Data-adaptive Filtering Example

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

MD-plot filtering

```
# log abundances from xcms pre-processing, without having
# used 'fillChromPeaks'
load("peakTable.RData")

# log abundances from xcms pre-processing after using
# 'fillChromPeaks'
load("filled_data.RData")

# CRC case status
load("cov.RData")

# list of good quality peaks, named 'pass'
load("good_quality_features.RData")

# list of poor quality peaks, names 'fail'
load("poor_quality_features.RData")

# Pooled QC names
qcNames1 <- colnames(peakTable)[grep("LocalQC", colnames(peakTable))]
qcNames1 <- qcNames1[grep("reinject", qcNames1)]

# Blank sample names
filterNames1 <- colnames(peakTable)[grep("_Blank", colnames(peakTable))]
filterNames1 <- filterNames1[grep("reinject", filterNames1)]

# Biological sample names. We consider here only the
# 'reinject' samples. The other samples we contamintaed by a
# gelled substance. Although all samples were pre-processed
# together in 'xcms'
obsNames1 <- colnames(peakTable)[grep("reinject_Sample", colnames(peakTable))]


# Mean log abundance of each feature across biological
# samples
obsMean1 <- as.vector(apply(filled[, obsNames1], 1, mean))
names(obsMean1) <- rownames(filled)

# Mean log abundance of each feature across blank samples
filterMean1 <- apply(filled[, filterNames1], 1, mean)
names(filterMean1) <- rownames(filled)
```

```
# 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[, filterNames1], 1, function(x) sum(x ==
    0))
names(num.zero) <- rownames(peakTable)


# start by filtering only features that are detected
# (non-zero) in all 3 blank samples zero.filt <-
# apply(filled[,filterNames1],1, function(x) 0%in%x)
line1 <- rownames(filled)[num.zero != 0]

# quantiles to partition the features along the x-axis
quantiles <- c(0.2, 0.4, 0.6, 0.8, 1)
breaks <- quantile(((filterMean1[num.zero == 0]) + (obsMean1[num.zero ==
    0]))/2, quantiles)

# difference in average log abundances between biological and
# blank samples
diff1 <- (obsMean1) - (filterMean1)
# average log abundances in biological and blank samples
mean1 <- ((filterMean1) + (obsMean1))/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 <- diff1[!rownames(filled) %in% line1 & diff1 < 0 & mean1 <=
    breaks[1]]
bin1 <- rownames(filled)[diff1 > 0 & mean1 <= breaks[1] & !rownames(filled) %in%
    line1 & diff1 > abs(summary(less1)[2])]

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

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

less4 <- diff1[!rownames(filled) %in% line1 & diff1 < 0 & mean1 <=
    breaks[4] & mean1 > breaks[3]]
bin4 <- rownames(filled)[diff1 > 0 & mean1 <= breaks[4] & mean1 >
    breaks[3] & !rownames(filled) %in% line1 & diff1 > abs(summary(less4)[2])]

less5 <- diff1[!rownames(filled) %in% line1 & diff1 < 0 & mean1 >
    breaks[4]]
bin5 <- rownames(filled)[diff1 > 0 & mean1 > breaks[4] & !rownames(filled) %in%
    line1 & diff1 > abs(summary(less5)[2])]

# create the mean-difference plot
```

```
smoothScatter(((filterMean1) + (obsMean1))/2, (obsMean1) - (filterMean1),
    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)

# plot the good and poor quality features
inds <- diff1[names(diff1) %in% fail] > 0

blanks1.rem <- (filterMean1)[names(filterMean1) %in% c(fail)]
obs1.rem <- (obsMean1)[names(obsMean1) %in% c(fail)]

points(((blanks1.rem)[inds] + (obs1.rem)[inds])/2, (obs1.rem)[inds] -
    (blanks1.rem)[inds], pch = 19, cex = 0.6)

inds <- diff1[names(diff1) %in% pass] > 0
blanks1.pass <- (filterMean1)[names(filterMean1) %in% c(pass)]
obs1.pass <- (obsMean1)[names(obsMean1) %in% c(pass)]

points(((blanks1.pass)[inds] + (obs1.pass)[inds])/2, (obs1.pass)[inds] -
    (blanks1.pass)[inds], col = "red", pch = 19, cex = 0.7)

# plot the cluster of features corresponding to features
# detected in all 3 blank samples
smoothScatter(((filterMean1)[num.zero == 0] + (obsMean1)[num.zero ==
    0])/2, (obsMean1)[num.zero == 0] - (filterMean1)[num.zero ==
    0], xlab = "Mean", ylab = "Difference", main = "Mean-Difference Plot",
    cex.lab = 1.4, cex.main = 1.5)


inds <- diff1[names(diff1) %in% fail] > 0 & num.zero[names(num.zero) %in%
    fail] == 0

blanks1.rem <- (filterMean1)[names(filterMean1) %in% c(fail)]
obs1.rem <- (obsMean1)[names(obsMean1) %in% c(fail)]

points(((blanks1.rem)[inds] + (obs1.rem)[inds])/2, (obs1.rem)[inds] -
    (blanks1.rem)[inds], pch = 19, cex = 0.6)

inds <- diff1[names(diff1) %in% pass] > 0 & num.zero[names(num.zero) %in%
    pass] == 0
blanks1.pass <- (filterMean1)[names(filterMean1) %in% c(pass)]
obs1.pass <- (obsMean1)[names(obsMean1) %in% c(pass)]

points(((blanks1.pass)[inds] + (obs1.pass)[inds])/2, (obs1.pass)[inds] -
    (blanks1.pass)[inds], col = "red", pch = 19, cex = 0.6)

# plot the paritions and filtering cutoffs for this cluster

abline(h = 0, lwd = 1, col = "blue")
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)

legend("bottomright", legend = c("Good Quality", "Poor Quality"),
    col = c("red", "black"), lwd = 2, cex = 1.2)

summary(diff1[diff1 < 0 & num.zero == 3])
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##
```

