Sensitivity and precision using the Bottomly et al. dataset

Michael Love

October 14, 2014

1 Load the benchmarking results

We load the benchmarking results, which were produced by the script `/inst/script/bottomlyDiffExpr.R`. The *SummarizedExperiment* object used for this analysis is contained in `/data/bottomly_sumexp.RData`.

```
library("DESeq2paper")
data("sensitivityPrecision")
```

The evaluation set results are contained in the `resTest` object and the verification set results are contained in the `resHeldout` object, each a list, one element for each random replicate, of data frames which contain a column for each algorithm giving the adjusted p-values for each gene. For p-value adjustment, the *p.adjust* function was used with `method="BH"` (Benjamini-Hochberg correction), over only those genes with non-zero row sum.

```
library("ggplot2")
library("reshape")

##
## Attaching package:  'reshape'
##
## The following objects are masked from 'package:IRanges':
##
##    expand, rename

getHeldoutCalls <- function(alpha) {
    t(sapply(1:nreps, function(i) sapply(namesAlgos, function(algo) {
        sum((resHeldout[[i]][[algo]] < alpha))
    })))
}
getTestCalls <- function(alpha) {
    t(sapply(1:nreps, function(i) sapply(namesAlgos, function(algo) {
        sum((resTest[[i]][[algo]] < alpha))
    })))
}

getSensitivityAlgoGold <- function(alpha, alphaOut, gold) {
    t(sapply(1:nreps, function(i) sapply(namesAlgos, function(algo) {
        sigHeldout <- resHeldout[[i]][[gold]] < alphaOut
        mean((resTest[[i]][[algo]] < alpha)[sigHeldout])
    })))
}

getPrecisionAlgoGold <- function(alpha, alphaOut, gold) {
    t(sapply(1:nreps, function(i) sapply(namesAlgos, function(algo) {
        sigTest <- resTest[[i]][[algo]] < alpha
        if (sum(sigTest) == 0)
            return(0)
        mean((resHeldout[[i]][[gold]] < alphaOut)[sigTest])
    })))
}
```

The following function helps to rename algorithms.

```
renameAtoB <- function(f, a, b) {
    levels(f)[levels(f) == a] <- b
    f
}
```

```
namesAlgos <- make.names(namesAlgos)
names(namesAlgos) <- namesAlgos
```

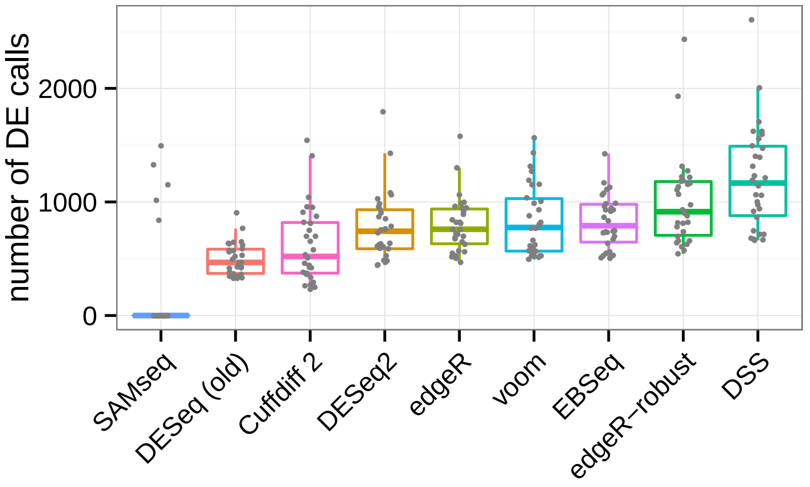


Figure 1: Evaluation set calls (adjusted p-value $< .1$)

2 Counting number of calls

Here we produce boxplots of the number of calls based on adjusted p-value for each algorithm in the evaluation set and verification set for each random replicate.

```
nreps <- length(resTest)
heldMat <- getHeldoutCalls(0.1)
testMat <- getTestCalls(0.1)
d <- data.frame(heldoutCalls = as.vector(heldMat), testCalls = as.vector(testMat),
    algorithm = factor(rep(namesAlgos, each = nrow(heldMat)), levels = namesAlgos))
d$algorithm <- renameAtoB(d$algorithm, "DESeq", "DESeq (old)")
d$algorithm <- renameAtoB(d$algorithm, "cuffdiff2", "Cuffdiff 2")
d$algorithm <- renameAtoB(d$algorithm, "edgeR.robust", "edgeR-robust")
```

```
p <- ggplot(d, aes(x = reorder(algorithm, testCalls, median), y = testCalls,
    color = algorithm))
p + geom_boxplot(outlier.colour = rgb(0, 0, 0, 0)) + theme_bw() + geom_point(position = position_jitter(w
    h = 0), color = "grey50", size = 1) + theme(axis.text.x = element_text(angle = 45,
    hjust = 1)) + xlab("") + scale_colour_discrete(guide = "none") + ylab("number of DE calls")
```

```
p <- ggplot(d, aes(x = reorder(algorithm, heldoutCalls, median), y = heldoutCalls,
    color = algorithm))
p + geom_boxplot(outlier.colour = rgb(0, 0, 0, 0)) + theme_bw() + geom_point(position = position_jitter(w
    h = 0), color = "grey50", size = 1) + theme(axis.text.x = element_text(angle = 45,
    hjust = 1)) + xlab("") + scale_colour_discrete(guide = "none") + ylab("number of DE calls")
```

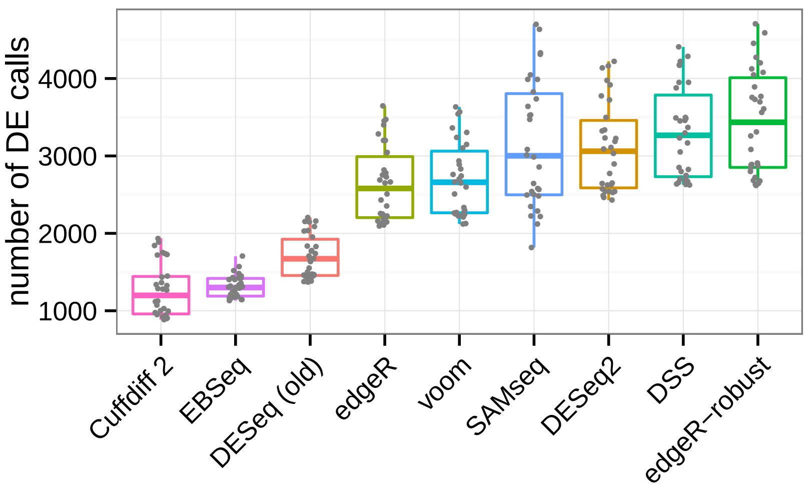


Figure 2: Verification set calls (adjusted p-value $< .1$)

