# Generalised empirical Bayesian methods for discovery of differential data in high-throughput biology: Supplemental Materials

Thomas J. Hardcastle [*][†]

## S1 Bootstrapping the weights on $\Theta_q$

For a given model defined by some set of equivalence classes, the distribution of parameters is estimated by sampling (without replacement) biomolecular events, and from the observed data for each biomolecular event $h$, estimating parameter sets $\eta_q^h$ for each equivalence class $E_q$. This allows the estimation of the likelihood of observed data given the model (Eqn. 4) and hence of posterior likelihoods of the model given the data.

However, if the distribution of $\theta_q$ is dissimilar from model to model, it may be advantageous to weight a sampled $\eta_q^h$ by the likelihood that the sampled biomolecular event $h$ is representative of a given model $M$. In this case, Eqn. 4 becomes Eqn. 8; that is,

$$\mathbb{P}(D_c \mid M) = \prod_q \frac{1}{\sum_h v_M^h} \sum_h v_M^h \prod_{D_{cj} \in D_{cE_q}} \mathbb{P}(D_{cj} \mid \eta_q^h)$$

where for model $M$ the estimated $\eta_q^h$ are weighted by $v_M^h$.

Since the weights $v_M^h$ represent the likelihood that the model $M$ applies to biomolecular event $h$, these can be acquired through estimation of the posterior likelihoods; $v_M^h = \mathbb{P}(M \mid D_h)$. Using these estimated weights in Eqn. 8 allows the posterior likelihoods to be updated; consequently, the weights used to refine the estimated distribution of parameters may be updated. This process may be repeated over a number of cycles to bootstrap to a set of refined weight estimates; convergence of posterior likelihoods to within small variations is likely, though not guaranteed for all biomolecular events under all distributional choices.

Two basic strategies are proposed for initial estimates of $v_M^h$. In the case where each model is defined by a unique set of equivalence classes, the posterior likelihoods can be estimated from Eqn. 3 and used as the first estimates for $v_M^h$. Where two or more models have identical equivalence class definitions (see Section S2) a partition of the data whereby $v_M^h$ is initially either 0 or 1 may be used to establish initial estimates of $v_M^h$ before bootstrapping a set of refined estimates.

[*] to whom correspondence should be addressed: tjh48@cam.ac.uk

[†] Department of Plant Sciences, University of Cambridge, Downing Street, Cambridge CB2 3EA, United Kingdom

# S2 Qualitative differences between subsets of biomolecular events

In high-throughput sequencing experiments, it is not uncommon to find a subset of biomolecular events that are qualitatively different from the remainder. In mRNA-Seq data, we expect a set of non-expressed genes to which only a small number of reads are assigned, for reasons such as sequencing error, misalignment or very low background levels of expression. Figure S16 shows the distribution of the log of the parameter associated with mean expression in RNA-seq data, assumed to be distributed negative binomially and equivalently across all samples. The tail of data to the left of the modal peak may be considered to represent non-expressed genes.

To distinguish between such qualitatively different events, we can construct additional models in `baySeq` v2. In the example above, we construct one model ($M_{NDE}$) for expressed but non-differentially expressed genes, and one model ($M_{NE}$) for non-expressed genes. These two models are identical in terms of their equivalence classes, but will differ in the assumed hyperdistribution.

Two principal options exist for varying the assumed hyperdistributions between models that share the same equivalence classes. Firstly, since the purpose of the two models is to separate two qualitatively different sets of biomolecular events, we may find some function of the values in $\Theta_q$ that splits the data. The data shown in Figure S16 can be split by minimising the intra-class variance [3]. Sampled values mapping to the left of the threshold indicated by the vertical red line represent the distribution of data for $M_{NE}$ while those to the right represent the distribution of data for $M_{NDE}$.

In some cases, the distinction between two quantitatively different models for gene expression introduces a natural variation between the hyperdistributions. For example, in paired data, a substantial proportion of the data may be equivalently expressed within all pairs, and this may be regarded as a qualitatively different scenario to equivalent expression across replicates but divergent expression within each pair. We have previously shown [2] that these cases can be analysed by constructing a model for equivalent expression across replicates. Assuming a beta-binomial distribution with parameters $p$, the proportion of counts observed in the first member of each pair, and $\phi$, the dispersion, a set $\Theta_q$ can be constructed by maximum likelihood methods, as discussed in Section 2. We can then construct a second model describing equivalent expression within pairs in which the calculated values for $\phi$ are used for the dispersions but in which the values for $p$ are set to 0.5, the value which corresponds to a hypothesis of balanced expression between pairs.

We simulate a set of data following [1, 5] in which data from ten thousand genes in ten samples are simulated from a negative binomial distribution, with means sampled from a SAGE dataset. Dispersions for each gene are sampled from a gamma distribution with shape = 0.85 and scale = 0.5. Library sizes for each sample are sampled from a uniform distribution between 30000 and 90000. One thousand of the genes are simulated to have an eight-fold differential expression in either direction between the first and second sets of five samples each. A further one thousand genes have their mean expression reduced by a factor of twenty; these represent a set of unexpressed genes within the data.

The parameters of an assumed negative-binomial distribution are $\mu_q^h, \phi^h$, where $\mu_q^h$ represents the estimated mean (scaled for library size) for some equivalence class $q$ estimated by sampling some genomic event $h$, and $\phi^h$ the similarly estimated dispersion. An initial weighting on these parameters for a model $M_{NE}$ of no expression is acquired by considering the log of the $\mu_q^h$ estimated in a model of no differential expression. Figure S17 shows the distribution of this random variable and the threshold $\psi$ which, if used to split this variable, minimises the intra-class variance. For a model $M_{NE}$ of no expression, the initial weights used are $v_{M_{NE}}^h = 1$ if $\log \mu_q^h < \psi$ and 0 otherwise, while for all other models, the weights used are $v_M^h = 1 - v_{M_{NE}}^h$.

