Assessment of DESeq2 through simulation

Michael Love, Wolfgang Huber, Simon Anders

Document compiled on:

```
## [1] "2014-10-08 14:00:57 EDT"
```

This document and associated files provide the code and plots for the simulation section of the *DESeq2* paper, so that these results can be easily updated over time. For more details, read the paper at the following URL:

http://dx.doi.org/10.1101/002832

Differential expression analysis

We assessed the sensitivity and specificity of various algorithms using simulation to complement an analysis on real data. The following Negative Binomial simulation samples (mean, dispersion) pairs from the joint distribution of estimated means and dispersions 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 `simulateDE.R` and the code to run each algorithm is in `runScripts.R`.

Note: *DSS* denotes running *DSS* and then Benjamini-Hochberg adjustment on *p*-values. *DSS-FDR* denotes the native FDR estimation of the *DSS* software. *SAMseq* denotes running *SAMseq* with *p*-value estimation and Benjamini-Hochberg adjustment for FDR. *SAMseq-FDR* denotes the native FDR estimation and no *p*-value estimation. *EBSeq* likewise only produces FDR.

```
load("results_simulateDE.RData")
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$algorithm)
resClean <- res[!is.na(res$oneminusspecpvals),]
```

```
library("ggplot2")
p <- ggplot(resClean, 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(-.003,.035)) +
  geom_vline(xintercept=.01) +
  scale_x_continuous(breaks=c(0,.02))
```

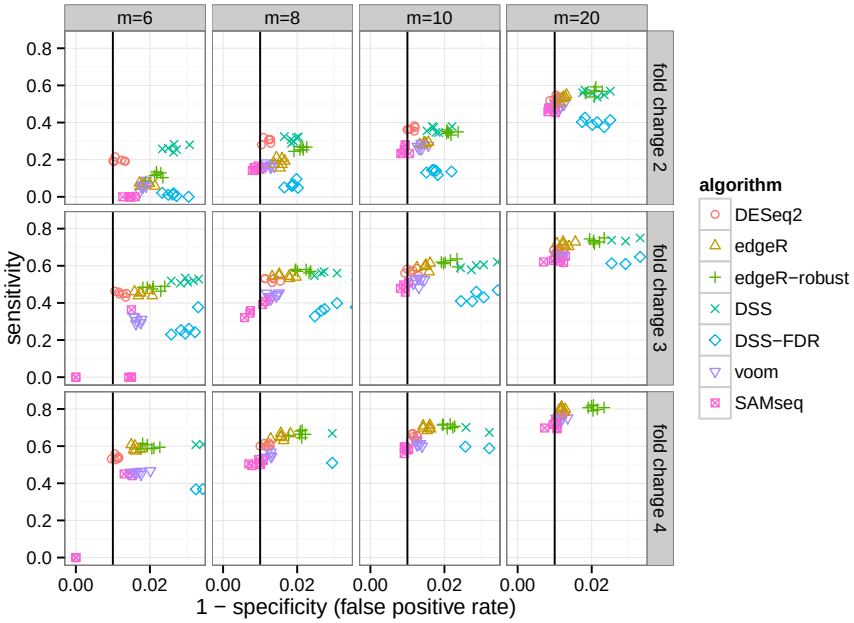

Figure 1: Sensitivity and specificity on simulated datasets.

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.

```
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(-.03, .3)) +
  geom_vline(xintercept=.1)
```

