scone))[!is_screened])
#get_params(my_scone)[,-1]

is_screened <- (get_params(my_scone)$adjust_batch %in% c("no_batch"))
my_scone = select_methods(my_scone,rownames(get_params(my_scone))[!is_screened])
#get_params(my_scone)[,-1]

my_scone <- scone(my_scone,scaling = scaling,k_ruv = k_ruv,adjust_bio = "yes", k_qc = k_qc,
                  adjust_batch = "yes",return_norm = "in_memory",
                  run=TRUE,verbose=TRUE,eval_kclust = 2:6)

scone.data <- get_normalized(my_scone,1,log=T)

plotRLE(exp(scone.data),outline=F,xaxt='n',col=as.numeric(scone.batch)+1,ylim=c(-2.5,2.5),
        main="RLA plot, data-adaptive processing",ylab='RLA')
legend('topleft',legend=c('Batch 1','Batch 1 Gel'),col=c(2,3),lwd=2,cex=.8)
legend('bottomright',legend=c('Batch 2'),col=4,lwd=2,cex=.8)
```







Metrics

The metrics used for evaluation of filtering and normalization procedures in Schiffman et al. are shown below. This includes the calculation of ranked p-values and fold changes, a PC and QQ-plot, and calculation of the RLA metrics.

```
## Calculate p-values for each feature DESEQ scaling function
DESEQ_FN = function(ei) {
    size_fac = calcNormFactors(ei, method = "RLE")
```

```
    scales = (colSums(ei) * size_fac)
    eo = t(t(ei) * mean(scales)/scales)
    return(eo)
}

deseq <- peakTable3[, obsNames]
deseq <- DESEQ_FN(exp(deseq))


crc.assoc <- foreach(j = 1:nrow(deseq), .combine = "rbind") %do% {
    lm.fit <- lm(log1p(as.numeric(deseq[j, obsNames])) ~ bio + scone.batch +
        as.factor(covars$gender) + covars$age + as.numeric(qc))
    return(2 * (1 - pnorm(abs(summary(lm.fit)[[4]][2, 1]/sqrt(hccm(lm.fit)[2,
        2])))))
}

names(crc.assoc) <- rownames(deseq)

ulfca <- c("6786", "6146", "6195", "6692", "7487", "7531", "8755", "10329")

sort(crc.assoc[names(crc.assoc) %in% ulfca])
```

```
##        6786        8755        6146       10329
## 0.001443418 0.008698714 0.015685246 0.329001440
```

```
which(names(sort(crc.assoc)) %in% ulfca)
```

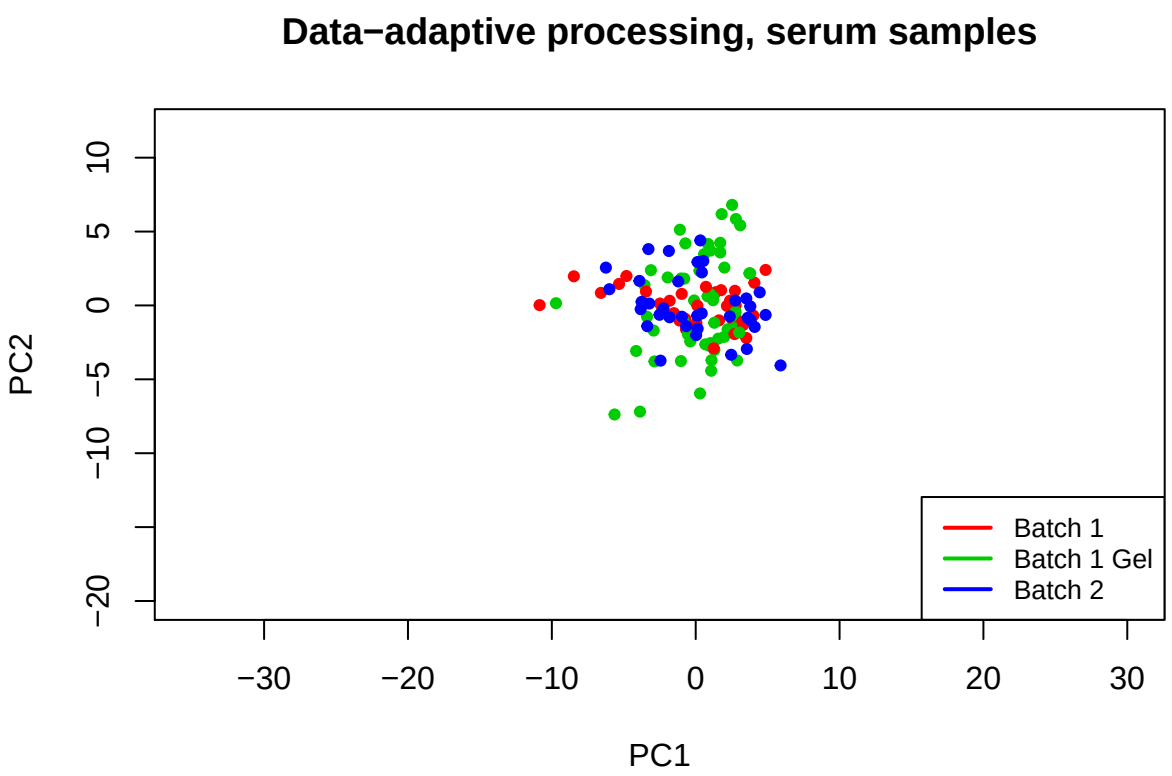
```
## [1]   2  12  22 187
```

```
FC <- foreach(j = 1:nrow(deseq), .combine = "rbind") %do% {
    lm.fit <- lm(log1p(as.numeric(deseq[j, obsNames])) ~ bio + scone.batch +
        as.factor(covars$gender) + covars$age + as.numeric(qc))
    return(summary(lm.fit)[[4]][2, 1])
}

names(FC) <- rownames(deseq)

FC <- exp(FC)

FC[names(FC) %in% ulfca]
```

```
##      6146      6786      8755     10329
## 0.7483079 0.7273324 0.7916024 0.9032472
```

```
### calculating RLA metrics

control <- scone.data[, bio == 0]
control.rle <- t(apply(exp(control), 1, function(x) log(x/median(x))))
med.control.rle <- apply(control.rle, 2, median)

med.control.rle.scone <- -mean(med.control.rle^2)
iqr.control.rle.scone <- -var(apply(control.rle, 2, IQR))

### bio==1

case <- scone.data[, bio == 1]
```

```
case.rle <- t(apply(exp(case), 1, function(x) log(x/median(x))))
med.case.rle <- apply(case.rle, 2, median)

med.case.rle.scone <- -mean(med.case.rle^2)
iqr.case.rle.scone <- -var(apply(case.rle, 2, IQR))


## visualize removal of unwanted variation
pca <- prcomp(t(scone.data))
plot(pca$x[, 1], pca$x[, 2], pch = 19, col = as.numeric(scone.batch) + 1, main = "Data-adaptive processi
    xlab = "PC1", ylab = "PC2", cex = 0.7, xlim = c(-35, 30), ylim = c(-20,
        12))
legend("bottomright", legend = c("Batch 1", "Batch 1 Gel", "Batch 2"), col = c(2:4),
    lwd = 2, cex = 0.8)
```




```
# visualize distribution of p-values
library(EnvStats)
qqPlot(crc.assoc, distribution = "unif", param.list = list(min = 0, max = 1),
    main = "Data-adaptive processing, Serum Samples", ylab = "Quantiles of P-values",
    cex = 0.8)
abline(0, 1, lwd = 2, col = "red")
```