We then iteratively use Eqn 8 to update these weightings and improve the posterior likelihoods acquired for each model. At each iteration, $v^h_M = 1 - v^h_{M_{NE}}$ (the current estimate of the posterior likelihood that gene $h$ is not expressed) and $v^h_M = 1 - v^h_{M_{NE}}$ for all other models. Figure S2 shows the performance of these methods in simultaneously identifying non-expressed and differentially expressed genes in these simulations.

## S3 Variable Model Priors

Figure S18 shows a reanalysis of the simulated data used in Soneson *et al* [6]. This figure is derived from simulated data equivalent to 12450 genes from 10 samples, of which approximately 1250 are differentially expressed between the first five and second five samples. In one set of simulations, the differentially expressed genes are equally likely to be up-regulated as down-regulated between the two groups in the data, while in the other, all differential expression is up-regulation of the second group relative to the first. Allowing `baySeq v2` to choose different model priors depending on which group has higher average expression gives a substantial increase in performance in the unbalanced case, while not affecting performance for the balanced data.

## S4 Simulation of zero-inflated negative binomial data

We base the simulation of these data on previous simulations developed to generate high-throughput sequencing data [1, 5] in which data from ten thousand genes in ten samples are simulated from a negative binomial distribution, with means sampled from a SAGE dataset. Increased sequencing depth can be explored by scaling these sampled means. Dispersions for each gene are sampled from a gamma distribution with shape = 0.85 and scale = 0.5. Library sizes for each sample are sampled from a uniform distribution between 30000 and 90000. One thousand of the genes are simulated to have an eight-fold differential expression in either direction between the first and second sets of five samples. For each gene, we then sample a proportion $p_c$ of zero-inflation from a uniform distribution between 0 and 0.5, and for each sample in that gene, replace the observed value with a zero with probability $p_c$.

# S5 Supplementary Figures

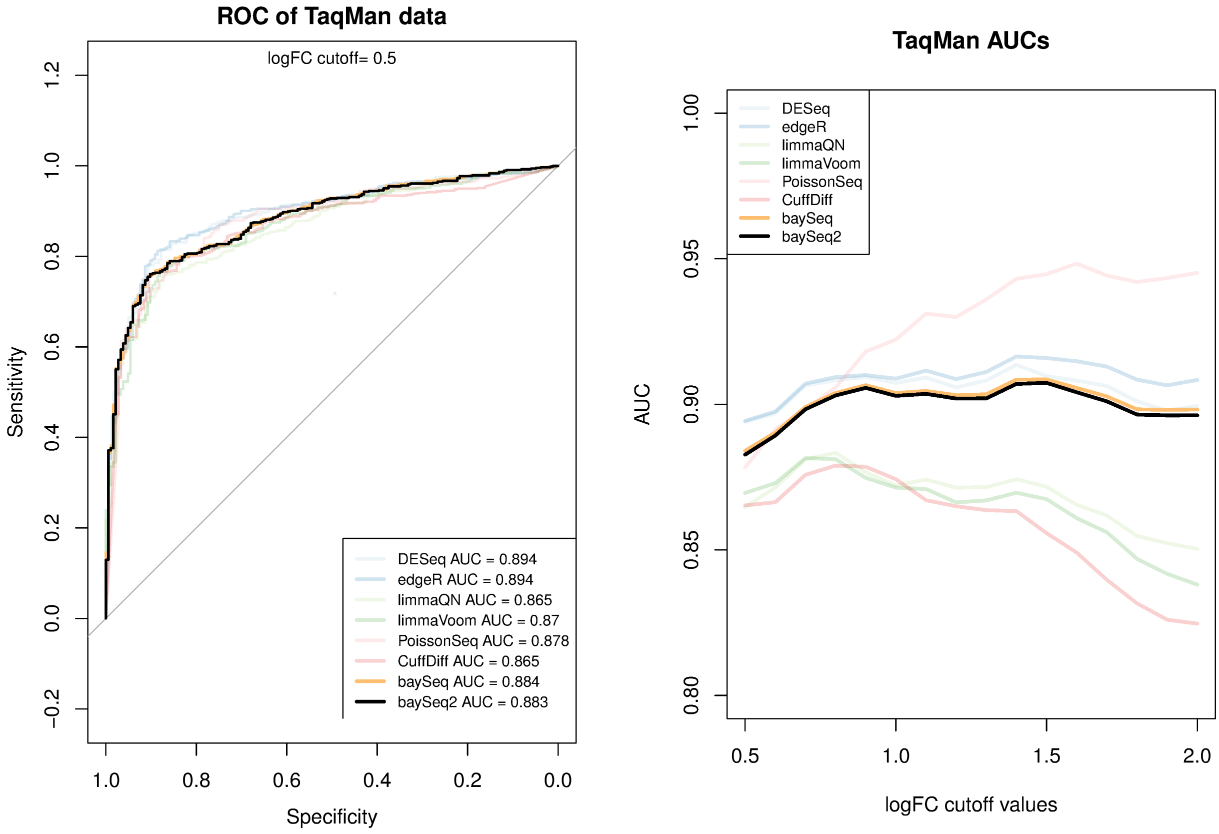

Figure S1: A reanalysis of the qRT-PCR validated gene TaqMan set described in [4], showing equal performance between the implementation of `baySeq` used in [4] and a negative-binomial implementation of `baySeq v2`. This figure reproduces (save for the addition of the `baySeq v2` curves) Fig. 2 of the [4] manuscript using source code made available by the authors.

