# Assessment of DESeq2 performance through simulation

Michael Love

October 14, 2014

## Contents

# 1 Differential expression analysis

We assessed the sensitivity and specificity of various algorithms using simulation to complement on analysis on real data. The following Negative Binomial simulation samples mean and dispersions from the joint distribution of estimated mean and dispersion from the Pickrell et al dataset. The true differences between two groups are drawn from either $z$, 0 or $-z$, where the 0 component represents 80% of the genes. The absolute value of the effect size $z$ for the 20% of genes with differential expression is varied, as is the total sample size $m$ (such that each group has $m/2$ samples). 10,000 genes were simulated, and each combination of parameters was repeated 6 times. The code to generate these results is in `/inst/script/simulateDE.R`

```
library("DESeq2paper")
```

```
data("results_simulateDE")
# SAMseq has no calls for the 3 vs 3 comparison
res <- res[!(res$m == 6 & res$algorithm == "SAMseq"), ]
res$m <- factor(res$m)
levels(res$m) <- paste0("m=", levels(res$m))
res$effSize <- factor(res$effSize)
levels(res$effSize) <- c("fold change 2", "fold change 3", "fold change 4")
res$algorithm <- factor(res$algorithm)
levels(res$algorithm)[levels(res$algorithm) == "DESeq"] <- "DESeq (old)"
resMinusEBSeq <- res[res$algorithm != "EBSeq", ]  # EBSeq does not produce p-values
resMinusEBSeq$algorithm <- droplevels(resMinusEBSeq$algorithm)
```

```
library("ggplot2")
p <- ggplot(resMinusEBSeq, aes(y = sensitivity, x = oneminusspecpvals, color = algorithm,
    shape = algorithm))
p + geom_point() + theme_bw() + facet_grid(effSize ~ m) + scale_shape_manual(values = 1:9) +
    xlab("1 - specificity (false positive rate)") + coord_cartesian(xlim = c(-0.003,
    0.035)) + geom_vline(xintercept = 0.01) + scale_x_continuous(breaks = c(0,
    0.02))
```

```
library("ggplot2")
p <- ggplot(res, aes(y = sensitivity, x = oneminusprec, color = algorithm, shape = algorithm))
p + geom_point() + theme_bw() + facet_grid(effSize ~ m) + scale_shape_manual(values = 1:9) +
    xlab("1 - precision (false discovery rate)") + coord_cartesian(xlim = c(-0.03,
    0.3)) + geom_vline(xintercept = 0.1)
```

```
library("reshape")
id.vars <- c("algorithm", "effSize", "m")
measure.vars <- c("sens0to20", "sens20to100", "sens100to300", "sensmore300")
melted <- melt(res[, c(id.vars, measure.vars)], id.vars = id.vars, measure.vars = measure.vars)
names(melted) <- c(id.vars, "aveexp", "sensitivity")
levels(melted$aveexp) <- c("<20", "20-100", "100-300", ">300")
```

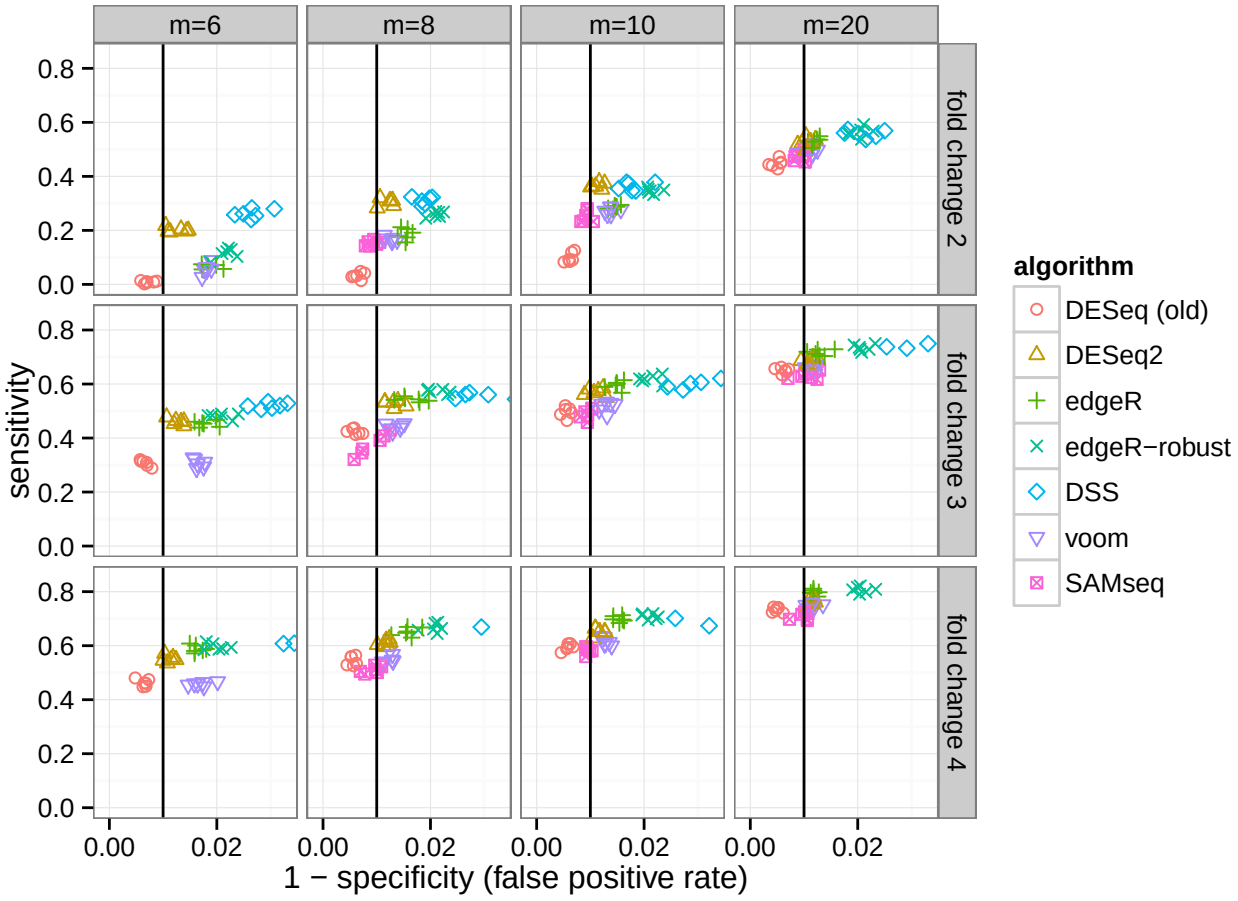


Figure 1: Use of simulation to assess the sensitivity and specificity of algorithms across combinations of sample size and effect size. The sensitivity was calculated as the fraction of genes with adjusted p-value less than 0.1 among the genes with true differences between group means. The specificity was calculated as the fraction of genes with p-value greater than 0.01 among the genes with no true differences between group means. The p-value was chosen instead of the adjusted p-value, as this allows for comparison against the expected fraction of p-values less than a critical value given the uniformity of p-values under the null hypothesis. DESeq2 often had the highest sensitivity of those algorithms which control the false positive rate, i.e., those algorithms which fall on or to the left of the vertical black line (1% p-values less than 0.01 for the non-DE genes).