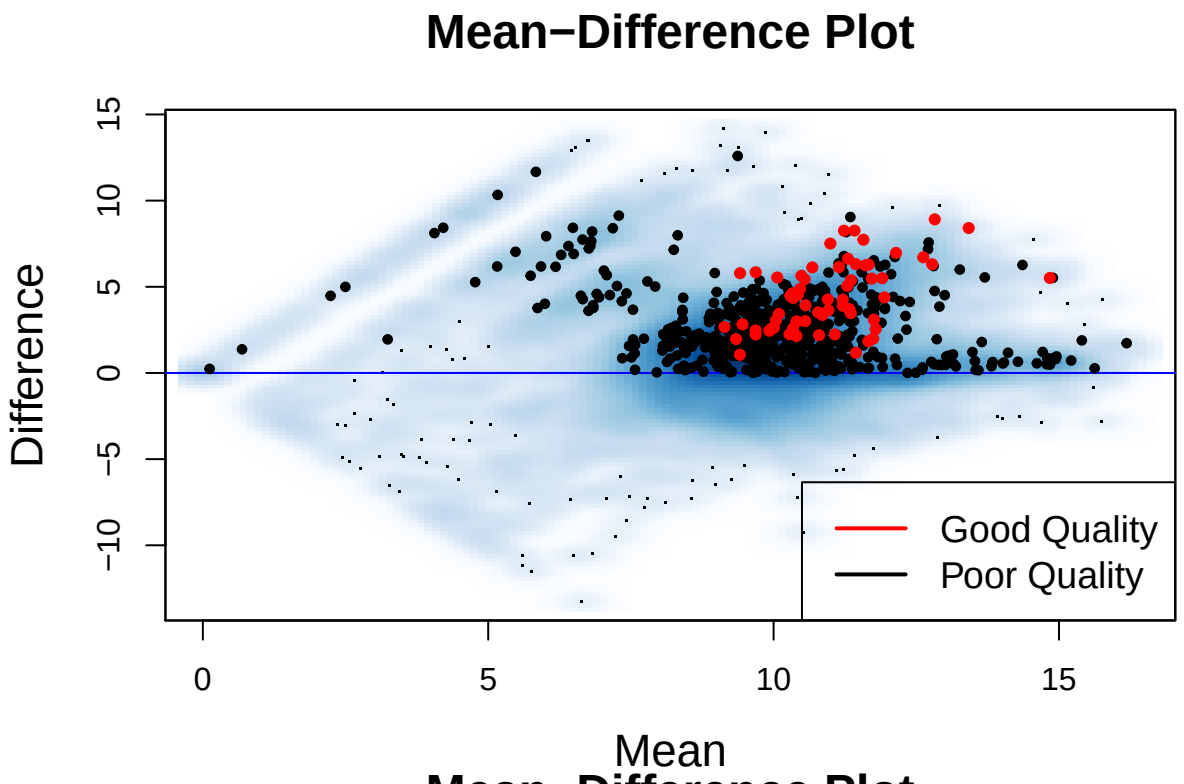
```
summary(diff1[diff1 < 0 & num.zero == 2])
```

