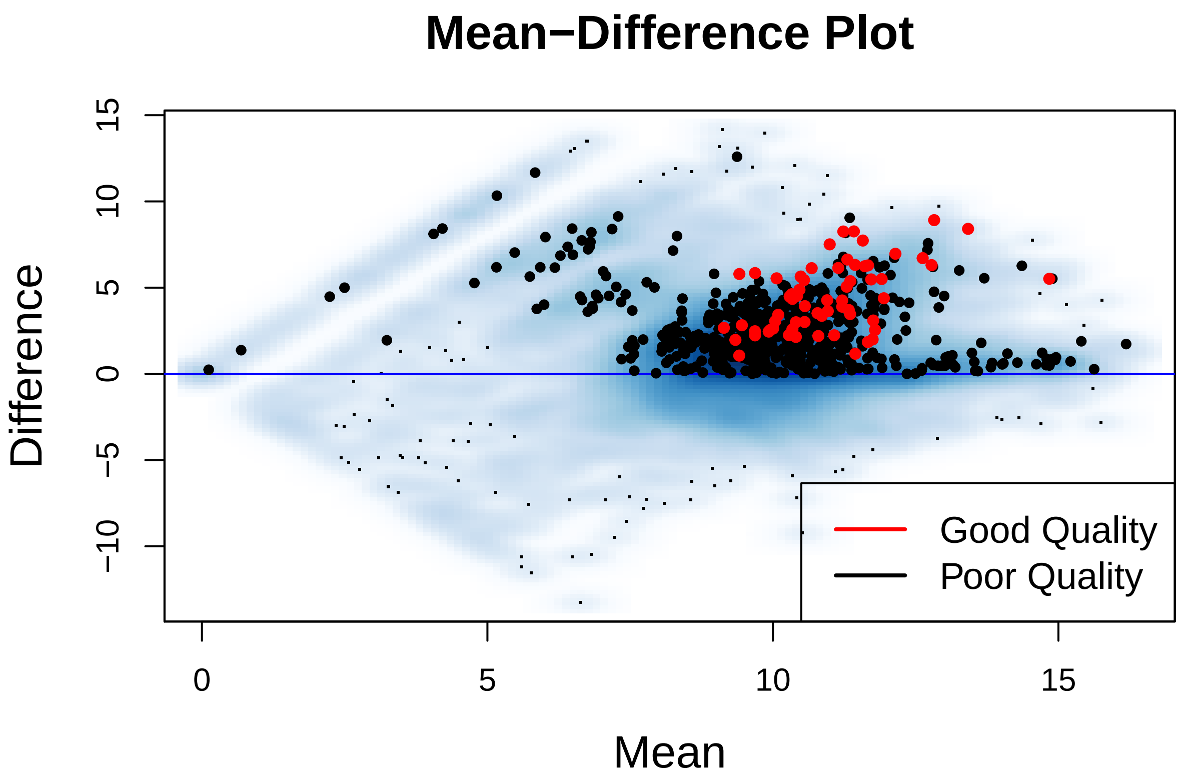
Data-adaptive Filtering Example

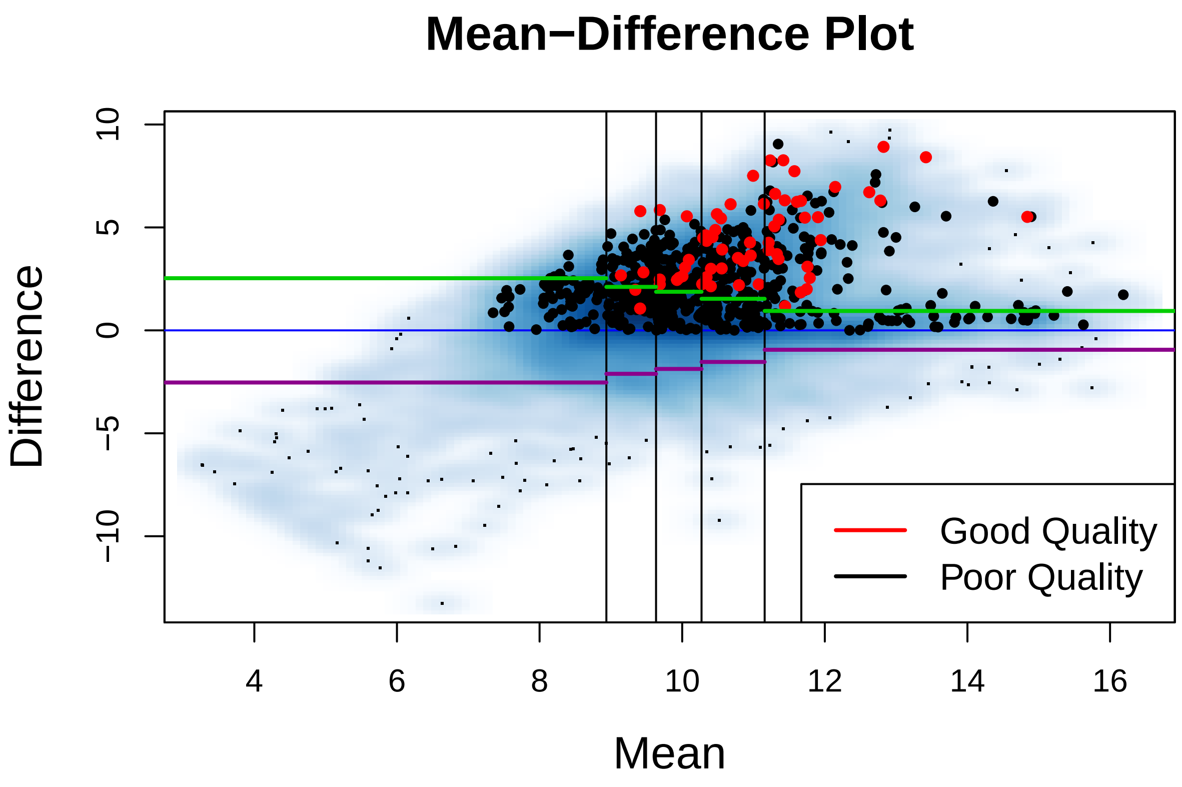
```r
libs <- c("xcms", "RColorBrewer", "tidyverse", "doParallel",
    "foreach", "parallel")
for (l in libs) {
    suppressPackageStartupMessages(library(l, character.only = T))
}
```

MD-plot filtering

```r
source("MDplot.R")

load("Data.RData")

MDplot(filled, blankNames, obsNames, high, low)


# Calculate the number of blank samples (0-3) each feature
# has a zero value in. Most features are detected in all 3
# blank samples (num.zero=0)
num.zero <- apply(filled[, blankNames], 1, function(x) sum(x ==
    0))
names(num.zero) <- rownames(filled)

filled <- filled[num.zero == 0, ]
unfilled <- unfilled[num.zero == 0, ]

obsMean <- as.vector(apply(filled[, obsNames], 1, mean))
names(obsMean) <- rownames(filled)
blankMean <- apply(filled[, blankNames], 1, mean)
names(blankMean) <- rownames(filled)