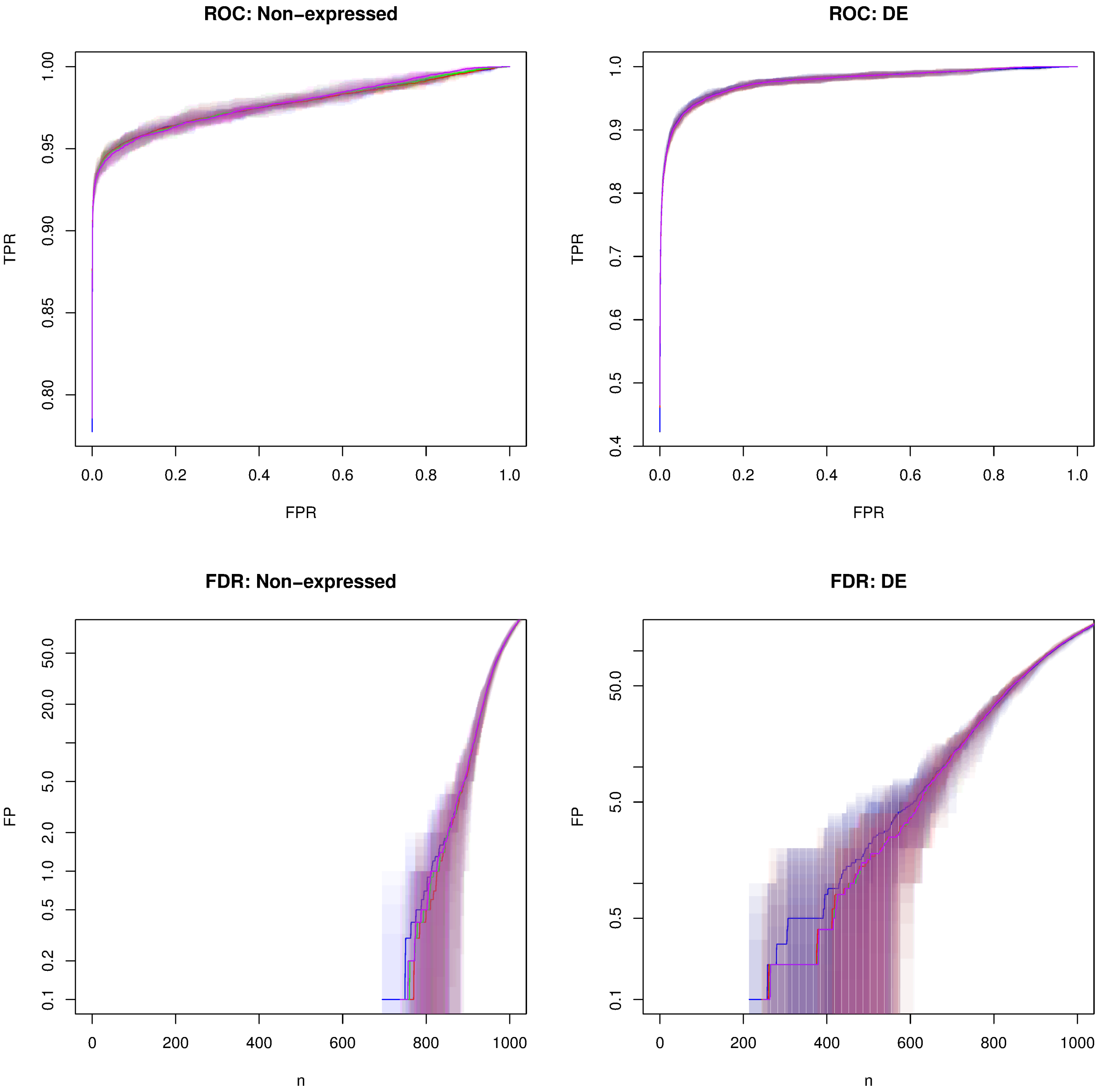

Figure S2: Mean ROC and FDR curves for discovery of non-expressed and differentially expressed data in simulation studies, with no bootstrapping (blue), and two (red), five (green) and ten (purple) cycles of bootstrapping. Percentiles of true discovery rates (for ROC curves) and false discovery rates (for FDR curves) across simulations are shown as transparent areas around curves.