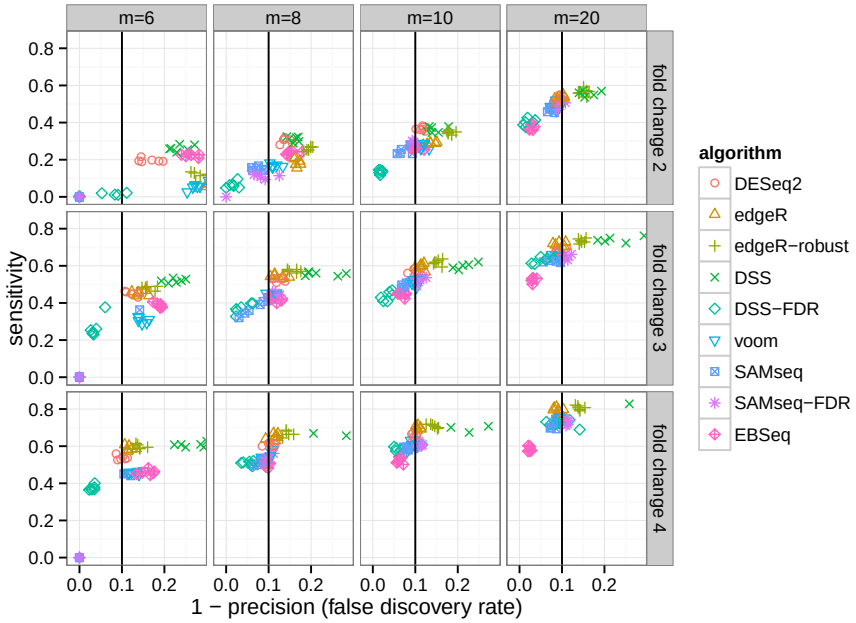

Figure 2: Sensitivity and precision on simulated datasets.

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.

```
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")
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")
```

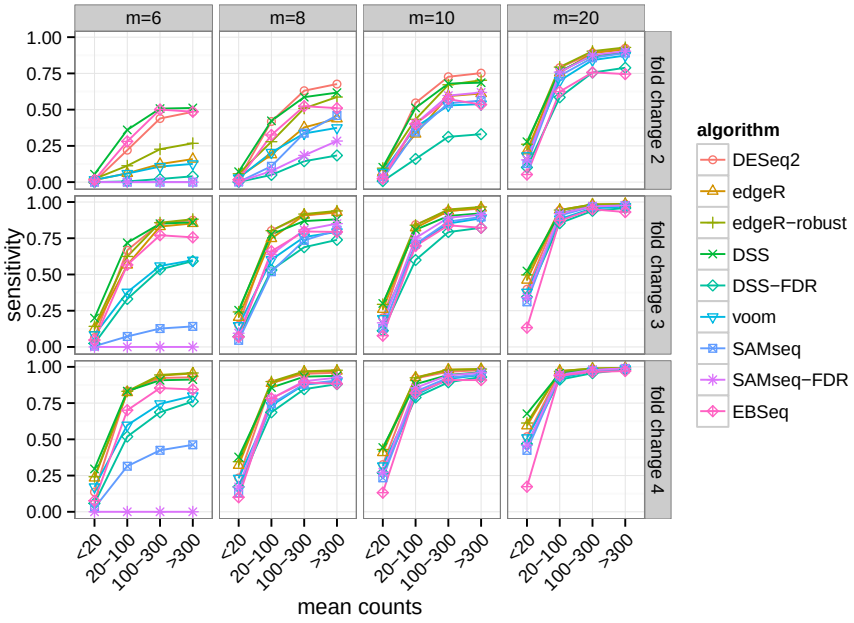

Figure 3: Sensitivity dependence on mean count.

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.

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 `simulateOutliers.R`.

```
load("results_simulateOutliers.RData")
# 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$senspadj < .1,]
resSensPadj <- resSensPadj[nrow(resSensPadj):1,]
resSensPadj <- resSensPadj[!duplicated(with(resSensPadj,
                                            paste(algorithm, m, percentOutlier))),]
summary(resSensPadj$senspadj)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0499  0.0504  0.0934  0.0999
```

```
library("ggplot2")
p <- ggplot(res, aes(x=oneminusspec, y=sensitivity, color=algorithm))
p + scale_x_continuous(breaks=c(0,.1,.2)) +
  scale_y_continuous(breaks=c(0,.2,.4,.6,.8)) +
  geom_line() + theme_bw() +
  facet_grid(m ~ percentOutlier) + xlab("1 - specificity") +
  coord_cartesian(xlim=c(-.03, .25), ylim=c(-.05, .9)) +
  geom_point(aes(x=oneminusspec, y=sensitivity, shape=algorithm),
             data=res[res$precpadj == .1,])
```

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 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).