3 Sensitivity and precision plots

We construct a data frame containing the sensitivity and precision estimates for every pair of algorithm in the evaluation set and the verification set.

```r
nreps <- length(resTest)
sensMat <- do.call(rbind, lapply(namesAlgos, function(algo) {
    res <- getSensitivityAlgoGold(0.1, 0.1, algo)
    data.frame(res, heldout = rep(algo, nrow(res)))
}))
sensMelt <- melt(sensMat, id = "heldout")
names(sensMelt) <- c("heldout", "test", "sensitivity")
names(sensMelt) <- c("verification", "evaluation", "sensitivity")

precMat <- do.call(rbind, lapply(namesAlgos, function(algo) {
    res <- getPrecisionAlgoGold(0.1, 0.1, algo)
    data.frame(res, heldout = rep(algo, nrow(res)))
}))
precMelt <- melt(precMat, id = "heldout")
names(precMelt) <- c("heldout", "test", "precision")
names(precMelt) <- c("verification", "evaluation", "precision")

d <- data.frame(sensMelt, precision = precMelt$precision)
d$evaluation <- factor(d$evaluation, levels = namesAlgos)
d$verification <- factor(d$verification, levels = namesAlgos)
```

```r
d$evaluation <- renameAtoB(d$evaluation, "DESeq", "DESeq (old)")
d$verification <- renameAtoB(d$verification, "DESeq", "DESeq (old)")
d$evaluation <- renameAtoB(d$evaluation, "cuffdiff2", "Cuffdiff 2")
d$verification <- renameAtoB(d$verification, "cuffdiff2", "Cuffdiff 2")
d$evaluation <- renameAtoB(d$evaluation, "edgeR.robust", "edgeR-robust")
d$verification <- renameAtoB(d$verification, "edgeR.robust", "edgeR-robust")
```

```r
p <- ggplot(d, aes(x = evaluation, y = sensitivity, color = evaluation))
p + geom_boxplot(outlier.colour = rgb(0, 0, 0, 0)) + theme_bw() + facet_wrap(~verification) +
    geom_point(position = position_jitter(w = 0.1, h = 0), color = "grey50",
        size = 1) + theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    xlab("")
```

```r
p <- ggplot(d, aes(x = evaluation, y = precision, color = evaluation))
p + geom_boxplot(outlier.colour = rgb(0, 0, 0, 0)) + theme_bw() + facet_wrap(~verification) +
    geom_point(position = position_jitter(w = 0.1, h = 0), color = "grey50",
        size = 1) + theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    xlab("")
```

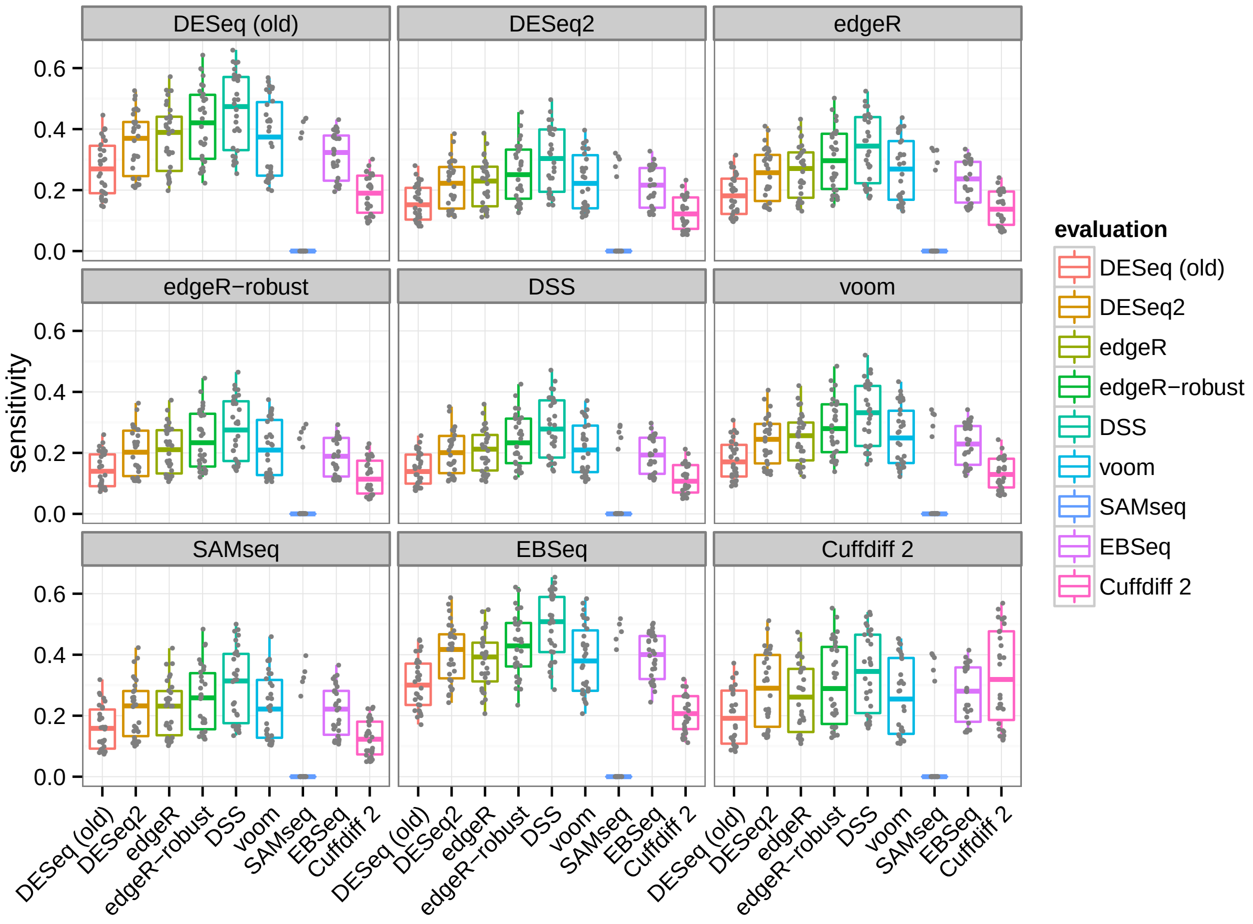


Figure 3: Sensitivity, where each algorithm's calls (adjusted p-value $< .1$) in the evaluation set (color boxes) is compared against another algorithm's calls (adjusted p-value $< .1$) in the verification set (grey labels).