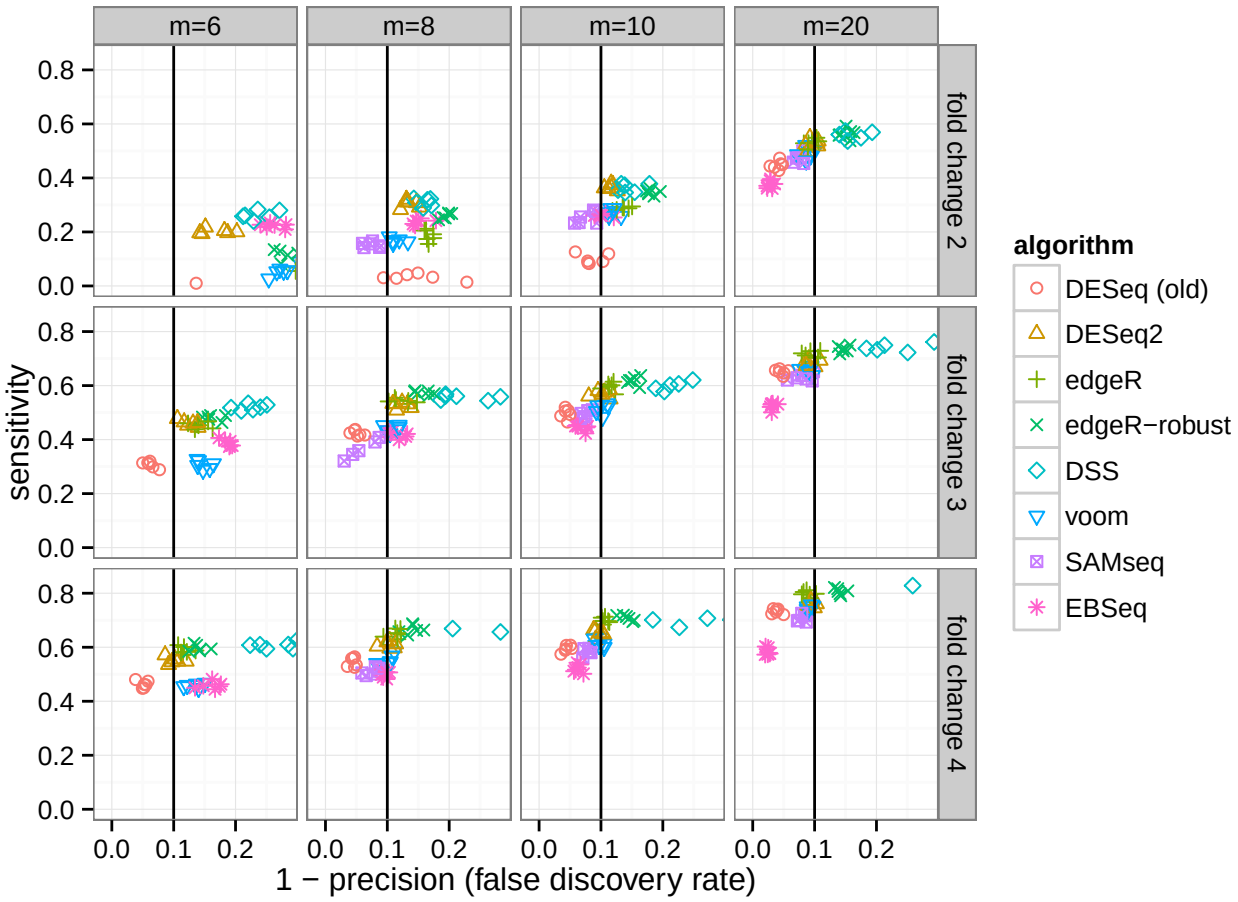


Figure 2: Sensitivity and precision of algorithms across combinations of sample size and effect size. The sensitivity was calculated as the fraction of genes with adjusted p-value less than 0.1 among the genes with true differences between group means. The precision was calculated as the fraction of genes with true differences between group means among those with adjusted p-value less than 0.1. DESeq2 often had the highest sensitivity of those algorithms which controlled the false discovery rate, i.,e., those algorithms which fall on or to the left of the vertical black line.

```
p <- ggplot(melted, aes(y = sensitivity, x = aveexp, group = algorithm, color = algorithm,
    shape = algorithm))
p + stat_summary(fun.y = "mean", geom = "line") + stat_summary(fun.y = "mean",
    geom = "point") + theme_bw() + facet_grid(effSize ~ m) + scale_shape_manual(values = 1:9) +
    theme(axis.text.x = element_text(angle = 45, hjust = 1)) + xlab("mean counts")
```