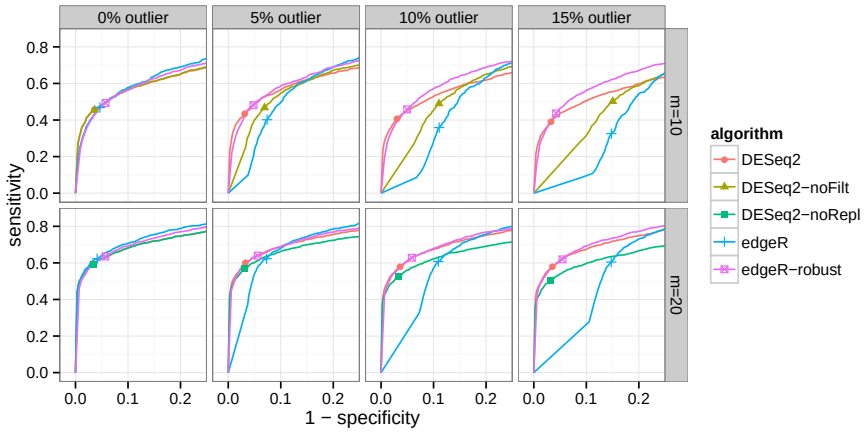

Figure 4: Sensitivity and specificity in presence of outliers.

```
p <- ggplot(res, aes(x=precpadj, y=oneminusprec, color=algorithm))
p + scale_x_continuous(breaks=c(0,.1,.2)) +
  scale_y_continuous(breaks=c(0,.1,.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(-.03, .25), ylim=c(-.05, .25)) +
  geom_point(aes(x=precpadj, y=oneminusprec, shape=algorithm),
             data=res[res$precpadj == .1,])
```

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.

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 `simulateLFCAccuracy.R`.

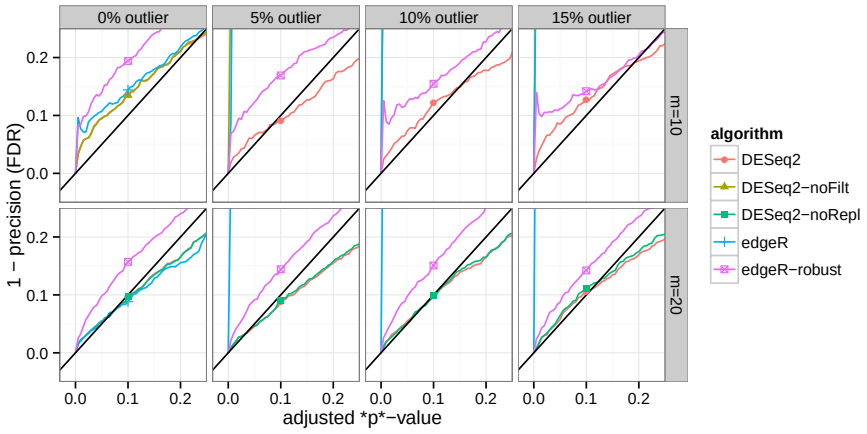

Figure 5: FDR and target FDR in presence of outliers.

```
par(mfrow=c(2,2),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
myhist <- function(x, ...) hist(x, breaks=brks, border="white",
                                col="blue", xlab="", ylab="", ...)
myhist(trimit(rnorm(n)), main="bell")
myhist(trimit(c(rnorm(n * 8/10), runif(n * 2/10, -4, 4))), main="slab bell")
myhist(c(rep(0, n * 8/10), runif(n * 2/10, -4, 4)), main="slab spike")
myhist(c(rep(0, n * 8/10), sample(c(-2, 2), n * 2/10, TRUE)), main="spike spike")
```

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.

```
load("results_simulateLFCAccuracy.RData")
library("ggplot2")
library("Hmisc")
p <- ggplot(data=res, aes(x=m, y=RMSE, color=method, shape=method))
```