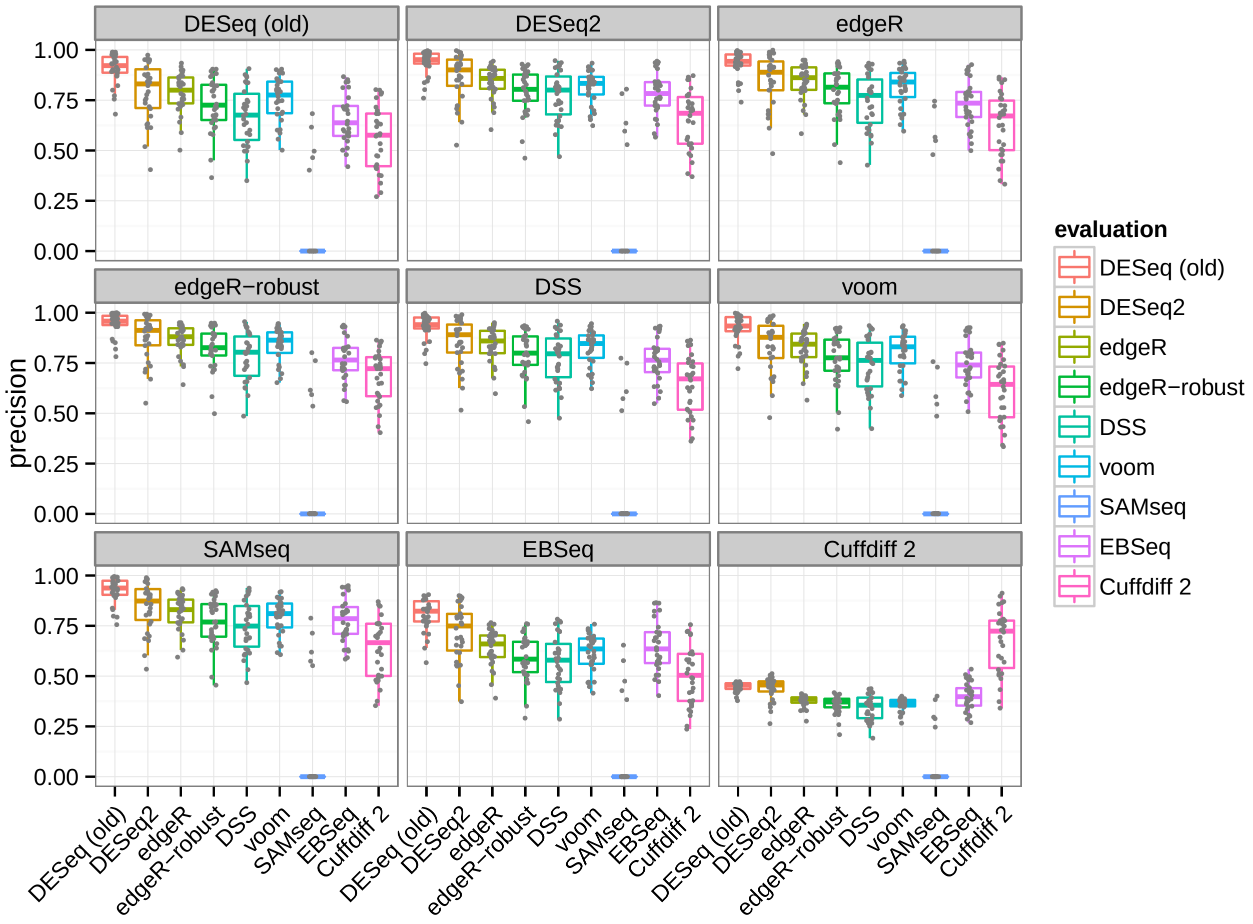


Figure 4: Precision, where each algorithm's calls (adjusted p-value $< .1$) in the evaluation set (color boxes) is compared against another algorithm's calls (adjusted p-value $< .1$) in the verification set (grey labels).

4 Compare sensitivity at a given precision

In this section, we examine the false discovery rate (1 - precision) for a given range of α, the adjusted p-value cutoff for calling a gene significant in the evaluation set. We then plot this curve for all algorithms performance in the evalulation set, and using each algorithm for determining the "true" calls in the verification set. We also use these curves to make a plot of sensitivities for each algorithm, using that α adjusted p-value threshold such that the precision is 0.9 (hence and the false discovery rate is the desired 0.1).

```
nreps <- length(resTest)
alphas <- exp(seq(from = log(0.001), to = log(0.2), length = 100))
alphaOut <- 0.1
fdrAtAlpha <- do.call(rbind, lapply(namesAlgos, function(algo) {
    sigTestList <- lapply(1:nreps, function(i) lapply(alphas, function(alpha) which(resTest[[i]][[algo]] <
        alpha)))
    callsTest <- sapply(1:nreps, function(i) {
        sapply(seq_along(alphas), function(j) {
            length(sigTestList[[i]][[j]])
        })
    })
    do.call(rbind, lapply(namesAlgos, function(algoOut) {
        fdr <- sapply(1:nreps, function(i) {
            sigHeldout <- resHeldout[[i]][[algoOut]] < alphaOut
            sapply(seq_along(alphas), function(j) {
                ifelse(sum(sigTestList[[i]][[j]]) > 0, mean(!sigHeldout[sigTestList[[i]][[j]]]),
                  0)
            })
        })
        data.frame(alpha = alphas, FDR = apply(fdr, 1, median), callsTest = apply(callsTest,
            1, median), algorithm = rep(algo, 100), verification = rep(algoOut,
            100))
    }))
}))
```

```
fdrAtAlphaPlot <- fdrAtAlpha
fdrAtAlphaPlot$algorithm <- renameAtoB(fdrAtAlphaPlot$algorithm, "DESeq", "DESeq (old)")
fdrAtAlphaPlot$verification <- renameAtoB(fdrAtAlphaPlot$verification, "DESeq",
    "DESeq (old)")
fdrAtAlphaPlot$algorithm <- renameAtoB(fdrAtAlphaPlot$algorithm, "cuffdiff2",
    "Cuffdiff 2")
fdrAtAlphaPlot$verification <- renameAtoB(fdrAtAlphaPlot$verification, "cuffdiff2",
    "Cuffdiff 2")
fdrAtAlphaPlot$algorithm <- renameAtoB(fdrAtAlphaPlot$algorithm, "edgeR.robust",
    "edgeR-robust")
fdrAtAlphaPlot$verification <- renameAtoB(fdrAtAlphaPlot$verification, "edgeR.robust",
    "edgeR-robust")
```

```
# only show results when the evaluation algorithm has positive median calls
p <- ggplot(fdrAtAlphaPlot[fdrAtAlphaPlot$callsTest > 0, ], aes(x = alpha, y = FDR,
    color = algorithm))
p + geom_line() + theme_bw() + geom_abline(intercept = 0, slope = 1) + facet_wrap(~verification) +
    ylab("FDR (1 - precision)") + xlab("evaluation set adjusted p-value cutoff") +
    scale_x_continuous(breaks = c(0, 0.05, 0.1, 0.15)) + scale_y_continuous(breaks = c(0,
    0.05, 0.1, 0.15, 0.2, 0.25)) + coord_cartesian(ylim = c(-0.01, 0.3), xlim = c(-0.01,
    0.2))
```