# quantiles to partition the features along the x-axis
quantiles <- c(0.2, 0.4, 0.6, 0.8, 1)
breaks <- quantile((blankMean + obsMean)/2, quantiles)

# difference in average log abundances between biological and
# blank samples
diff <- (obsMean) - (blankMean)
# average log abundances in biological and blank samples
mean <- (blankMean + obsMean)/2

# find features in each partition above the absolute value of
# the lower quartile of differences below the zero-difference
# line in each partition

less1 <- diff[diff < 0 & mean <= breaks[1]]
bin1 <- rownames(filled)[diff > 0 & mean <= breaks[1] & diff >
    abs(summary(less1)[2])]

less2 <- diff[diff < 0 & mean <= breaks[2] & mean > breaks[1]]
```

```r
bin2 <- rownames(filled)[diff > 0 & mean <= breaks[2] & mean >
    breaks[1] & diff > abs(summary(less2))[2]]

less3 <- diff[diff < 0 & mean <= breaks[3] & mean > breaks[2]]
bin3 <- rownames(filled)[diff > 0 & mean <= breaks[3] & mean >
    breaks[2] & diff > abs(summary(less3)[2])]

less4 <- diff[diff < 0 & mean <= breaks[4] & mean > breaks[3]]
bin4 <- rownames(filled)[diff > 0 & mean <= breaks[4] & mean >
    breaks[3] & diff > abs(summary(less4)[2])]

less5 <- diff[diff < 0 & mean > breaks[4]]
bin5 <- rownames(filled)[diff > 0 & mean > breaks[4] & diff >
    abs(summary(less5)[2])]


# plot the paritions and filtering cutoffs for this cluster

smoothScatter((blankMean + obsMean)/2, (obsMean) - (blankMean),
    xlab = "Mean", ylab = "Difference", main = "Mean-Difference Plot",
    cex.lab = 1.4, cex.main = 1.5)
abline(h = 0, lwd = 1, col = "blue")
legend("bottomright", legend = c("Good Quality", "Poor Quality"),
    col = c("red", "black"), lwd = 2, cex = 1.2)

blanks.low <- blankMean[names(blankMean) %in% low]
obs.low <- (obsMean)[names(obsMean) %in% low]
points((blanks.low + obs.low)/2, obs.low - blanks.low, pch = 19,