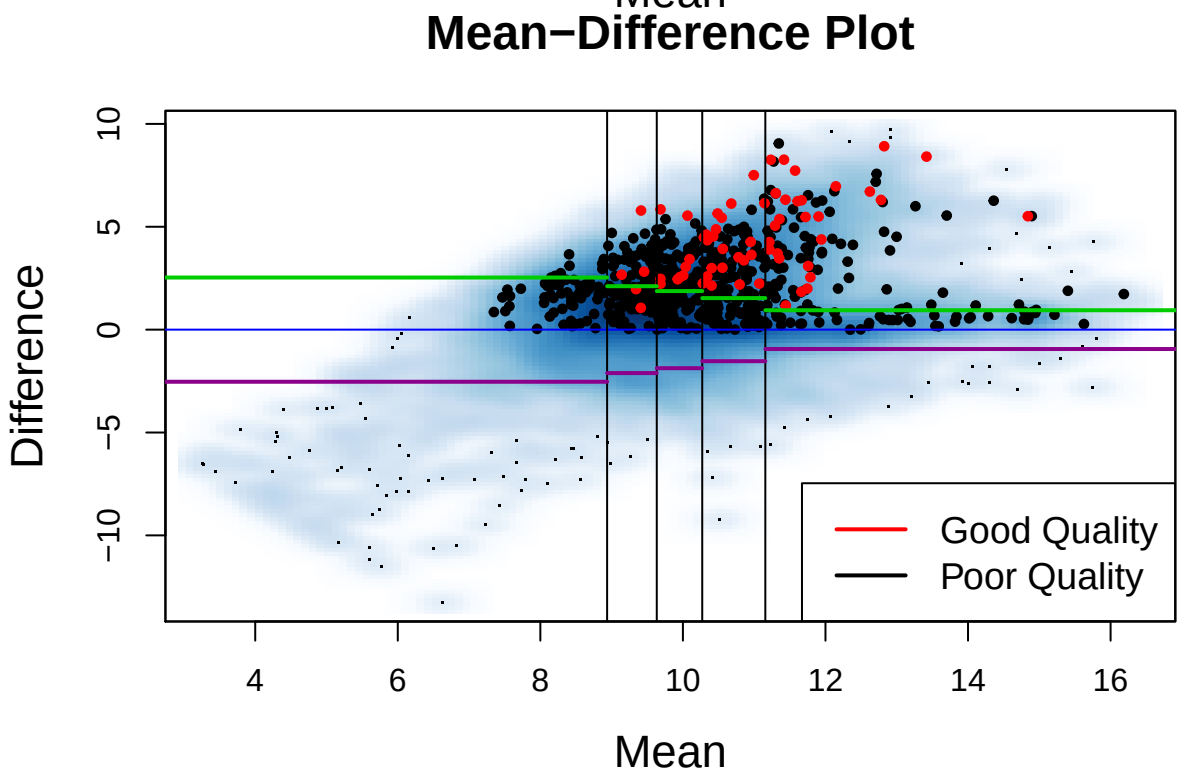
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -3.556  -2.805  -1.897  -1.951  -1.289  -0.219
```

```
summary(diff1[diff1 < 0 & num.zero == 1])
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -5.52869 -3.47702 -2.17160 -2.29853 -0.93319 -0.01158
```

```
line1 <- rownames(filled)[num.zero == 3 | (num.zero == 2 & diff1 >
    2.805) | (num.zero == 1 & diff1 > 3.47702)]