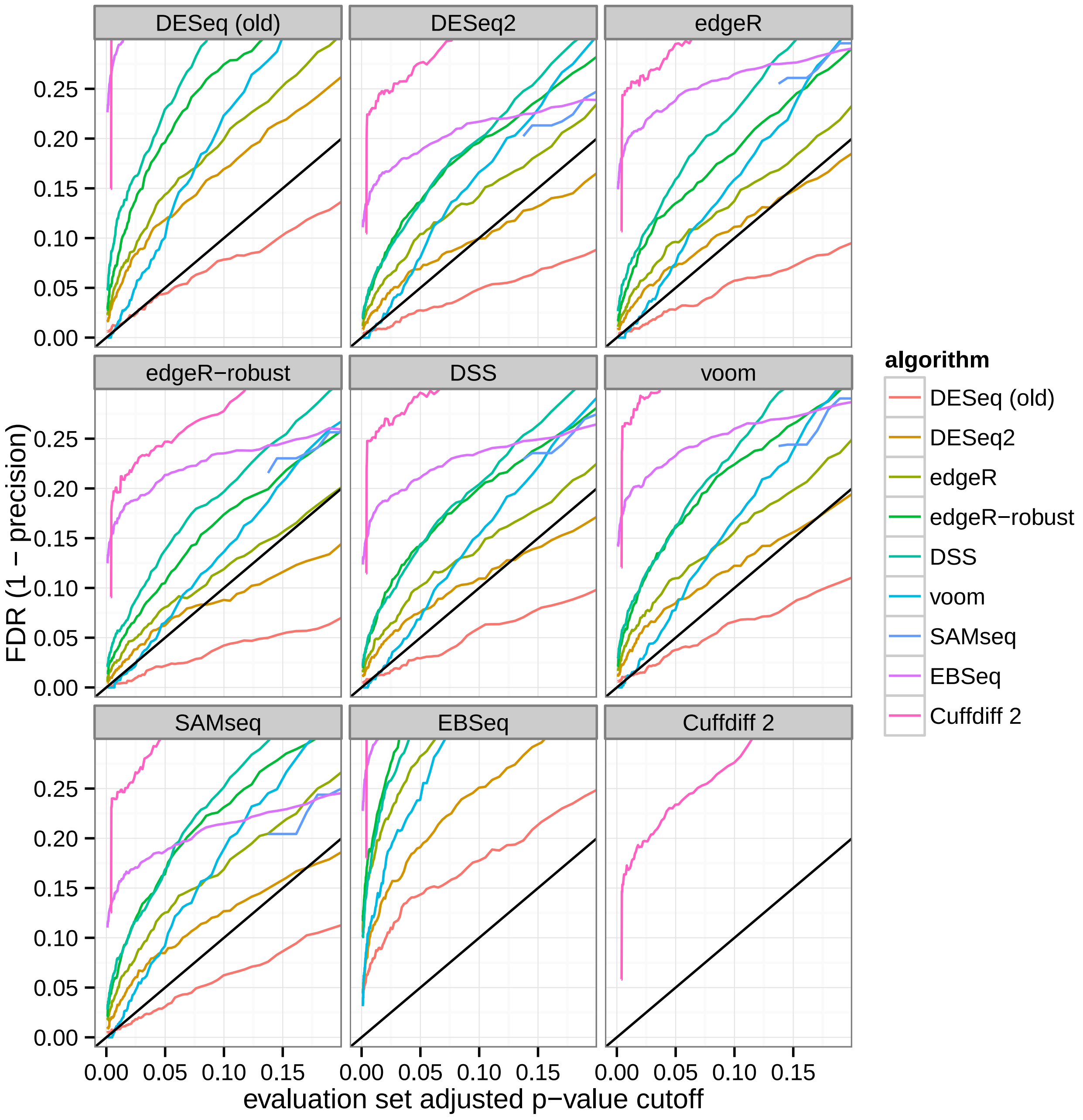


Figure 5: Actual versus nominal false discovery rate for the Bottomly et al. dataset. The actual false discovery rate was calculated using the median of (1 - precision) as in the previous precision plots, though here varying the adjusted p-value cutoff, i.e., the nominal FDR, for the evaluation set. A false positive was defined as a call in the evaluation set for a given critical value of adjusted p-value which did not have adjusted p-value less than 0.1 in the verification set. Ideally, curves should fall on the identity line (indicated by a black line); curves that fall above indicate that an algorithm is too permissive (anti-conservative), curves falling below indicate that an algorithm does not use its type-I error budget, i.e., is conservative. DESeq2 had a false discovery rate nearly matching the nominal false discovery rate (black diagonal line) for the majority of algorithms used to determine the verification set calls. The old DESeq tool was often too conservative.

```r
alphaTarget <- 0.1
alphaOut <- 0.1
sensAtTarget <- do.call(rbind, lapply(namesAlgos, function(algoOut) {
    do.call(rbind, lapply(namesAlgos, function(algo) {
        sens <- sapply(1:nreps, function(i) {
            sigHeldout <- resHeldout[[i]][[algoOut]] < alphaOut
            idx <- fdrAtAlpha$algorithm == algo & fdrAtAlpha$verification ==
                algoOut
            lessTarget <- which(fdrAtAlpha$FDR[idx] < alphaTarget)
            if (length(lessTarget) == 0)
                return(0)
            alpha <- alphas[lessTarget[length(lessTarget)]]
            sigTest <- resTest[[i]][[algo]] < alpha
            ifelse(sum(sigHeldout) > 0, mean(sigTest[sigHeldout]), 0)
        })
        data.frame(sensitivity = sens, algorithm = rep(algo, nreps), verification = rep(algoOut,
            nreps))
    }))
}))
```

```r
sensAtTarget$algorithm <- renameAtoB(sensAtTarget$algorithm, "DESeq", "DESeq (old)")
sensAtTarget$verification <- renameAtoB(sensAtTarget$verification, "DESeq",
    "DESeq (old)")
sensAtTarget$algorithm <- renameAtoB(sensAtTarget$algorithm, "cuffdiff2", "Cuffdiff 2")
sensAtTarget$verification <- renameAtoB(sensAtTarget$verification, "cuffdiff2",
    "Cuffdiff 2")
sensAtTarget$algorithm <- renameAtoB(sensAtTarget$algorithm, "edgeR.robust",
    "edgeR-robust")
sensAtTarget$verification <- renameAtoB(sensAtTarget$verification, "edgeR.robust",
    "edgeR-robust")
```

```r
p <- ggplot(sensAtTarget, aes(x = algorithm, y = sensitivity, color = algorithm))
p + geom_boxplot(outlier.colour = rgb(0, 0, 0, 0)) + theme_bw() + facet_wrap(~verification) +
    geom_point(position = position_jitter(w = 0.1, h = 0), color = "grey50",
        size = 1) + theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
    xlab("")
```