    cex = 0.6)
blanks.high <- blankMean[names(blankMean) %in% high]
obs.high <- obsMean[names(obsMean) %in% high]
points((blanks.high + obs.high)/2, obs.high - blanks.high, col = "red",
    pch = 19, cex = 0.7)

abline(v = breaks[1], lwd = 1)
abline(v = breaks[2], lwd = 1)
abline(v = breaks[3], lwd = 1)
abline(v = breaks[4], lwd = 1)
segments(x0 = -1, y0 = summary(less1)[2], x1 = breaks[1], col = "darkmagenta",
    lwd = 2)
segments(x0 = -1, y0 = abs(summary(less1)[2]), x1 = breaks[1],
    col = "green3", lwd = 2)
segments(x0 = breaks[1], y0 = summary(less2)[2], x1 = breaks[2],
    col = "darkmagenta", 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 = "darkmagenta", 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 = "darkmagenta", 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 = "darkmagenta",
    lwd = 2)
segments(x0 = breaks[4], y0 = abs(summary(less5)[2]), x1 = 20,
    col = "green3", lwd = 2)


# features that are retained

batch1.features <- c(bin1, bin2, bin3, bin4, bin5)


filled <- filled[rownames(filled) %in% batch1.features, ]
unfilled <- unfilled[rownames(unfilled) %in% batch1.features,
    ]
```





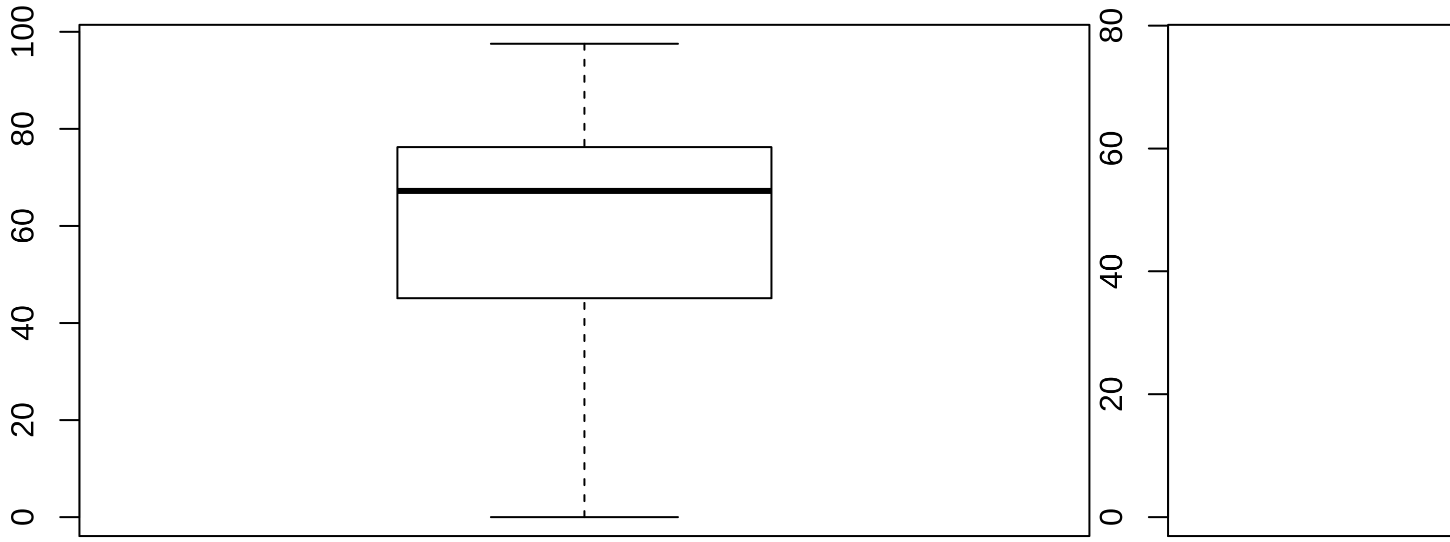
Percent missing filtering

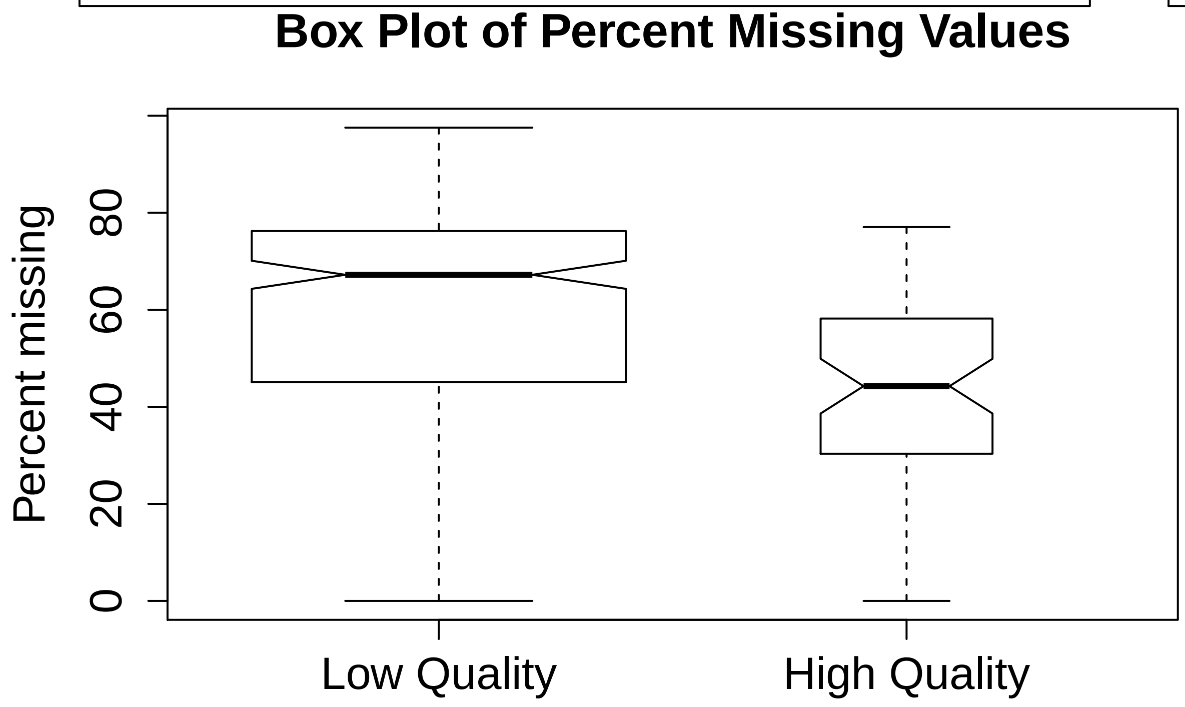
```
source("Percent_missing.R")

boxplot.na(unfilled, c(obsNames, obsNames2), high, low)
```

```
## $LowQuality
##           [,1]
## [1,]  0.00000
## [2,] 45.08197
## [3,] 67.21311
## [4,] 76.22951
## [5,] 97.54098
##
## $HighQuality
##           [,1]
## [1,]  0.00000
## [2,] 30.32787
## [3,] 44.26230
## [4,] 58.19672
## [5,] 77.04918
```

```
fish.pvals <- diff.miss.fish(unfilled, obsNames, bio)

na.threshold <- 0.67

pval.thresh <- 0.01
```

```
keep.features <- unique(c(filter.na(na.threshold, unfilled, c(obsNames,
    obsNames2)), filter.fish(pval.thresh, fish.pvals)))