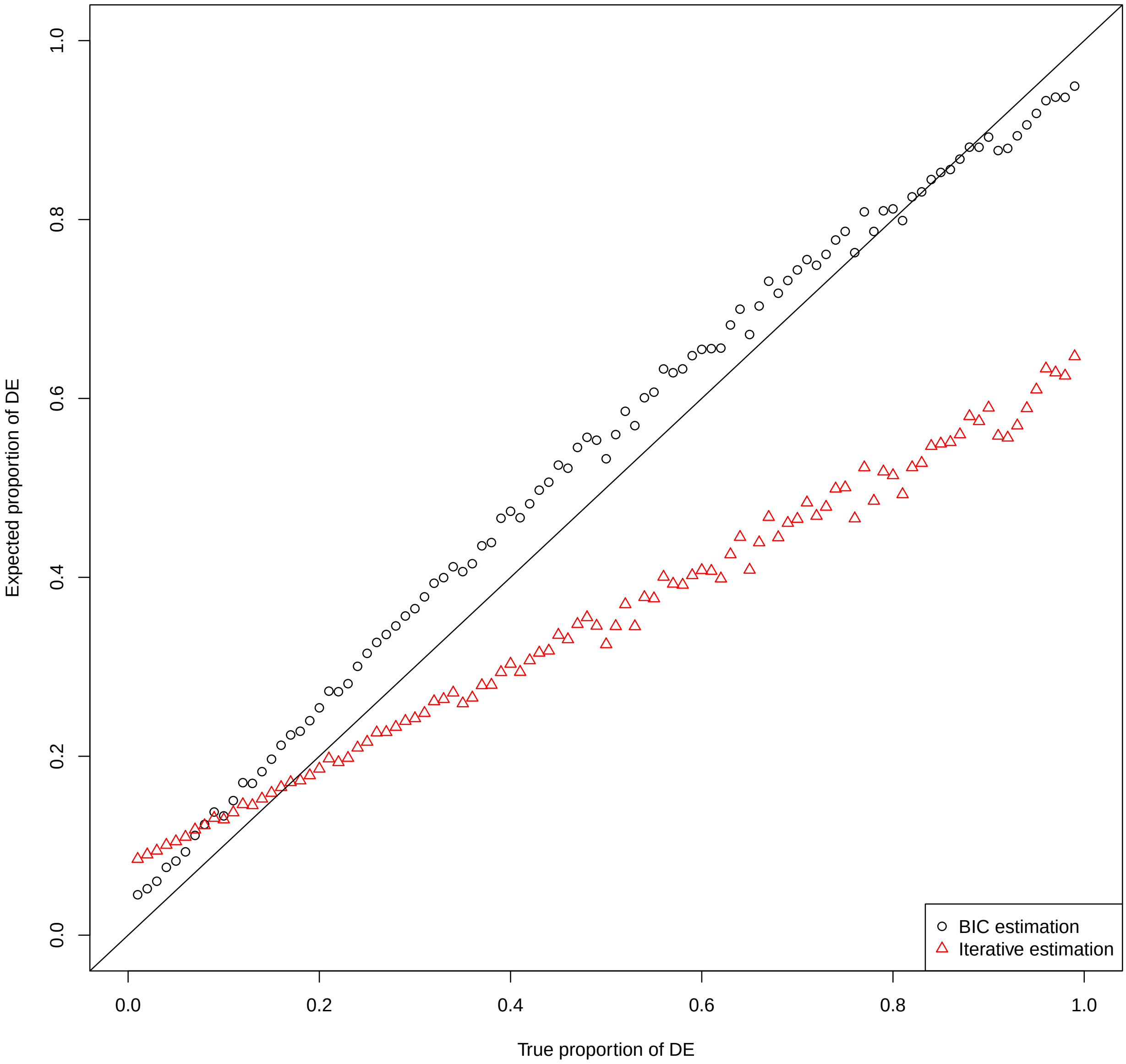

Figure S3: The expected proportion of differentially expressed genes against the true proportion in simulation studies following [1, 5] with 10 libraries. The expected proportion of differentially expressed genes was calculated by summing the posterior likelihoods of differential expression, and dividing by the total number of genes. Model priors were calculated using the BIC method described in 2 or the iterative method described in Hardcastle (2010) [1].