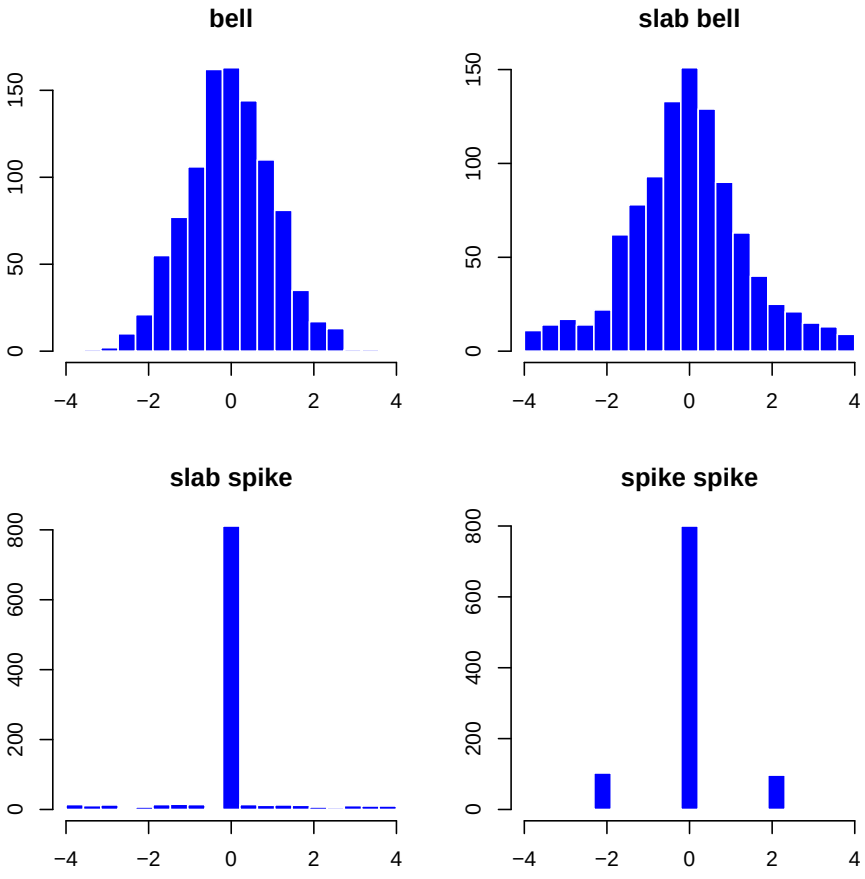

Figure 6: Examples of simulated log2 fold changes.

```
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))
```

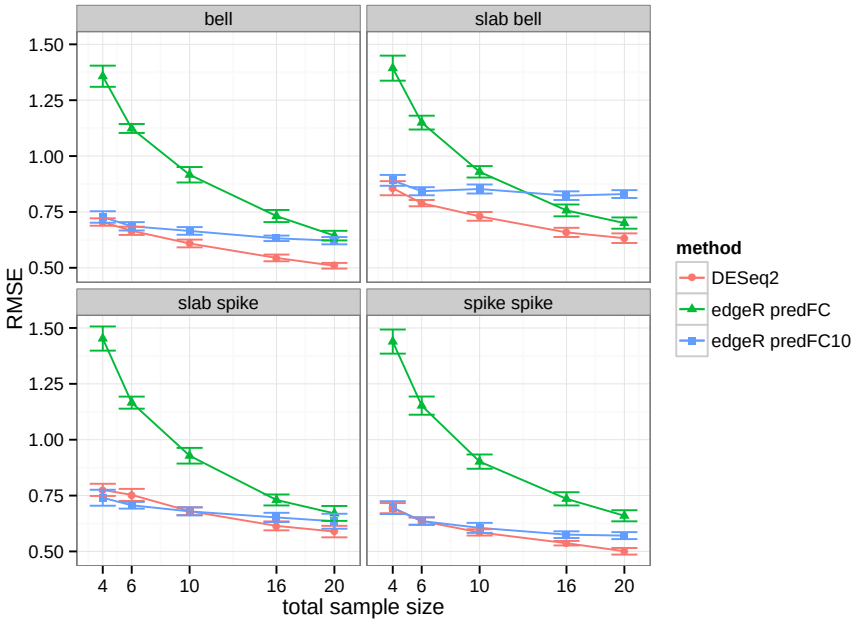

Figure 7: Root mean squared error in estimating log2 fold changes.