# features that are retained

batch1.features <- c(line1, bin1, bin2, bin3, bin4, bin5)  # 8006



peakTable2 <- peakTable[rownames(peakTable) %in% batch1.features,
    ]
filled2 <- filled[rownames(filled) %in% batch1.features, ]
```





Percent missing filtering

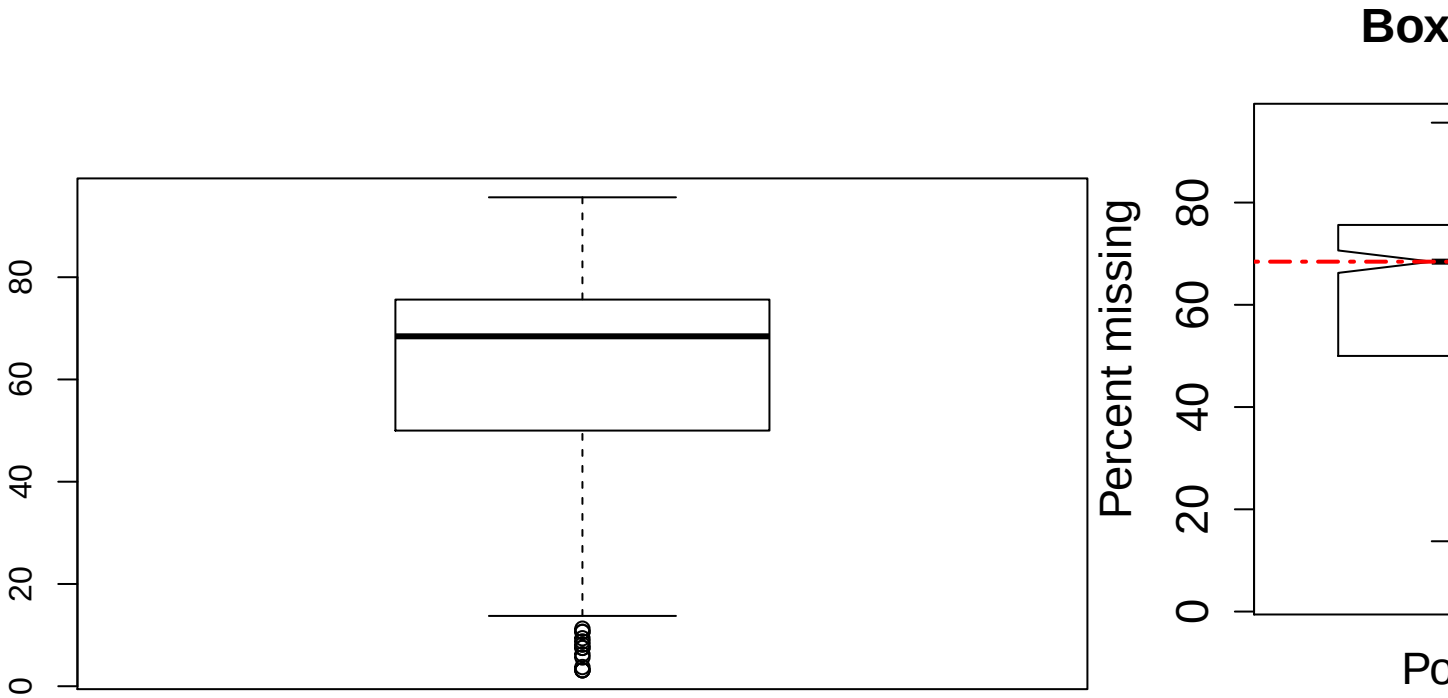
```
# calculate percent missing for each feature across the
# peakTable
```

```
percent.zero1 <- apply(peakTable2[, 12:171], 1, function(x) mean(is.na(x)))
names(percent.zero1) <- rownames(peakTable2)

# look at the box plot statistics to explore possible
# filtering cutoffs
bp <- boxplot(percent.zero1[names(percent.zero1) %in% c(fail)] *
    100)

bp$stats
```

```
##         [,1]
## [1,] 13.7500
## [2,] 50.0000
## [3,] 68.4375
## [4,] 75.6250
## [5,] 95.6250
```

```
boxplot(percent.zero1[names(percent.zero1) %in% c(pass, fail)] *
    100 ~ as.factor(names(percent.zero1[names(percent.zero1) %in%
    c(pass, fail)]) %in% pass), notch = T, names = c("Poor Quality",
    "Good Quality"), main = "Box Plot of Percent Missing Values",
    cex.lab = 1.4, cex.main = 1.5, ylab = "Percent missing",
    cex.axis = 1.4, varwidth = T)

# We use the median of the poor quality features
abline(h = bp$stats[3, ], lty = 6, col = "red", lwd = 2)


## calculate p-value for fisher exact test to evaluate
## dependence between case-control status and
## missing/non-missing for each feature
percent.zero <- apply(peakTable2[, obsNames1], 1, function(x) mean(is.na(x)))

fish.pvals <- c()

for (i in 1:nrow(peakTable2)) {
    if (percent.zero[i] == 0 | percent.zero[i] == 1) {
        fish.pvals[i] <- 1
    } else {
        fish.pvals[i] <- fisher.test(as.numeric(is.na(peakTable2[i,
            obsNames1])), cov$Case)$p.value
    }
}