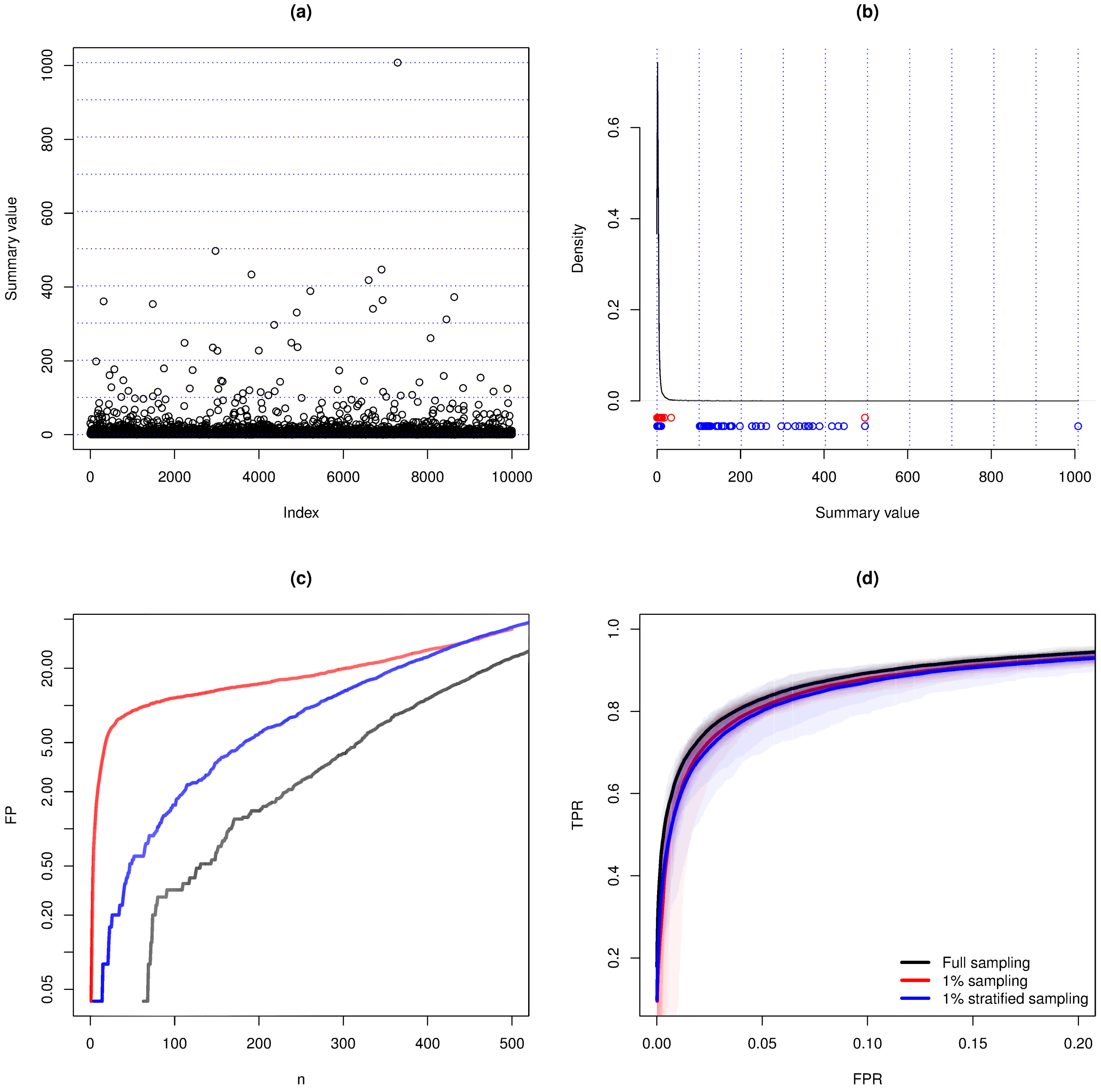

Figure S4: An example of stratified sampling for detection of differential expression in simulation studies following [1, 5] with 10 libraries and 10000 'genes'. In (a), the summary value of average count across samples is plotted against row index; dashed blue lines indicate the strata from which values are sampled. In (b), the density of the average count is shown with strata indicated by dashed vertical blue lines. Red dots indicate 1% (100 genes) sampling, while blue dots indicate a stratified sampling of 1%. In (c), average false discoveries (over twenty-five simulations) are shown against the number of selected genes for full sampling (data on all genes are used), for a 1% sampling, and for a 1% stratified sampling. In (d), average ROC curves (over twenty-five simulations) are shown for full, 1% and 1% stratified samplings.