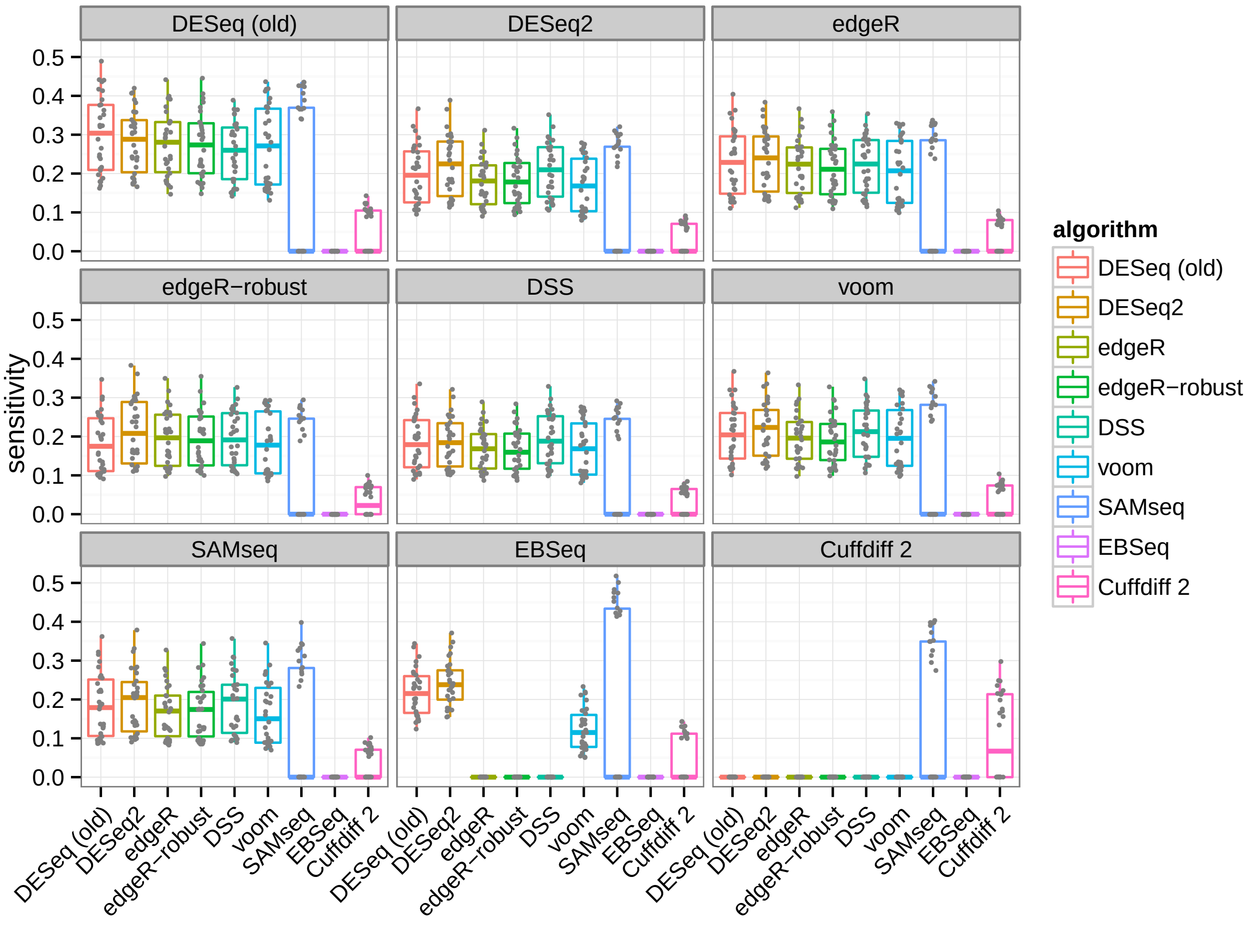
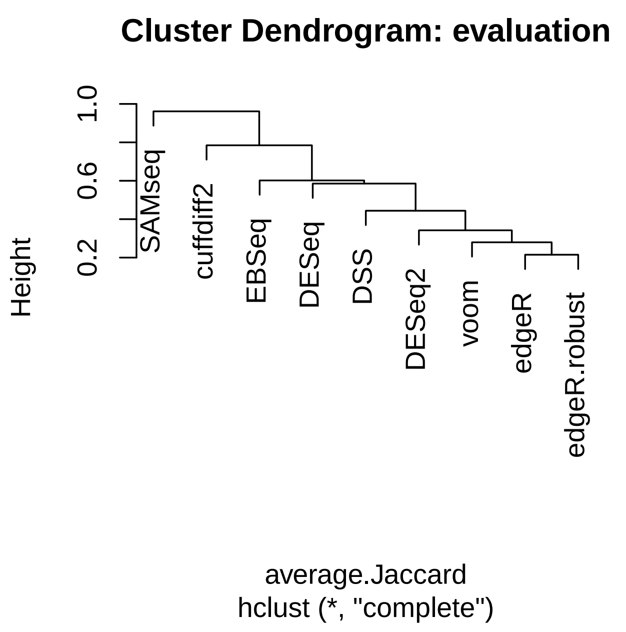


Figure 6: Sensitivity of algorithms evaluated while controlling the median precision. While it was generally noted that sensitivity and precision were negatively correlated, here this effect was controlled by setting the adjusted p-value cutoff for the evaluation set calls such that the median precision of all algorithms would be 0.9 (actual false discovery rate of 0.1). This amounted to finding the point on the x-axis in the previous figure, where the curve crosses 0.1 on the y-axis. For most algorithms, this meant setting an adjusted p-value cutoff below 0.1. DESeq2 often had the highest median sensitivity for a given target precision, though the variability across random replicates was generally larger than the difference between algorithms.