# keep features below the percent missing cutoff or with
# fisher exact p-values less than the hundredth percentile of
# p-values.
keep.features2 <- fish.pvals <= quantile(fish.pvals, 0.01) |
    (percent.zero1 <= bp$stats[3, ]/100)


filled2 <- filled2[keep.features2, ]

peakTable2 <- peakTable2[keep.features2, ]
```

```r
feature.info <- filled2[, 1:12]

filled2 <- filled2[, -c(1:12)]

peakTable2 <- peakTable2[, -c(1:12)]


# matrix of quality control samples
qc.matrix <- filled2[, c(qcNames1)]

filled3 <- filled2[, obsNames1]

peakTable3 <- peakTable2[, obsNames1]

filled3 <- as.matrix(filled3)
peakTable3 <- as.matrix(peakTable3)


filled3[filled3 == 0] <- NA

qc.matrix[qc.matrix == 0] <- NA

# remove a few features with a lot of missing values after
# using 'fillChromPeaks'

percent.zero.filled <- apply(filled3[, obsNames1], 1, function(x) mean(is.na(x)))
names(percent.zero.filled) <- rownames(filled2)

filled3 <- filled3[percent.zero.filled <= 0.05, ]
qc.matrix <- qc.matrix[percent.zero.filled <= 0.05, ]
```



ICC filtering

```
registerDoParallel(cores = 4)

# ICC estimation

vars1 <- foreach(i = (1:nrow(filled3))[-c(1800, 4134)], .packages = "nlme",
    .combine = "rbind") %dopar% {
    reps <- factor(c(1:length(obsNames1), rep(length(obsNames1) +
        1, length(qcNames1))))
    data <- data.frame(y = c(as.numeric(filled3[i, ]), as.numeric(qc.matrix[i,
        ])), reps = reps)
    mm <- lme(y ~ 1, random = ~1 | reps, data = data, na.action = na.omit)
    return(as.numeric(VarCorr(mm)[1:2]))
}


ICC1 <- apply(vars1, 1, function(x) x[1]/sum(x))
names(ICC1) <- rownames(filled3)[-c(1800, 4134)]

# CV estimation for comparison

myCV <- foreach(i = (1:nrow(filled3))[-c(1800, 4134)], .packages = c("nlme",
    "sjstats"), .combine = "rbind") %dopar% {
    reps <- factor(c(1:length(obsNames1), rep(length(obsNames1) +
        1, length(qcNames1))))
    data <- data.frame(y = c(as.numeric(filled3[i, ]), as.numeric(qc.matrix[i,
        ])), reps = reps)
    mm <- lme(y ~ 1, random = ~1 | reps, data = data, na.action = na.omit)
    return(cv(mm))
```

```
}

CV1 <- myCV * 100
names(CV1) <- rownames(filled3)[-c(1800, 4134)]


# View box plot statistics for appropriate cutoffs

bp <- boxplot(ICC1[names(ICC1) %in% c(pass)])

bp$stats
```

```
##           [,1]
## [1,] 0.7204488
## [2,] 0.8668857
## [3,] 0.9633082
## [4,] 0.9844072
## [5,] 0.9969657
```

```
boxplot(ICC1[names(ICC1) %in% c(pass, fail)] ~ as.factor(names(ICC1[names(ICC1) %in%
    c(pass, fail)]) %in% pass), notch = T, names = c("Poor Quality",
    "Good Quality"), main = "Box Plot of Intra-class Correlation Coefficient",
    cex.lab = 1.4, cex.main = 1.5, ylab = "ICC", cex.axis = 1.4,
    varwidth = T)


abline(h = bp$stats[1, ], lty = 6, col = "red", lwd = 2)


# View CV distributions for comparison

boxplot(CV1[names(CV1) %in% c(pass, fail)] ~ as.factor(names(CV1[names(CV1) %in%
    c(pass, fail)]) %in% pass), notch = T, names = c("Poor Quality",
    "Good Quality"), main = "Box Plot of Coefficient of Variation",
    cex.lab = 1.4, cex.main = 1.5, ylab = "CV", cex.axis = 1.4,
    varwidth = T, ylim = c(0, 35))

abline(h = 30, lty = 6, lwd = 2)

legend("topright", legend = c("Typical CV cutoff"), lwd = 2,
    col = "black", lty = 6, cex = 1.5)

sum(CV1[names(CV1) %in% c(fail)] < 30)
```

```
## [1] 170
```

```
filled4 <- filled3[-c(1800, 4134), ]

keep.features5 <- ICC1 >= bp$stats[1, ]

filled4 <- filled4[keep.features5, ]

sum(fail %in% rownames(filled4))
```

```
## [1] 105
```

```r
sum(pass %in% rownames(filled4))
```

```
## [1] 46
```