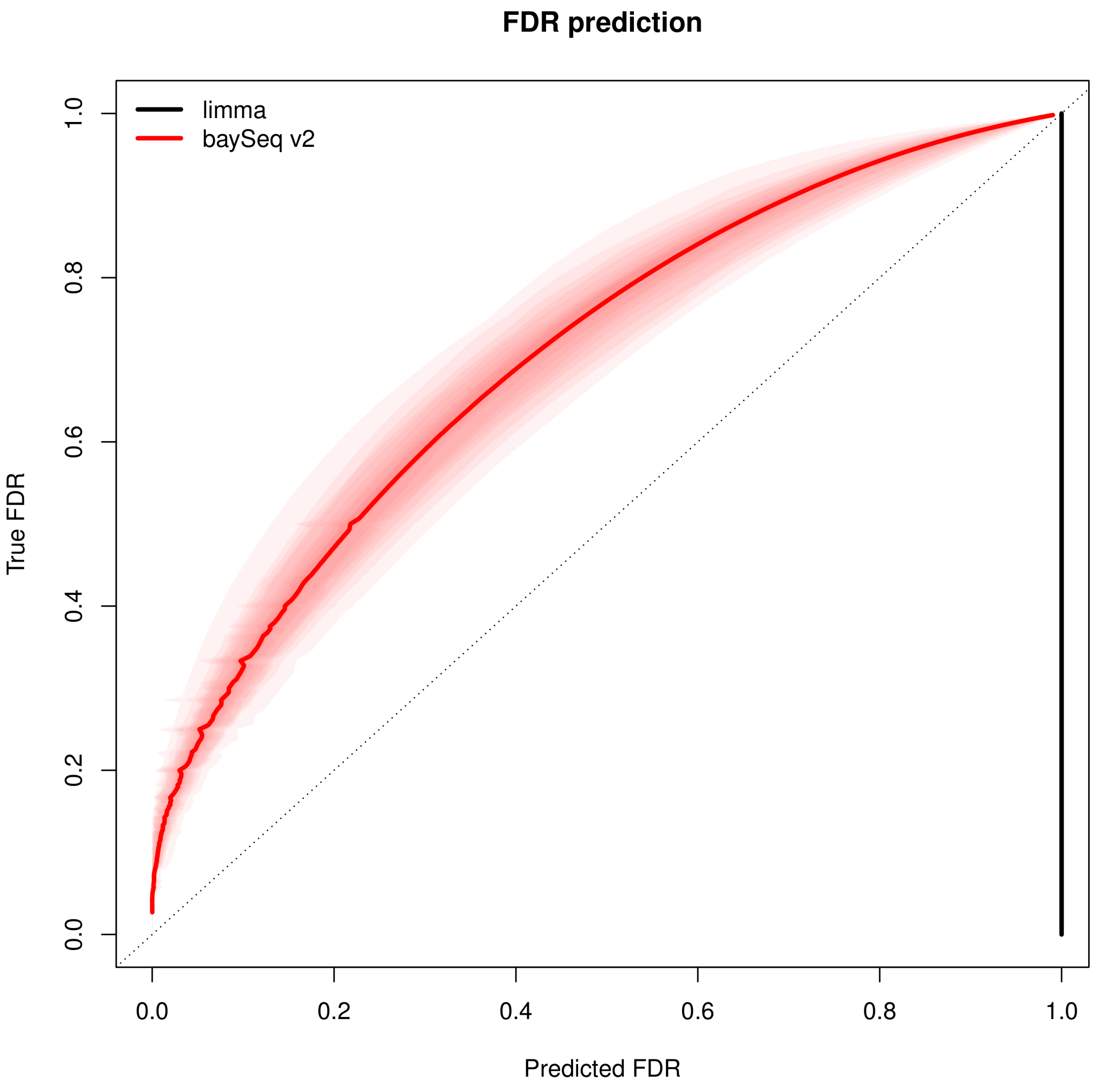

Figure S5: Average predicted FDRs against true FDRs for `baySeq v2` and `limma` analyses of samplings of non-overlapping pairs of hybridisations in the Affymetrix HGU133A Latin Square data. `baySeq v2` tends to underestimate the true FDR for these data, whereas `limma` predicts no true positives in any analysis. Percentiles of predicted FDRs across samplings are shown as transparent areas around curves.

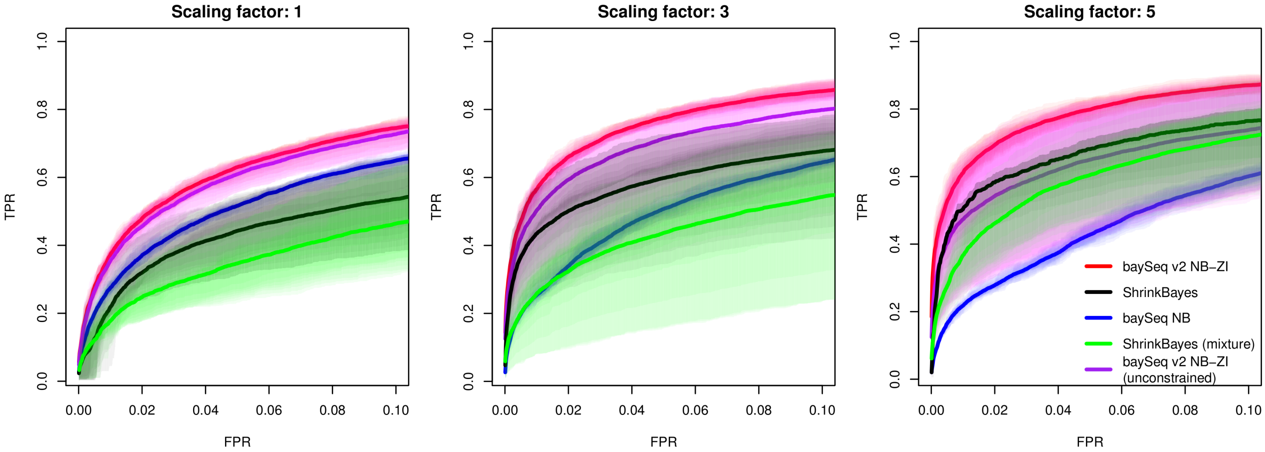

Figure S6: Average ROC curves for `baySeq v2` (zero-inflated negative binomial), ShrinkBayes, `baySeq v2` (negative binomial), ShrinkBayes with mixture model priors and `baySeq v2` (zero-inflated negative binomial with unconstrained domain in prior estimation) analyses of differential expression in zero-inflated negative binomially distributed data. Library scaling factors are increased by factors of 1, 3, and 5. Percentiles of true positive rates across samplings are shown as transparent areas around curves.

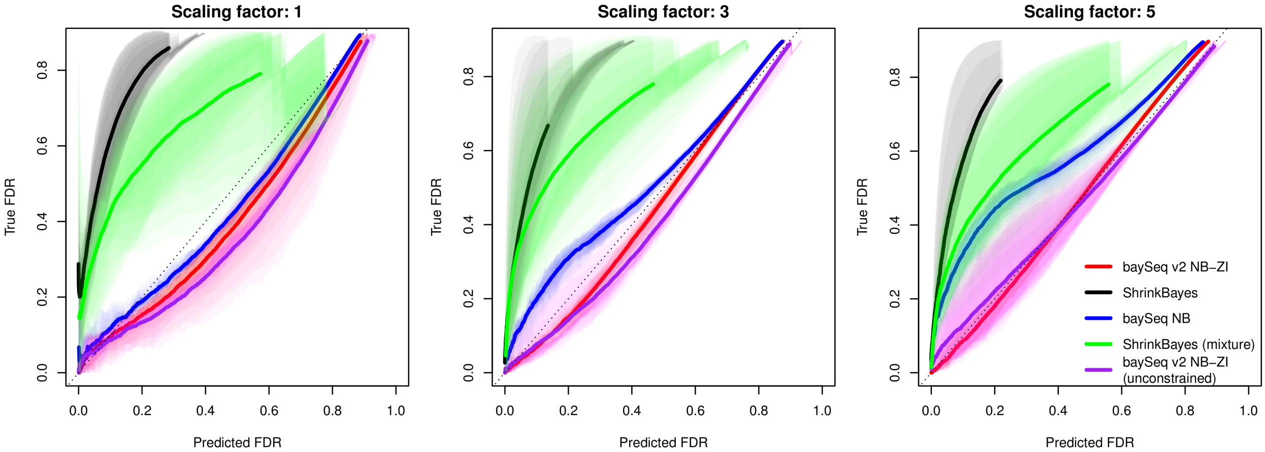

Figure S7: Average predicted FDRs against true FDRs for `baySeq v2` (zero-inflated negative binomial), ShrinkBayes, `baySeq v2` (negative binomial), ShrinkBayes with mixture model priors and `baySeq v2` (zero-inflated negative binomial with unconstrained domain in prior estimation) analyses of differential expression in zero-inflated negative binomially distributed data. Library scaling factors are increased by factors of 1, 3, and 5. Percentiles of predicted FDRs across samplings are shown as transparent areas around curves.