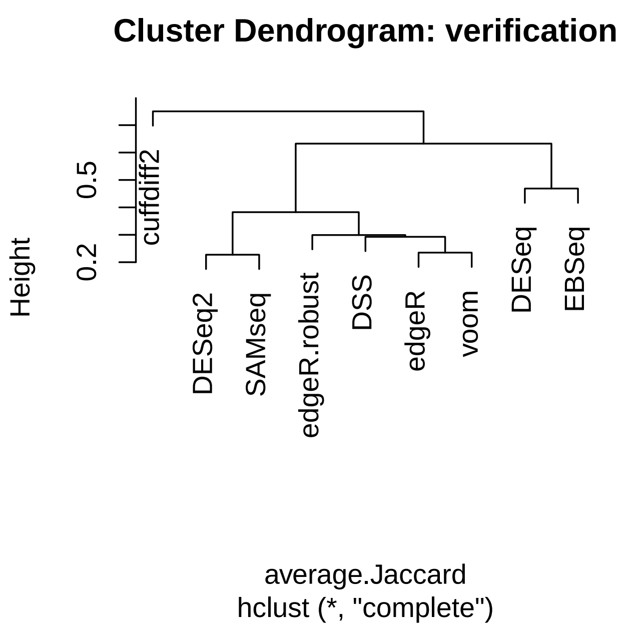


Figure 7: Clustering of calls (adjusted p-value $< .1$) with distances based on the Jaccard index

5 Clustering of calls

```
alpha <- 0.1
library("abind")
# first with evaluation sets
j0 <- lapply(1:nreps, function(i) as.matrix(dist(t(resTest[[i]] < alpha), method = "binary")))
j <- abind(j0, along = 3)
average.Jaccard <- apply(j, c(1, 2), mean)
average.Jaccard <- as.dist(average.Jaccard)
hcTest <- hclust(average.Jaccard)

# again with verification sets
j0 <- lapply(1:nreps, function(i) as.matrix(dist(t(resHeldout[[i]] < alpha),
    method = "binary")))
j <- abind(j0, along = 3)
average.Jaccard <- apply(j, c(1, 2), mean)
average.Jaccard <- as.dist(average.Jaccard)
hcHeldout <- hclust(average.Jaccard)
```

```
par(mfrow = c(1, 2))
plot(hcTest, main = "Cluster Dendrogram: evaluation")
plot(hcHeldout, main = "Cluster Dendrogram: verification")
```

6 Heatmap of calls for a single random replicate

```
alpha <- 0.1
library("gplots")

##
## Attaching package:  'gplots'
##
## The following object is masked from 'package:IRanges':
##
##     space
##
## The following object is masked from 'package:stats':
##
##     lowess

# which replicate had median number of calls
median(colMeans(sapply(1:30, function(i) colSums(resTest[[i]] < alpha))))

## [1] 700.4

colMeans(sapply(1:30, function(i) colSums(resTest[[i]] < alpha)))

##  [1]  897.2 1704.9  468.0  867.0  433.7  750.8  685.8  568.0  444.6  420.3
## [11]  618.7  465.8  532.6  873.7  947.9  424.3  943.7 1405.8  891.7  825.2
## [21]  597.1  453.7  715.1  809.1  994.3  881.1 1159.2  550.0  585.1  607.0

i <- 23
nMat <- resTest[[i]] < alpha
colnames(nMat)[colnames(nMat) == "DESeq"] <- "DESeq (old)"
colnames(nMat)[colnames(nMat) == "cuffdiff2"] <- "Cuffdiff 2"
colnames(nMat)[colnames(nMat) == "edgeR.robust"] <- "edgeR-robust"
nMat <- nMat[rowSums(nMat) > 0, ]
mode(nMat) <- "numeric"
hc <- hclust(dist(t(nMat), method = "binary"))
y <- sweep(nMat, 2, 2^(ncol(nMat) - order(hc$order)), "*")
z <- nMat[order(-rowSums(y)), ]
```

```
heatmap.2(z, main = paste(nrow(nMat), "out of", nrow(resTest[[1]]), "genes"),
    trace = "none", key = FALSE, Rowv = FALSE, labRow = FALSE, Colv = as.dendrogram(hc),
    dendrogram = "column", scale = "none", col = c("grey", "red"), mar = c(15,
        5), lwid = c(2, 10), cexCol = 2.5)
```

```
nMat <- resHeldout[[i]] < alpha
colnames(nMat)[colnames(nMat) == "DESeq"] <- "DESeq (old)"
colnames(nMat)[colnames(nMat) == "cuffdiff2"] <- "Cuffdiff 2"
colnames(nMat)[colnames(nMat) == "edgeR.robust"] <- "edgeR-robust"
nMat <- nMat[rowSums(nMat) > 0, ]
mode(nMat) <- "numeric"
hc <- hclust(dist(t(nMat), method = "binary"))
y <- sweep(nMat, 2, 2^(ncol(nMat) - order(hc$order)), "*")
z <- nMat[order(-rowSums(y)), ]
```