filled <- filled[rownames(filled) %in% keep.features, ]
unfilled <- unfilled[rownames(unfilled) %in% keep.features, ]
```





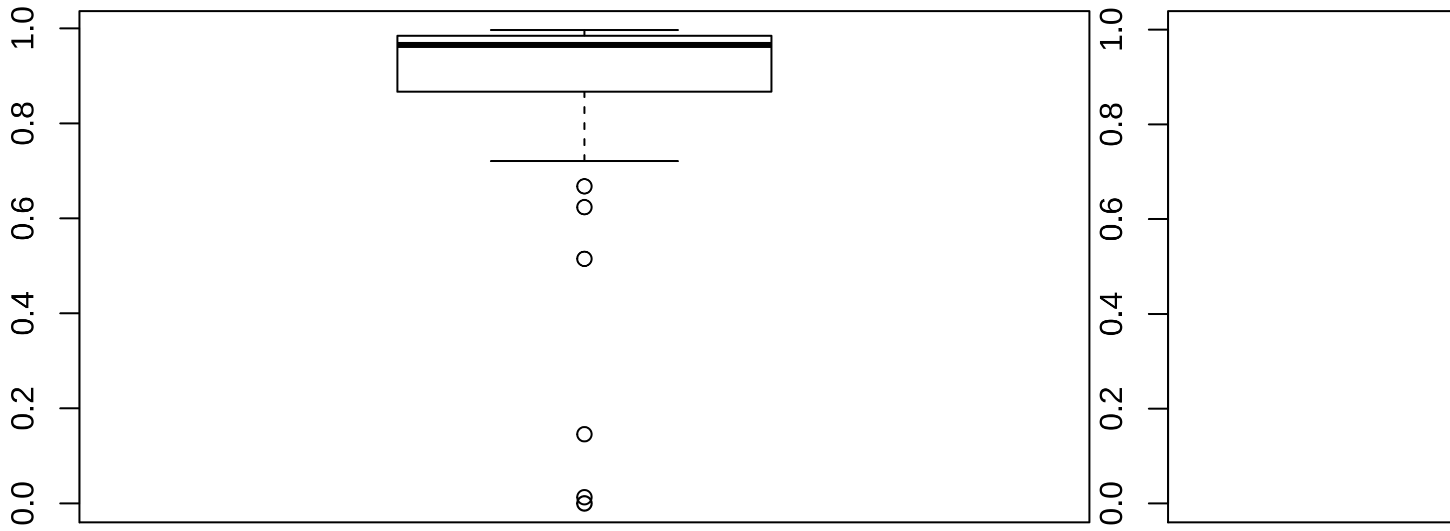
ICC filtering

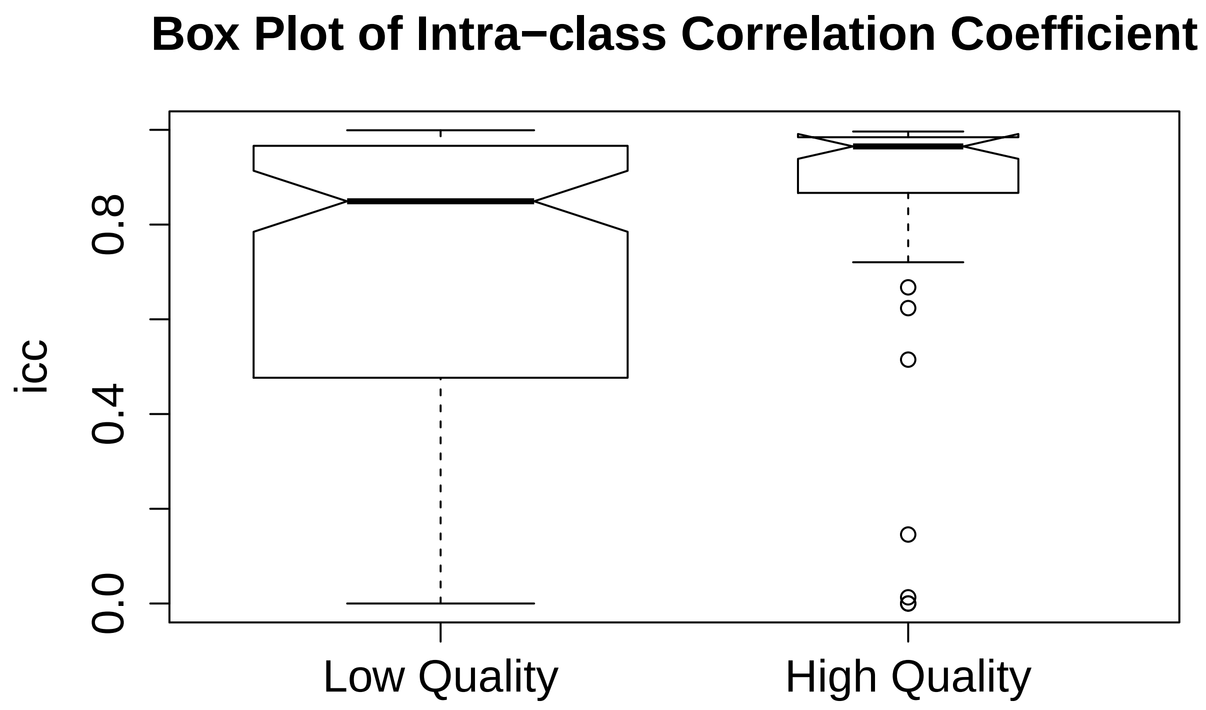
```
source("ICC.R")
registerDoParallel(cores = detectCores())

ICC <- calc.icc(filled, obsNames, qcNames)
```

```
boxplot.icc(ICC, high, low)
```

```
## $LowQuality
##              [,1]
## [1,] 3.430397e-10
## [2,] 4.766144e-01
## [3,] 8.492017e-01
## [4,] 9.662846e-01
## [5,] 9.991053e-01
##
## $HighQuality
##           [,1]
## [1,] 0.7204488
## [2,] 0.8668857
## [3,] 0.9650184
## [4,] 0.9844072
## [5,] 0.9964607
```

```
threshold <- 0.72
filled <- filled[rownames(filled) %in% filter.icc(ICC, threshold),
    ]
unfilled <- unfilled[rownames(unfilled) %in% filter.icc(ICC,
    threshold), ]
```




Box Plot of Intra−class Correlation Coefficient
icc
0.0
0.4
0.8
Low Quality
High Quality