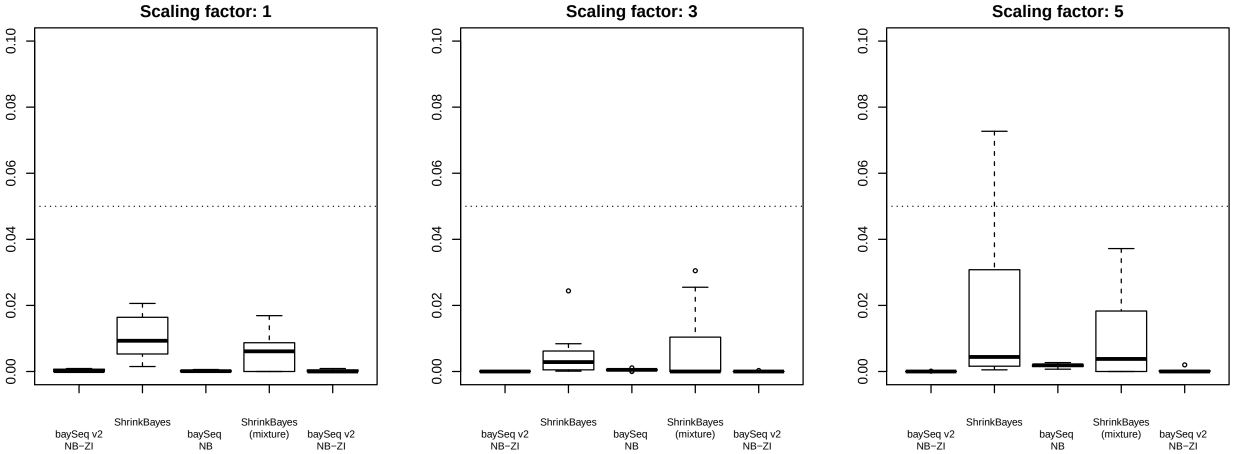

Figure S8: Proportions of false discoveries chosen with an FDR threshold of 0.05 for `baySeq v2` (zero-inflated negative binomial), ShrinkBayes, `baySeq v2` (negative binomial), ShrinkBayes with mixture model priors and `baySeq v2` (zero-inflated negative binomial with unconstrained domain in prior estimation) analyses of differential expression in zero-inflated negative binomially distributed data with no truly differentially expressed data. Library scaling factors are increased by factors of 1, 3, and 5.