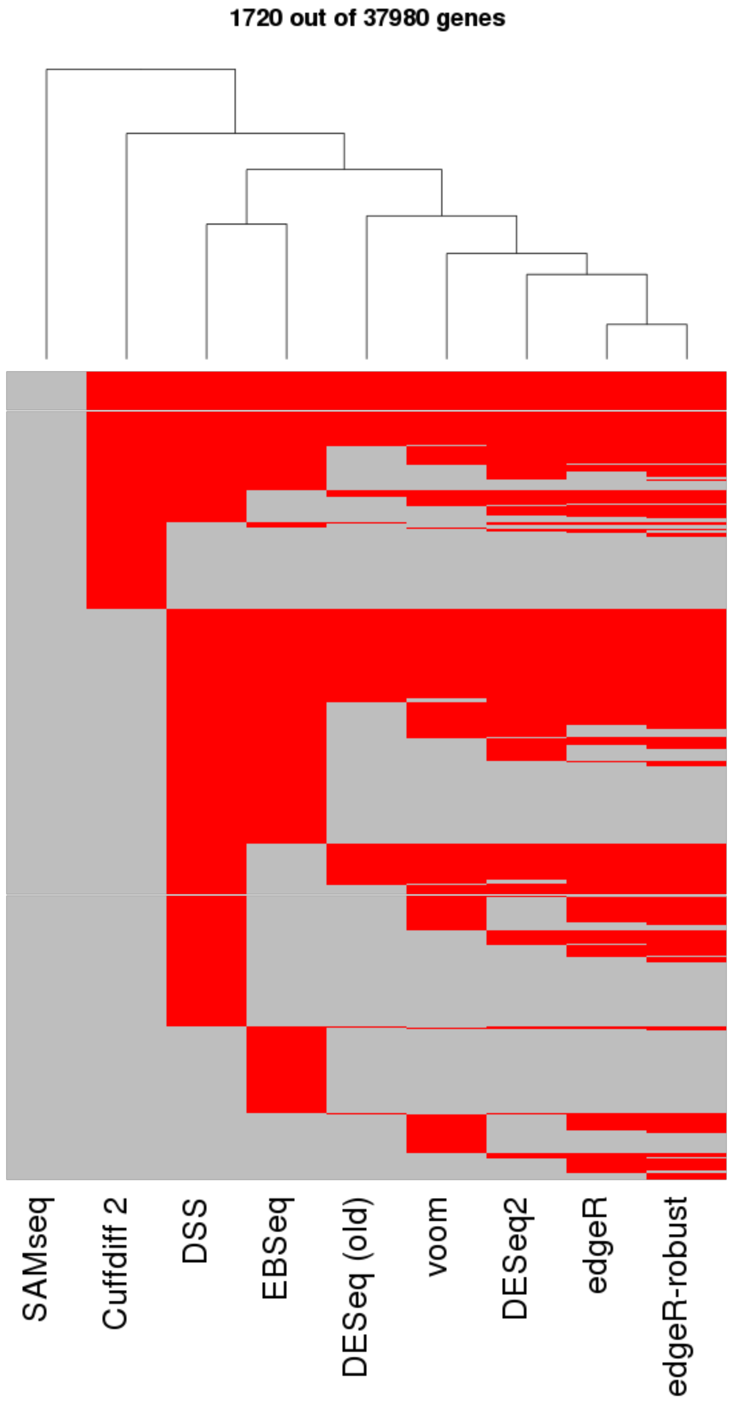


Figure 8: Example of evaluation set calls (adjusted p-value $< .1$) for a single replicate of the random sampling

```r
heatmap.2(z, main = paste(nrow(nMat), "out of", nrow(resTest[[1]]), "genes"),
    trace = "none", key = FALSE, Rowv = FALSE, labRow = FALSE, Colv = as.dendrogram(hc),
    dendrogram = "column", scale = "none", col = c("grey", "red"), mar = c(15,
        5), lwid = c(2, 10), cexCol = 2.5)
```

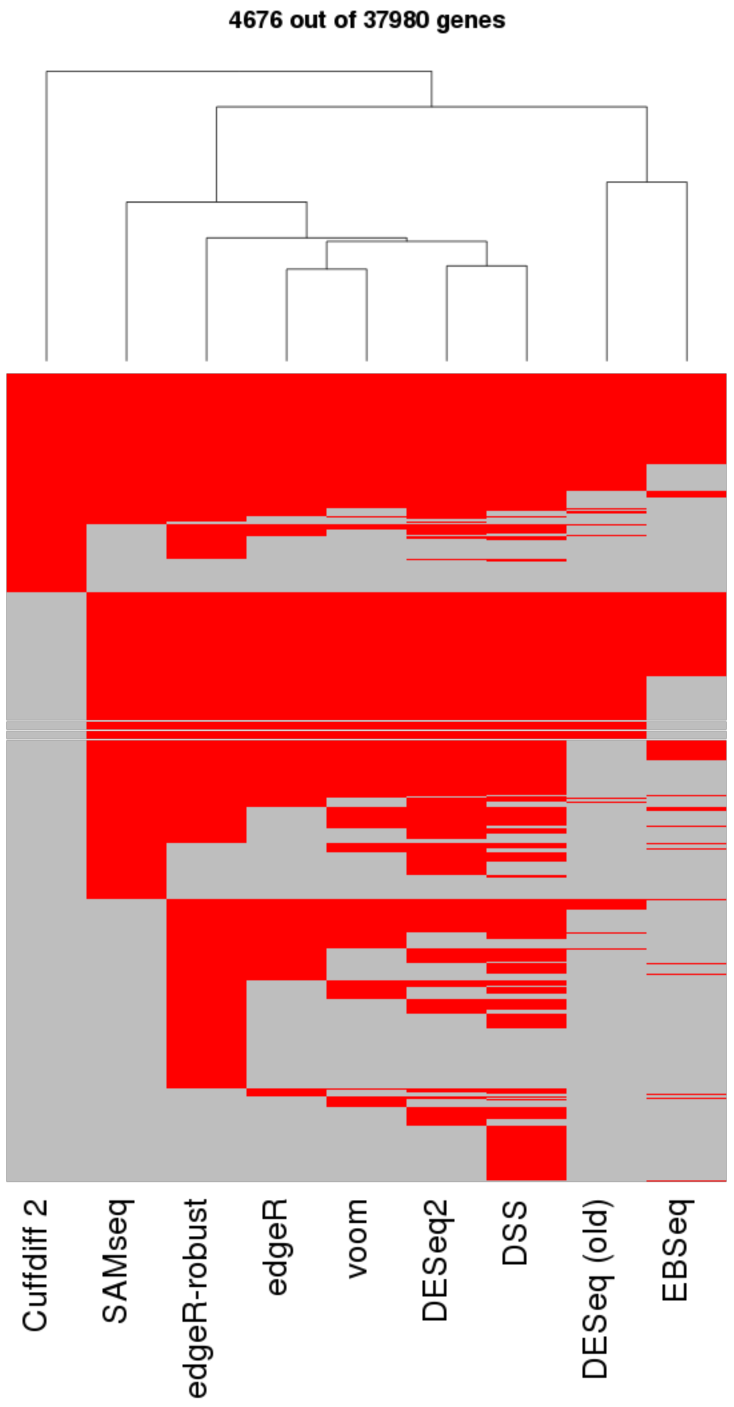


Figure 9: Example of verification set calls (adjusted p-value $< .1$) for a single replicate of the random sampling

7 Logarithmic fold changes from a single random replicate

```r
m <- as.matrix(lfcHeldout[[i]])
# use DESeq2 to remove those genes with all zero counts
m <- m[!is.na(m[, "DESeq2"]), ]
# EBSeq does not return LFC estimates
m <- m[, -which(colnames(m) == "EBSeq")]
colnames(m)[colnames(m) == "DESeq"] <- "DESeq (old)"
colnames(m)[colnames(m) == "cuffdiff2"] <- "Cuffdiff 2"
colnames(m)[colnames(m) == "edgeR.robust"] <- "edgeR-robust"
```

```r
library("LSD")
heatpairs(m, xlim = c(-1, 1), ylim = c(-1, 1), cor.cex = 3, main = "")
```