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.

```
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))
```

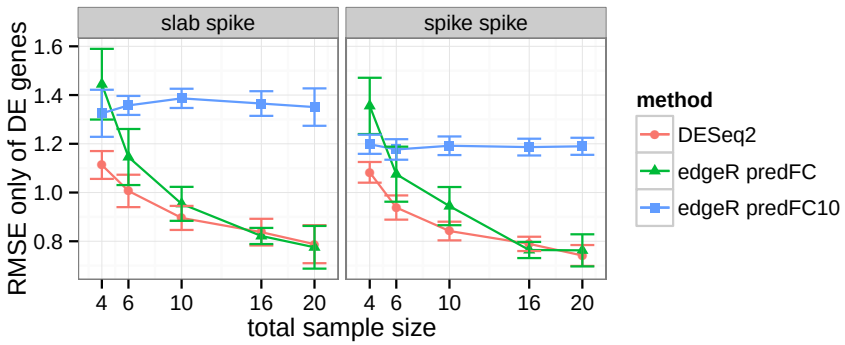

Figure 8: Root mean squared error for only differentially expressed genes.

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).

```
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))
```

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.

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

Mean absolute error (MAE) of logarithmic fold change estimates, only considering those genes with non-zero true logarithmic fold change.

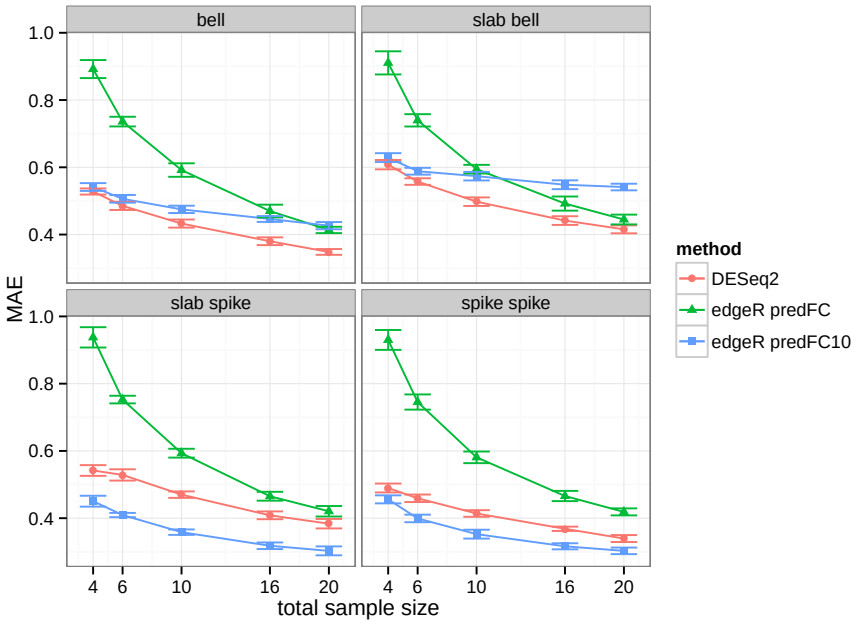

Figure 9: Mean absolute error in estimating log2 fold changes.