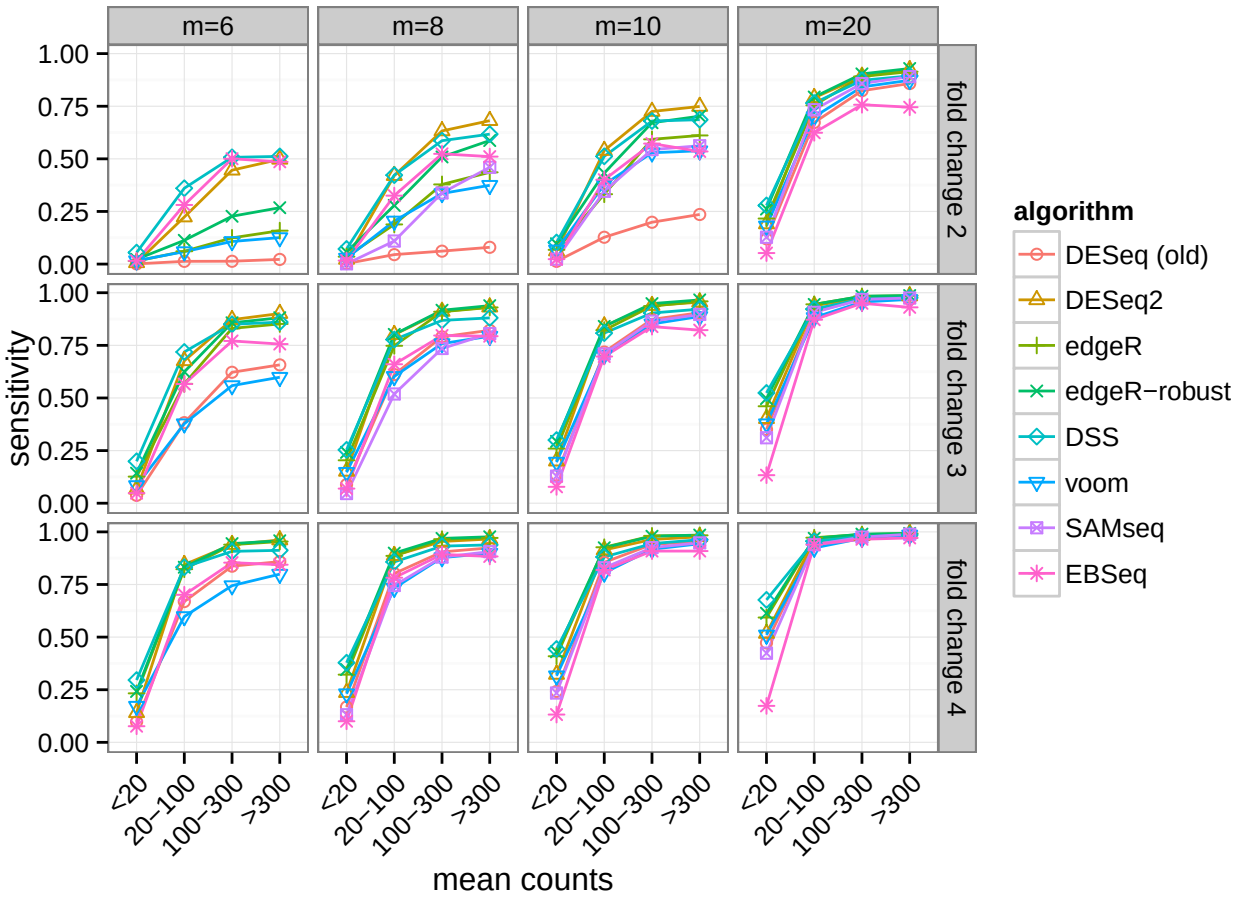


Figure 3: The sensitivity of algorithms across combinations of sample size and effect size, and further stratified by the mean of counts of the differentially expressed genes in the simulation data. Points indicate the average over 6 replicates. Algorithms all show a similar dependence of sensitivity on the mean of counts. The height of the sensitivity curve should be compared with the previous plot indicating the total sensitivity and specificity of each algorithm.

## 2 Performance in the presence of outliers

The following plots examine the affect of outliers on differential calls by the two Negative-Binomial-based methods *DESeq2* and *edgeR*. *DESeq2* was run with default settings, after turning off gene filtering, and after turning off outlier replacement. *edgeR* was run with default settings, and after using the `robust` option. The code to generate these results is in `/inst/script/simulateOutliers.R`

```
data("results_simulateOutliers")
# when < 7 replicates DESeq does not replace
res <- res[!(res$algorithm == "DESeq2-noRepl" & res$m < 14), ]
# when >= 7 replicates DESeq does not filter
res <- res[!(res$algorithm == "DESeq2-noFilt" & res$m >= 14), ]
res$m <- factor(res$m)
levels(res$m) <- paste0("m=", levels(res$m))
res$percentOutlier <- 100 * res$percentOutlier
res$percentOutlier <- factor(res$percentOutlier)
levels(res$percentOutlier) <- paste0(levels(res$percentOutlier), "% outlier")
```

Because the sensitivity-specificity curve is evaluated using the p value, we use fhe following code to pick out the point on the sensitivity-specificity curve with largest p value such that the nominal adjusted p-value is less than 0.1.

```
resSensPadj <- res[res$senspadj < 0.1, ]
resSensPadj <- resSensPadj[nrow(resSensPadj):1, ]
resSensPadj <- resSensPadj[!duplicated(with(resSensPadj, paste(algorithm, m,
    percentOutlier))), ]
summary(resSensPadj$senspadj)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0785  0.0895  0.0918  0.0912  0.0948  0.0996
```

```
library("ggplot2")
p <- ggplot(res, aes(x = oneminusspec, y = sensitivity, color = algorithm))
p + scale_x_continuous(breaks = c(0, 0.1, 0.2)) + scale_y_continuous(breaks = c(0,
    0.2, 0.4, 0.6, 0.8)) + geom_line() + theme_bw() + facet_grid(m ~ percentOutlier) +
    xlab("1 - specificity") + coord_cartesian(xlim = c(-0.03, 0.25), ylim = c(-0.05,
    0.9)) + geom_point(aes(x = oneminusspec, y = sensitivity, shape = algorithm),
    data = resSensPadj)
```

```
p <- ggplot(res, aes(x = precpadj, y = oneminusprec, color = algorithm))
p + scale_x_continuous(breaks = c(0, 0.1, 0.2)) + scale_y_continuous(breaks = c(0,
    0.1, 0.2)) + geom_line() + theme_bw() + facet_grid(m ~ percentOutlier) +
    geom_abline(intercept = 0, slope = 1) + xlab("adjusted p-value") + ylab("1 - precision (FDR)") +
    coord_cartesian(xlim = c(-0.03, 0.25), ylim = c(-0.05, 0.25)) + geom_point(aes(x = precpadj,
    y = oneminusprec, shape = algorithm), data = res[res$precpadj == 0.1, ])
```