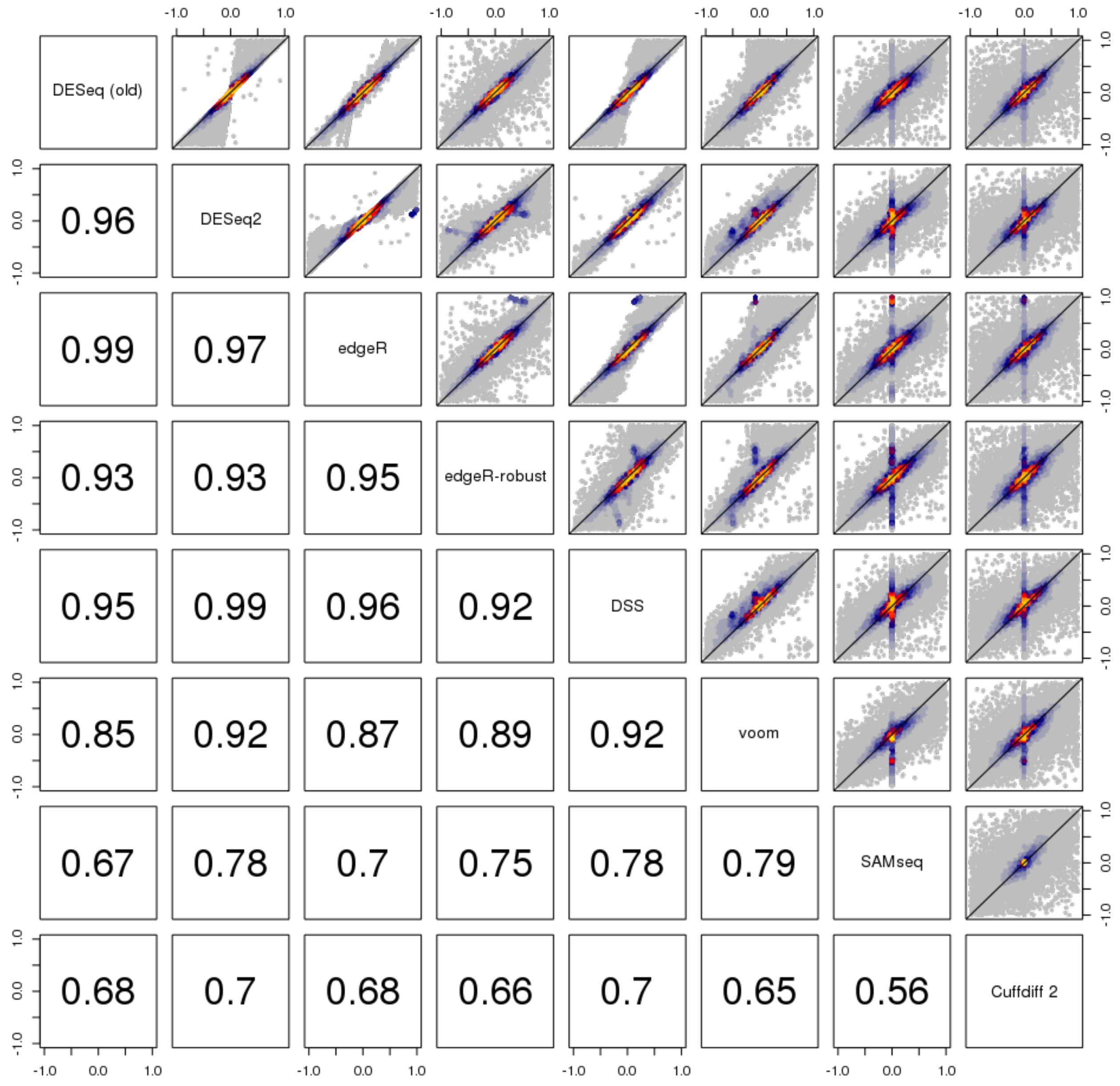


Figure 10: Logarithmic (base 2) fold changes for a single replicate of random sampling. Bottom panels show Pearson correlations.

8 Sign change of logarithmic fold change

The following code is used to evaluate the number of LFC sign changes for genes called with adjusted p-value $< .1$ by either *DESeq2* or *cuffdiff2*. We count a wrong sign if one algorithm calls the gene, and the sign of the LFC for the gene is positive for one algorithm and negative for the other.

In the evaluation set:

```
alpha <- 0.1
wrongSign <- sapply(1:nreps, function(i) {
    idx <- resTest[[i]]$DESeq2 < alpha | resTest[[i]]$cuffdiff2 < alpha
    sum(sign(lfcTest[[i]]$DESeq2[idx]) == -1 * sign(lfcTest[[i]]$cuffdiff2[idx]),
        na.rm = TRUE)/sum(idx)
})
wrongSign

##  [1] 0.03141 0.03500 0.04488 0.02857 0.03793 0.03718 0.03252 0.03774
##  [9] 0.06087 0.05882 0.03765 0.04444 0.05423 0.02430 0.02727 0.04047
## [17] 0.02861 0.03542 0.03761 0.01981 0.04121 0.05326 0.03484 0.03575
## [25] 0.04463 0.02989 0.03779 0.04208 0.04491 0.05107

mean(wrongSign)

## [1] 0.03901
```

In the verification set:

```
alpha <- 0.1
wrongSign <- sapply(1:nreps, function(i) {
    idx <- resHeldout[[i]]$DESeq2 < alpha | resHeldout[[i]]$cuffdiff2 < alpha
    sum(sign(lfcHeldout[[i]]$DESeq2[idx]) == -1 * sign(lfcHeldout[[i]]$cuffdiff2[idx]),
        na.rm = TRUE)/sum(idx)
})
wrongSign

##  [1] 0.04052 0.04162 0.03870 0.04391 0.03611 0.03917 0.04093 0.03827
##  [9] 0.03866 0.03523 0.04187 0.03892 0.03988 0.04399 0.04373 0.03571
## [17] 0.04111 0.04186 0.03796 0.03715 0.04149 0.03992 0.03869 0.04085
## [25] 0.03749 0.04346 0.04209 0.03920 0.03745 0.03650

mean(wrongSign)

## [1] 0.03975
```

9 Session information

- R version 3.1.0 (2014-04-10), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Base packages: base, datasets, grDevices, graphics, grid, methods, parallel, stats, utils
- Other packages: Biobase 2.24.0, BiocGenerics 0.10.0, DESeq2 1.4.0, DESeq2paper 1.3, GenomeInfoDb 1.0.0, GenomicRanges 1.16.0, IRanges 1.21.45, LSD 2.5, MASS 7.3-31, RColorBrewer 1.0-5, Rcpp 0.11.1, RcppArmadillo 0.4.200.0, abind 1.4-0, colorRamps 2.3, ellipse 0.3-8, ggplot2 0.9.3.1, gplots 2.13.0, gridExtra 0.9.1, gtools 3.3.1, hexbin 1.27.0, knitr 1.5, reshape 0.8.5, schoolmath 0.4, vsn 3.32.0, xtable 1.7-3
- Loaded via a namespace (and not attached): AnnotationDbi 1.26.0, BiocInstaller 1.14.2, DBI 0.2-7, KernSmooth 2.23-12, RSQLite 0.11.4, XML 3.98-1.1, XVector 0.4.0, affy 1.42.2, affyio 1.32.0, annotate 1.42.0, bitops 1.0-6, caTools 1.16, codetools 0.2-8, colorspace 1.2-4, dichromat 2.0-0, digest 0.6.4, evaluate 0.5.5, formatR 0.10, gdata 2.13.3, genefilter 1.46.0, geneplotter 1.42.0, gtable 0.1.2, highr 0.3, labeling 0.2, lattice 0.20-29, limma 3.20.1, locfit 1.5-9.1, munsell 0.4.2, plyr 1.8.1, preprocessCore 1.26.1, proto 0.3-10, reshape2 1.4, scales 0.2.3, splines 3.1.0, stats4 3.1.0, stringr 0.6.2, survival 2.37-7, tools 3.1.0, zlibbioc 1.10.0