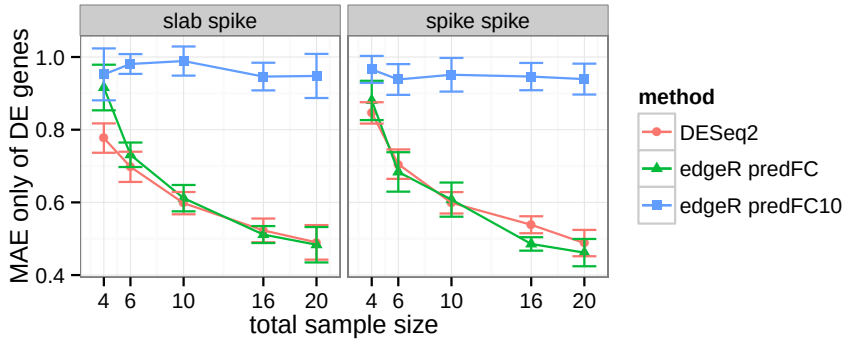

Figure 10: Mean absolute error for only differentially expressed genes.

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, 1/3, 3]. The code to generate these results is in `simulateCluster.R`.

```
load("results_simulateCluster.RData")
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")
```

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 was used for hierarchical clustering. The points indicate the mean from 20 simulations and the bars are 95 percent confidence intervals.

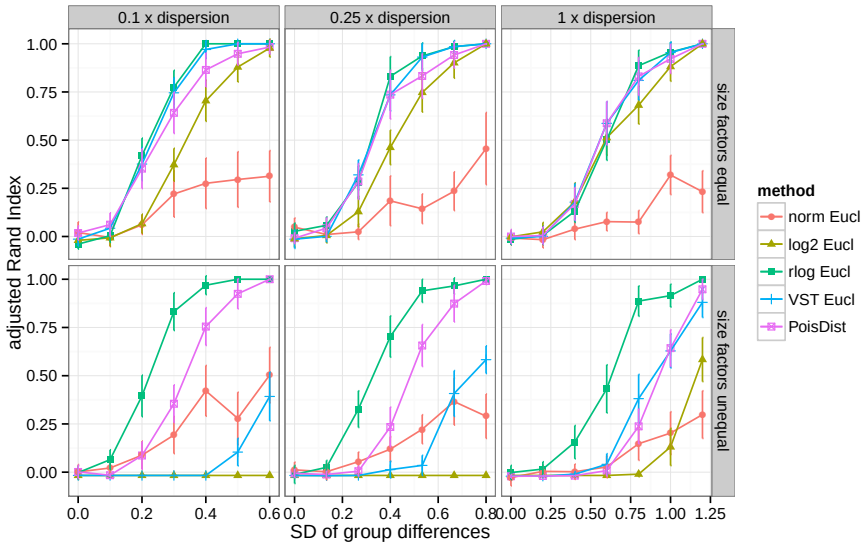

Figure 11: Clustering accuracy over the size of group differences.

Genes expressed in only one condition

As discussed by Robinson and Smyth and by Rapaport et al., it is desirable that, in the situation in which a gene is only expressed in one condition, the statistical significance increase with the signal to noise ratio of expression in the expressed condition. For example, Rapaport et al. plot the $-\log_{10}(p)$ for p values over the signal to noise ratio. In the following code chunk we demontrate that *DESeq2* has increasing $-\log_{10}(p)$ in a comparison of two conditions in which one group has all zero counts, e.g.: $\{0, 0, 0\}$ vs $\{10, 10, 10\}$.

```
m <- 6
disp <- 0.5
ii <- seq(from=1,to=4,length=7)
coldata <- DataFrame(x=factor(rep(1:2,each=m/2)))
pvals <- sapply(ii, function(i) {
  mat <- matrix(as.integer(c(rep(0,m/2),rep(10^i,m/2))),nrow=1)
  dds <- DESeqDataSetFromMatrix(mat, coldata, ~ x)
  sizeFactors(dds) <- rep(1,m)
  dispersions(dds) <- 0.5
  results(nbinomWaldTest(dds))$pvalue
})
plot(10^ii, -log10(pvals), log="x", type="b", xaxt="n",
     xlab="group 2 counts", ylab=expression(-log[10](p~value)))
axis(1,10^(1:4),10^(1:4))
```