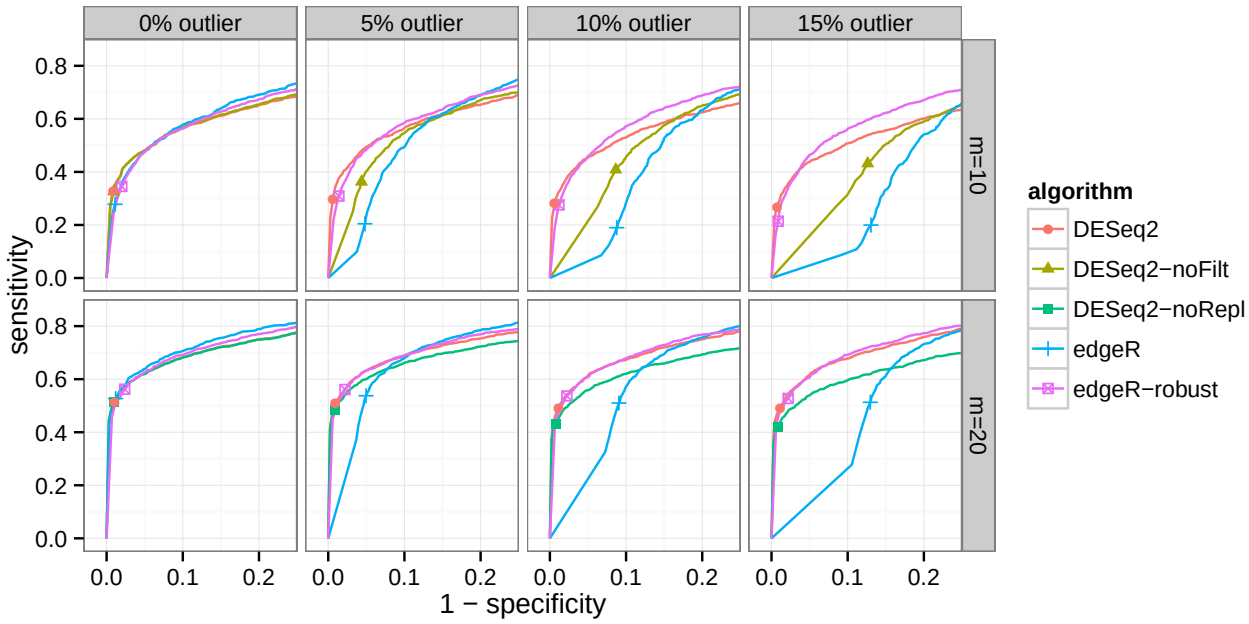


Figure 4: Sensitivity-specificity curves for detecting true differences in the presence of outliers. Negative Binomial counts were simulated for 4000 genes and total sample sizes (m) of 10 and 20, for a two-group comparison. 80% of the simulated genes had no true differential expression, while for 20% of the genes true logarithmic (base 2) fold changes of -1 or 1. The number of genes with simulated outliers was increased from 0% to 15%. The outliers were constructed for a gene by multiplying the count of a single sample by 100. Sensitivity and specificity were calculated by thresholding on p-values. Points indicate an adjusted p-value of 0.1. DESeq2 with the default settings and edgeR with the robust setting had higher area under the curve compared to running edgeR without the robust option, turning off DESeq2 gene filtering, and turning off DESeq2 outlier replacement. DESeq2 filters genes with potential outliers for samples with 3 to 6 replicates, while replacing outliers for samples with 7 or more replicates, hence the filtering can be turned off for the top row (m=10) and the replacement can be turned off for the bottom row (m=20).

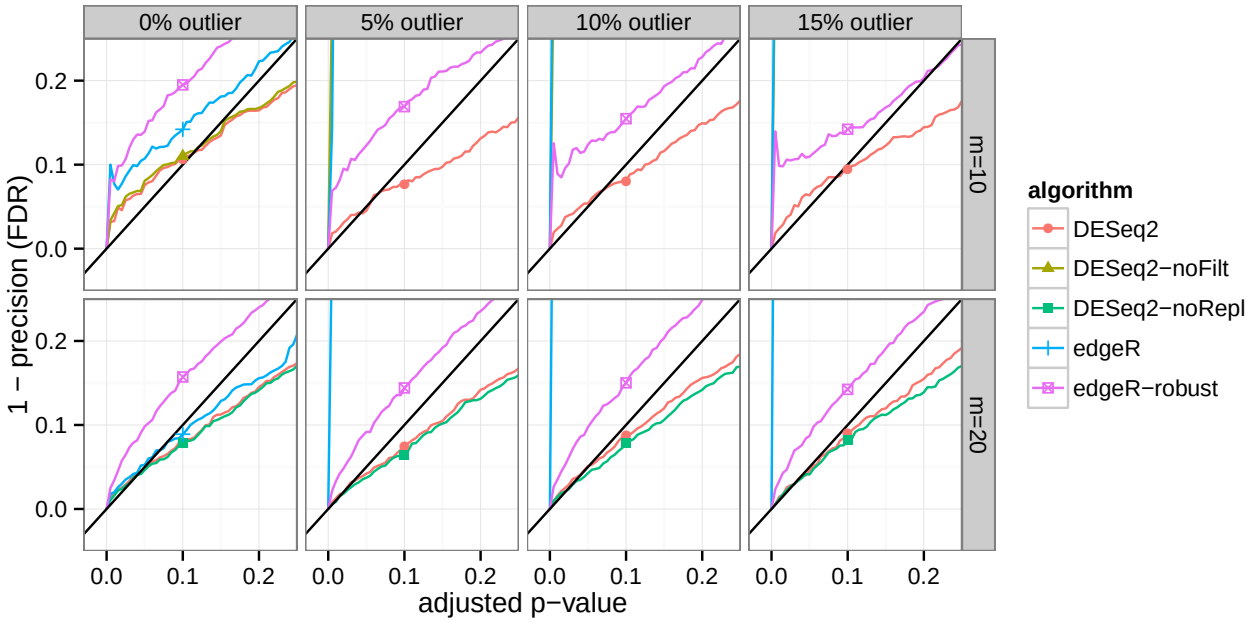


Figure 5: Outlier handling: One minus the precision (false discovery rate) plotted over various thresholds of adjusted p-value. Shown is the results for the same simulation with outliers described in the previous figure. Points indicate an adjusted p-value of 0.1. edgeR run with the robust setting had false discovery rate generally above the nominal value from the adjusted p-value threshold (black diagonal line). DESeq2 run with default settings was generally at or below the line, which indicates control of the false discovery rate.

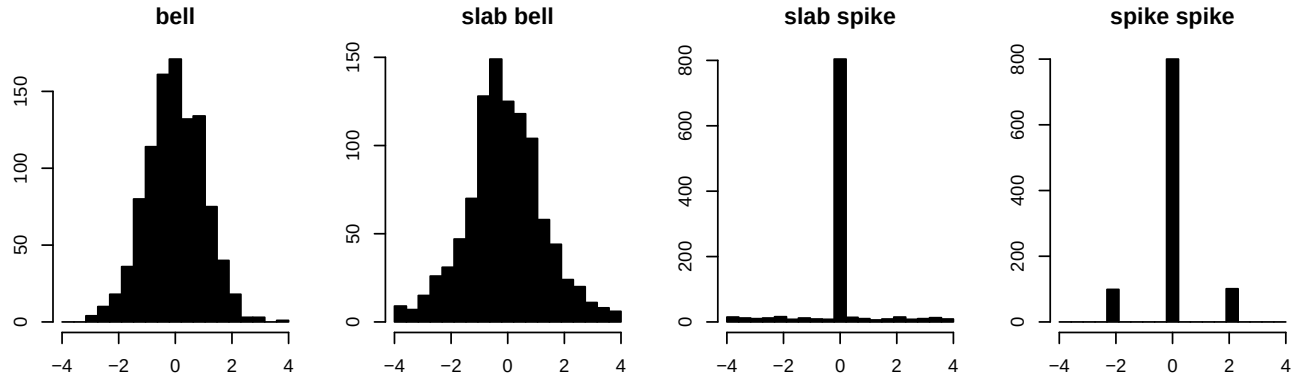


Figure 6: Benchmarking LFC estimation: Models for simulating logarithmic (base 2) fold changes. For the bell model, true logarithmic fold changes were drawn from a Normal with mean 0 and variance 1. For the slab bell model, true logarithmic fold changes were drawn for 80% of genes from a Normal with mean 0 and variance 1 and for 20% of genes from a Uniform distribution with range from -4 to 4. For the slab spike model, true logarithmic fold changes were drawn similarly to the slab bell model except the Normal is replaced with a spike of logarithmic fold changes at 0. For the spike spike model, true logarithmic fold changes were drawn according to a spike of logarithmic fold changes at 0 (80%) and a spike randomly sampled from -2 or 2 (20%). These spikes represent fold changes of 1/4 and 4, respectively.

# 3 Accuracy of log fold change estimates

The following simulations used Negative Binomial random variables with mean and dispersion pairs samples from the joint distribution of mean-dispersion estimates from the Pickrell data. In addition, true differences between two groups were randomly generated, according to the following models, diagrammed below. The accuracy of four methods for estimating the log fold change between groups were compared by the root mean squared error (RMSE) and the mean absolute error (MAE). The four methods were chosen for their particular focus on the logs fold change estimate. The code to generate these results is in `/inst/script/simulateLFCAccuracy.R`.

```
par(mfrow = c(1, 4), mar = c(3, 3, 3, 1))
n <- 1000
brks <- seq(from = -4, to = 4, length.out = 20)
trimit <- function(x) x[x > -4 & x < 4]  # for visualization only
hist(trimit(rnorm(n)), breaks = brks, col = "black", main = "bell", xlab = "",
    ylab = "")
hist(trimit(c(rnorm(n * 8/10), runif(n * 2/10, -4, 4))), breaks = brks, col = "black",
    main = "slab bell", xlab = "", ylab = "")
hist(c(rep(0, n * 8/10), runif(n * 2/10, -4, 4)), breaks = brks, col = "black",
    main = "slab spike", xlab = "", ylab = "")
hist(c(rep(0, n * 8/10), sample(c(-2, 2), n * 2/10, TRUE)), breaks = brks, col = "black",
    main = "spike spike", xlab = "", ylab = "")
```

```
data("results_simulateLFCAccuracy")
library("ggplot2")
library("Hmisc")
```

```
## Loading required package:  lattice
## Loading required package:  survival
## Loading required package:  splines
## Loading required package:  Formula
##
## Attaching package:  'Hmisc'
##
## The following objects are masked from 'package:Biobase':
##
##    combine, contents
##
## The following objects are masked from 'package:xtable':
##
##    label, label<-
##
## The following object is masked from 'package:BiocGenerics':
##
##    combine
##
## The following objects are masked from 'package:base':
##
##    format.pval, round.POSIXt, trunc.POSIXt, units
```

```
p <- ggplot(data = res, aes(x = m, y = RMSE, color = method, shape = method))
p + stat_summary(fun.y = mean, geom = "point") + stat_summary(fun.y = mean,
    geom = "line") + stat_summary(fun.data = mean_cl_normal, geom = "errorbar") +
    theme_bw() + xlab("total sample size") + facet_wrap(~type) + scale_x_continuous(breaks = unique(res$m))
```

```
p <- ggplot(data = res[grepl("spike", res$type), ], aes(x = m, y = DiffRMSE,
    color = method, shape = method))
p + stat_summary(fun.y = mean, geom = "point") + stat_summary(fun.y = mean,
    geom = "line") + stat_summary(fun.data = mean_cl_normal, geom = "errorbar") +
    theme_bw() + xlab("total sample size") + ylab("RMSE only of DE genes") +
    facet_wrap(~type) + scale_x_continuous(breaks = unique(res$m))
```

```
p <- ggplot(data = res, aes(x = m, y = MAE, color = method, shape = method))
p + stat_summary(fun.y = mean, geom = "point") + stat_summary(fun.y = mean,
    geom = "line") + stat_summary(fun.data = mean_cl_normal, geom = "errorbar") +
    theme_bw() + xlab("total sample size") + ylab("MAE") + facet_wrap(~type) +
    scale_x_continuous(breaks = unique(res$m))
```

```
p <- ggplot(data = res[grepl("spike", res$type), ], aes(x = m, y = DiffMAE,
    color = method, shape = method))
p + stat_summary(fun.y = mean, geom = "point") + stat_summary(fun.y = mean,
```