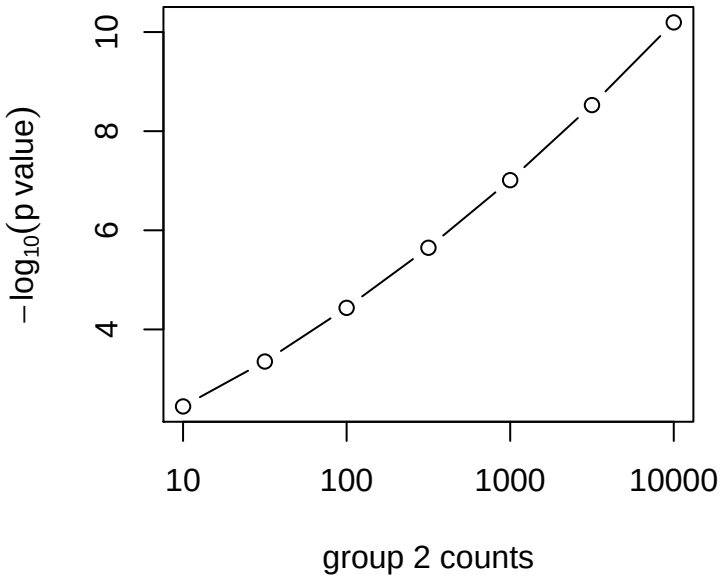

Figure 12: Simulated gene expressed in only one condition

Session information

The session information saved in the `simulateDE` script:

```
sessInfo
```

```
## R Under development (unstable) (2014-07-10 r66124)
## Platform: x86_64-unknown-linux-gnu (64-bit)
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
##  [1] splines   stats4    parallel  stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] BiocParallel_0.99.22      futile.logger_1.3.7
##  [3] EBSeq_1.5.5               gplots_2.14.2
##  [5] blockmodeling_0.1.8       DSS_2.3.0
##  [7] locfdr_1.1-7              samr_2.0
##  [9] matrixStats_0.10.0        impute_1.39.0
## [11] edgeR_3.7.18              limma_3.21.19
## [13] DESeq2_1.5.73             RcppArmadillo_0.4.450.1.0
## [15] Rcpp_0.11.3               GenomicRanges_1.17.42
## [17] GenomeInfoDb_1.1.23       IRanges_1.99.28
## [19] S4Vectors_0.2.4           Biobase_2.25.0
## [21] BiocGenerics_0.11.5       devtools_1.6
## [23] knitr_1.6                 BiocInstaller_1.15.5
##
## loaded via a namespace (and not attached):
##  [1] acepack_1.3-3.3       annotate_1.43.5       AnnotationDbi_1.27.16
##  [4] base64enc_0.1-2       BatchJobs_1.4         BBmisc_1.7
##  [7] bitops_1.0-6          brew_1.0-6            bsseq_1.1.2
## [10] caTools_1.17.1        checkmate_1.4         cluster_1.15.3
## [13] codetools_0.2-9       colorspace_1.2-4      DBI_0.3.1
## [16] digest_0.6.4          evaluate_0.5.5        fail_1.2
## [19] foreach_1.4.2         foreign_0.8-61        formatR_0.10
## [22] Formula_1.1-2         futile.options_1.0.0  gdata_2.13.3
## [25] genefilter_1.47.6     geneplotter_1.43.0    ggplot2_1.0.0
```

```
## [28] grid_3.2.0            gtable_0.1.2          gtools_3.4.1
## [31] Hmisc_3.14-5          iterators_1.0.7       KernSmooth_2.23-13
## [34] lambda.r_1.1.6        lattice_0.20-29       latticeExtra_0.6-26
## [37] locfit_1.5-9.1        MASS_7.3-35           munsell_0.4.2
## [40] nnet_7.3-8            plyr_1.8.1            proto_0.3-10
## [43] RColorBrewer_1.0-5    RCurl_1.95-4.3        reshape2_1.4
## [46] R.methodsS3_1.6.1     rpart_4.1-8           RSQLite_0.11.4
## [49] scales_0.2.4          sendmailR_1.2-1       stringr_0.6.2
## [52] survival_2.37-7       tools_3.2.0           XML_3.98-1.1
## [55] xtable_1.7-4          XVector_0.5.8
```