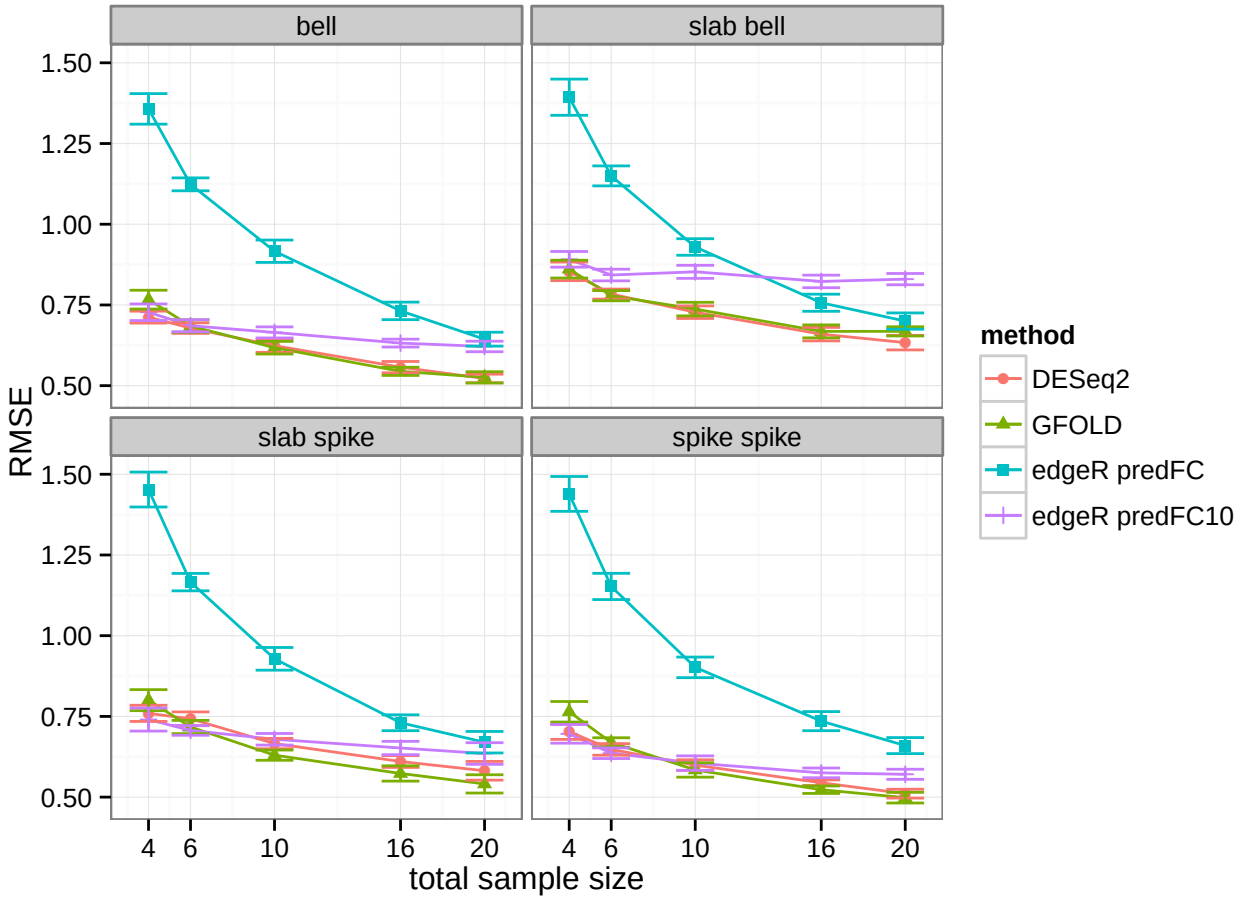


Figure 7: Root mean squared error (RMSE) for estimating logarithmic fold changes under the four models of logarithmic fold changes and varying total sample size m. Simulated Negative Binomial counts were generated for two groups and for 1000 genes. Points and error bars are drawn for the mean and 95% confidence interval over 10 replications. DESeq2 and GFOLD, which both implement posterior logarithmic fold change estimates, had lower root mean squared error to the true logarithmic fold changes over all genes, compared to predictive logarithmic fold changes from edgeR, either using the default value of 0.125 for the edgeR argument prior.count, or after increasing prior.count to 10 (edgeR predFC10).

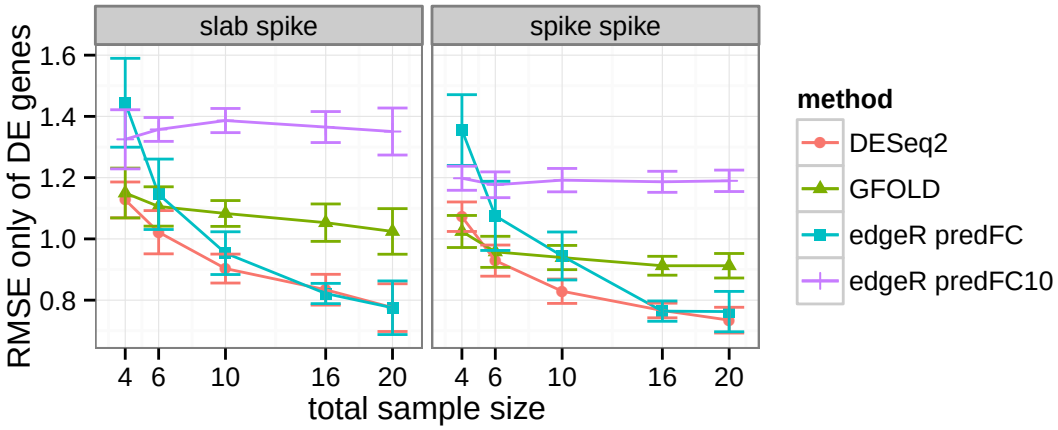


Figure 8: Root mean squared error (RMSE) of logarithmic fold change estimates, only considering genes with non-zero true logarithmic fold change. For the same simulation, shown here is the error only for the 20% of genes with non-zero true logarithmic fold changes (for bell and slab bell all genes have non-zero logarithmic fold change). DESeq2 had generally lower root mean squared error, compared to GFOLD which had higher error for large sample size and edgeR which had higher error for low sample size.

```
    geom = "line") + stat_summary(fun.data = mean_cl_normal, geom = "errorbar") +
    theme_bw() + xlab("total sample size") + ylab("MAE only of DE genes") +
    facet_wrap(~type) + scale_x_continuous(breaks = unique(res$m))
```

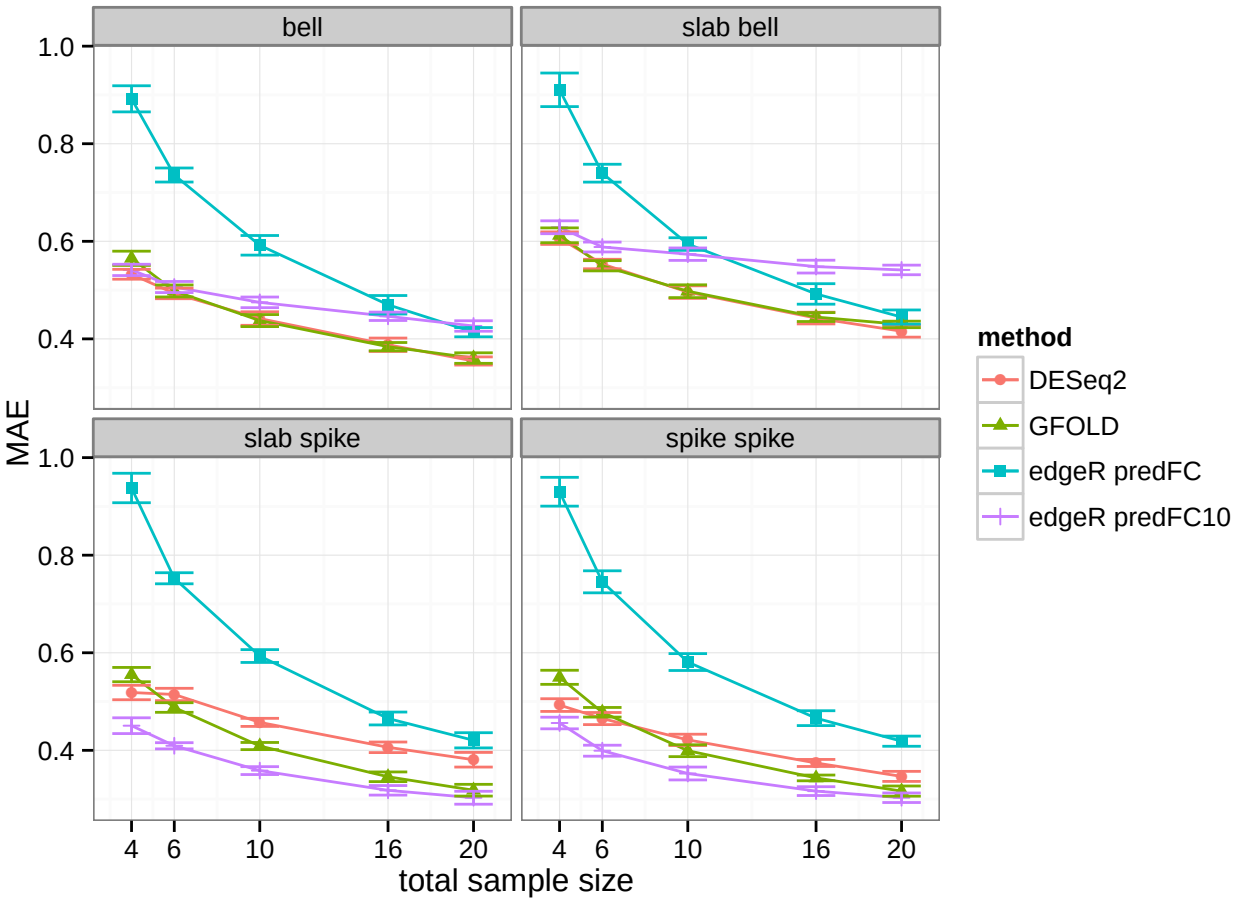


Figure 9: Mean absolute error (MAE) of logarithmic fold change estimates. Results for the same simulation, however here using median absolute error in place of root mean squared error. Mean absolute error places less weight on the largest errors. For the bell and slab bell models, DESeq2 and GFOLD had the lowest mean absolute error, while for the slab spike and spike spike models, GFOLD and edgeR with a prior.count of 10 had lowest mean absolute error.

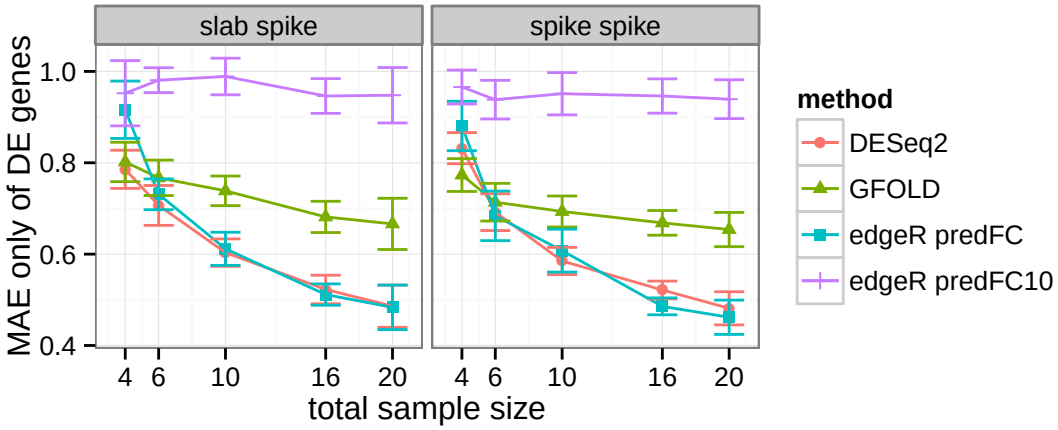


Figure 10: Mean absolute error (MAE) of logarithmic fold change estimates, only considering those genes with non-zero true logarithmic fold change. While in the previous figure, considering all genes for the slab spike and spike spike models, GFOLD and edgeR with a prior.count of 10 had lowest median absolute error, the median absolute error for these methods was relatively large for large sample size, when considering only the 20% of genes with true differentially expression. DESeq2 and edgeR generally had the lowest mean absolute error.

# 4 Transformations and distances for recovery of true clusters

The following simulation evaluated a set of methods for transformation, and for calculating distances betweeen vectors of counts, for their performance in recapturing true clusters in simulated data. Negative Binomial counts were generated in four groups, each with four samples. These groups were generated with 20% of genes given Normally-distributed log fold changes from a centroid. The standard deviation of the Normal for the non-null genes was varied to make the clustering easier or more difficult. The mean of the centroid and the dispersion of the counts were drawn as pairs from the joint distribution of estimates from the Pickrell et al dataset. As the Pickrell dataset has high dispersion (RNA-Seq counts of lymphoblast cells across a population of individuals), simulations were also considered wherein the dispersion was 0.1 and 0.25 times the Pickrell dispersions. Hierarchical clustering with complete linkage was used to separate the samples into four predicted clusters, using a variety of combinations of transformation and distance. These predicted clusters were then compared to the true clusters according to the simulation using the adjusted Rand Index. Furthermore, two variations were considered, one in which the size factors between conditions were equal and one in which the size factors within each group were $[1, 1, \frac{1}{3}, 3]$. The code to generate these results is in `/inst/script/simulateCluster.R`

```
data("results_simulateCluster")
library("ggplot2")
library("Hmisc")
res$sizeFactor <- factor(res$sizeFactor)
levels(res$sizeFactor) <- paste("size factors", levels(res$sizeFactor))
res$dispScale <- factor(res$dispScale)
levels(res$dispScale) <- paste(levels(res$dispScale), "x dispersion")
p <- ggplot(res, aes(x = rnormsd, y = ARI, color = method, shape = method))
p + stat_summary(fun.y = mean, geom = "point", aes(shape = method)) + stat_summary(fun.y = mean,
    geom = "line") + stat_summary(fun.data = mean_cl_normal, geom = "errorbar") +
    facet_grid(sizeFactor ~ dispScale, scale = "free") + theme_bw() + ylab("adjusted Rand Index") +
    xlab("SD of group differences")
```

```
data("results_simulateCluster")
library("ggplot2")
library("Hmisc")
res$sizeFactor <- factor(res$sizeFactor)
levels(res$sizeFactor) <- paste("size factors", levels(res$sizeFactor))
res <- res[res$dispScale == 1, ]
p <- ggplot(res, aes(x = rnormsd, y = ARI, color = method, shape = method))
p + stat_summary(fun.y = mean, geom = "point", aes(shape = method)) + stat_summary(fun.y = mean,
    geom = "line") + stat_summary(fun.data = mean_cl_normal, geom = "errorbar") +
    facet_wrap(~sizeFactor) + theme_bw() + ylab("adjusted Rand Index") + xlab("SD of group differences")
```

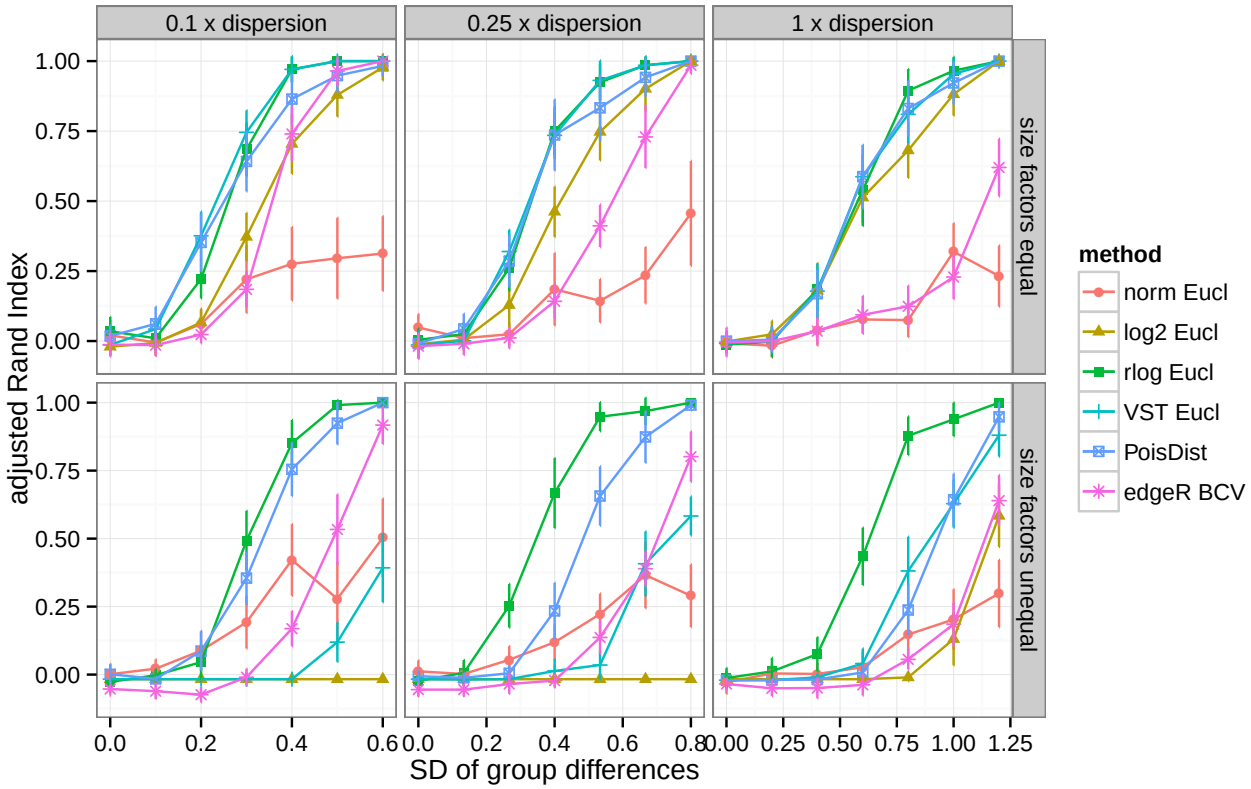


Figure 11: Adjusted Rand Index from clusters using various transformation and distances compared to the true clusters from simulation. The methods assessed were Euclidean distance on counts normalized by size factor, log2 of normalized counts plus a pseudocount of 1, and after applying the rlog and variance stabilizing transformation. Additionally, the Poisson Distance from the PoiClaClu package and the Biological Coefficient of Variation (BCV) distance from the plotMDS function of the edgeR package were used for hierarchical clustering. The points indicate the mean from 20 simulations and the bars are 95 percent confidence intervals. In the equal size factor simulations, the Poisson Distance, variance stablizing transformation (VST), and the rlog transformation had the highest accuracy in recovering true clusters. In the unequal size factor simulations, the size factors for the 4 samples of each group were set to [1, 1, 1/3, 3]. Here, the rlog outperformed the Poisson distance and the VST.

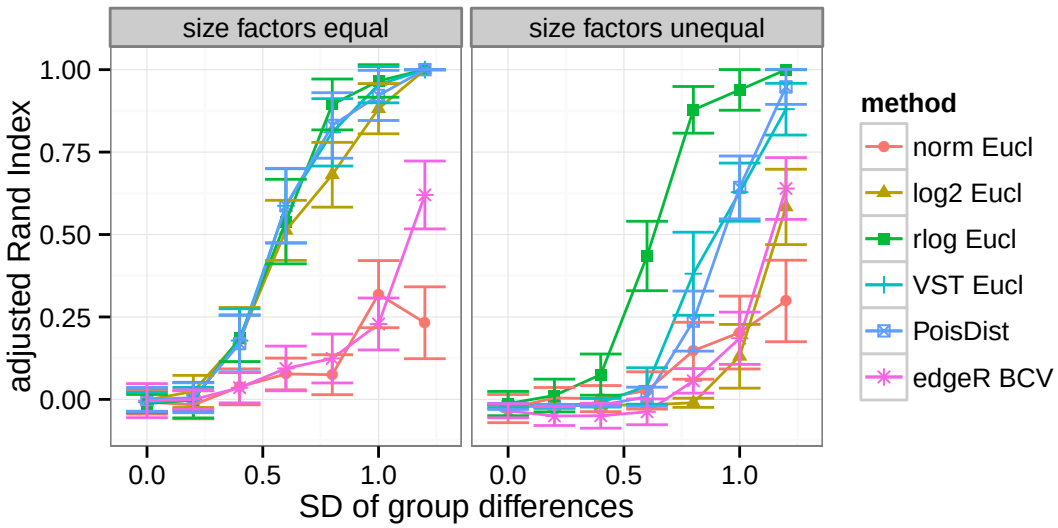


Figure 12: Adjusted Rand Index from clusters using various transformation and distances compared to the true clusters from simulation. The same results as the previous figure, only showing the panels with dispersion equal to the estimates from the Pickrell et al dataset (1 x dispersion).

# 5 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, splines, stats, utils
- Other packages: Biobase 2.24.0, BiocGenerics 0.10.0, DESeq2 1.4.0, DESeq2paper 1.3, Formula 1.1-1, GenomeInfoDb 1.0.0, GenomicRanges 1.16.0, Hmisc 3.14-4, 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, lattice 0.20-29, reshape 0.8.5, schoolmath 0.4, survival 2.37-7, 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, cluster 1.15.2, 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, latticeExtra 0.6-26, 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, stats4 3.1.0, stringr 0.6.2, tools 3.1.0, zlibbioc 1.10.0