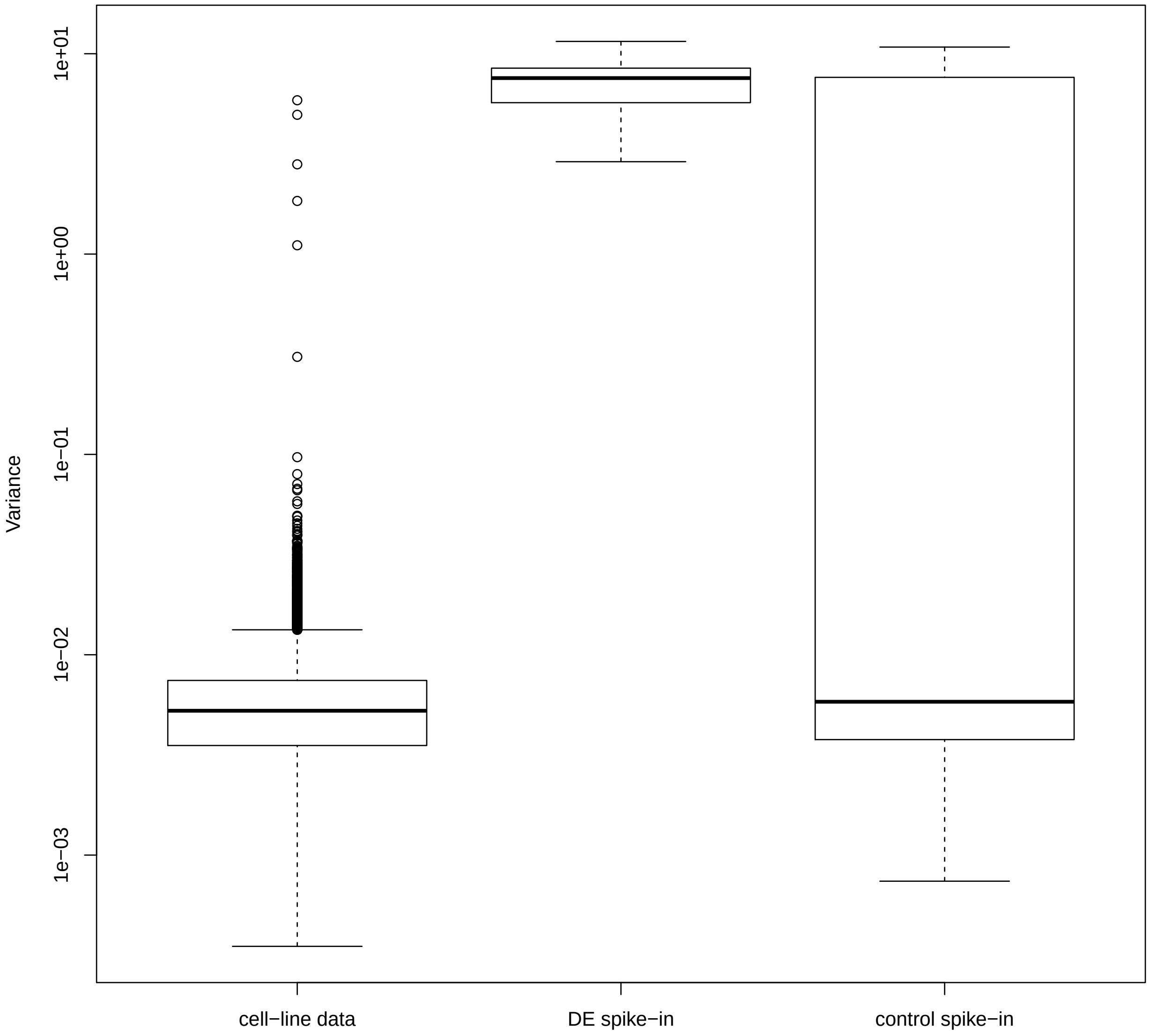

Figure S9: Gene/spike-in variances of expression across all arrays in the Affymetrix HGU133A Latin Square data. Differentially expressed spike-ins have much higher variance than data describing gene expression in cell-lines, as expected. Variance in the non-differentially expressed control spike-ins can be as high as that seen in differentially expressed data, suggesting that these should be removed from differential expression analyses.

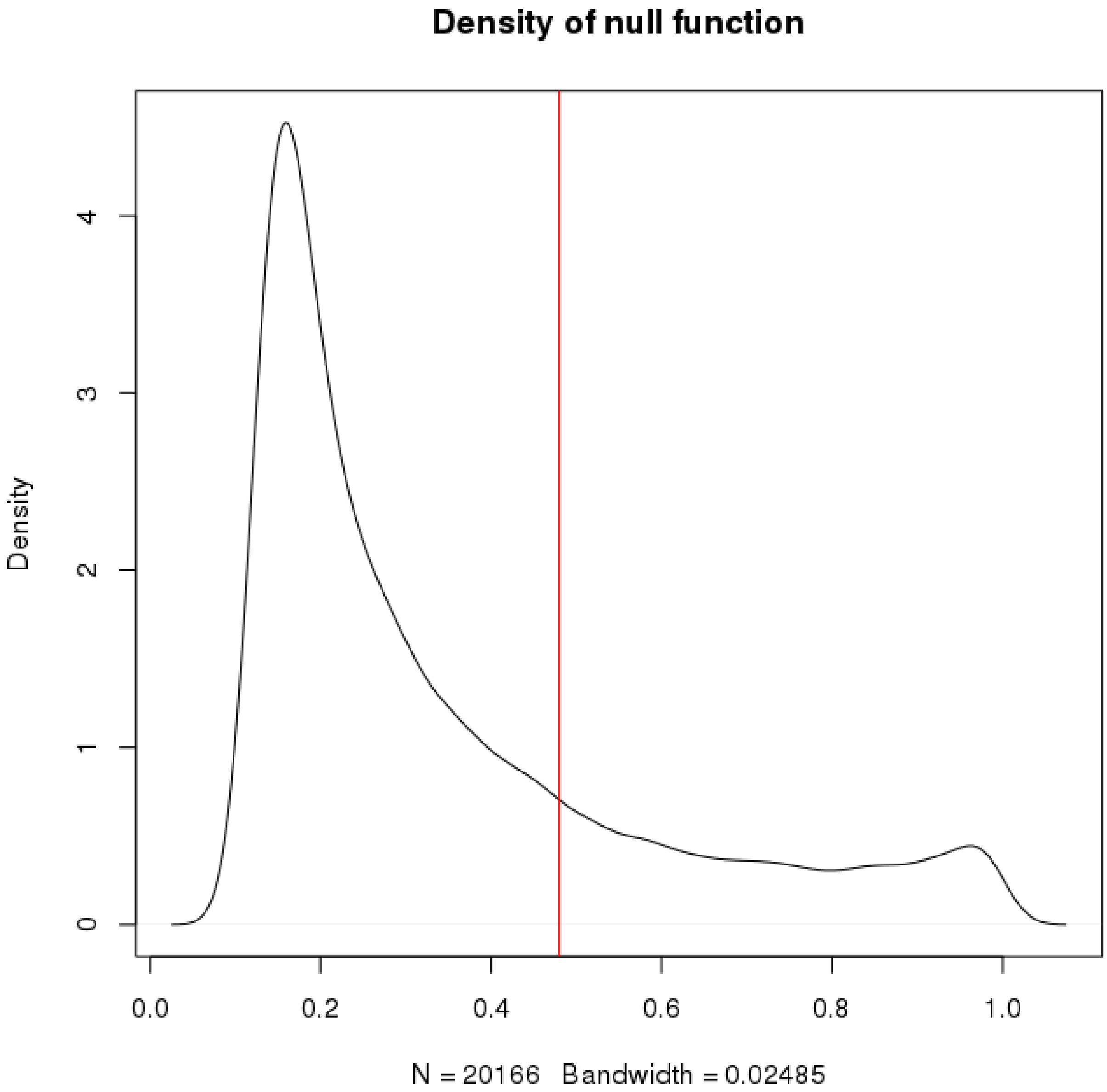

Figure S10: Distribution of $p_{q1}$ in a model of consistent expression between age groups for the observed expression from from ten tissue types (adrenal gland, brain, heart, kidney, liver, lung, muscle, spleen, thymus, and uterus) in female rats, comparing four juvenile (2-week old) to four aged (104-week old) individuals in the Rat BodyMap project [7]. The threshold (red) is chosen to minimise the intra-class variance of the partitioned data.

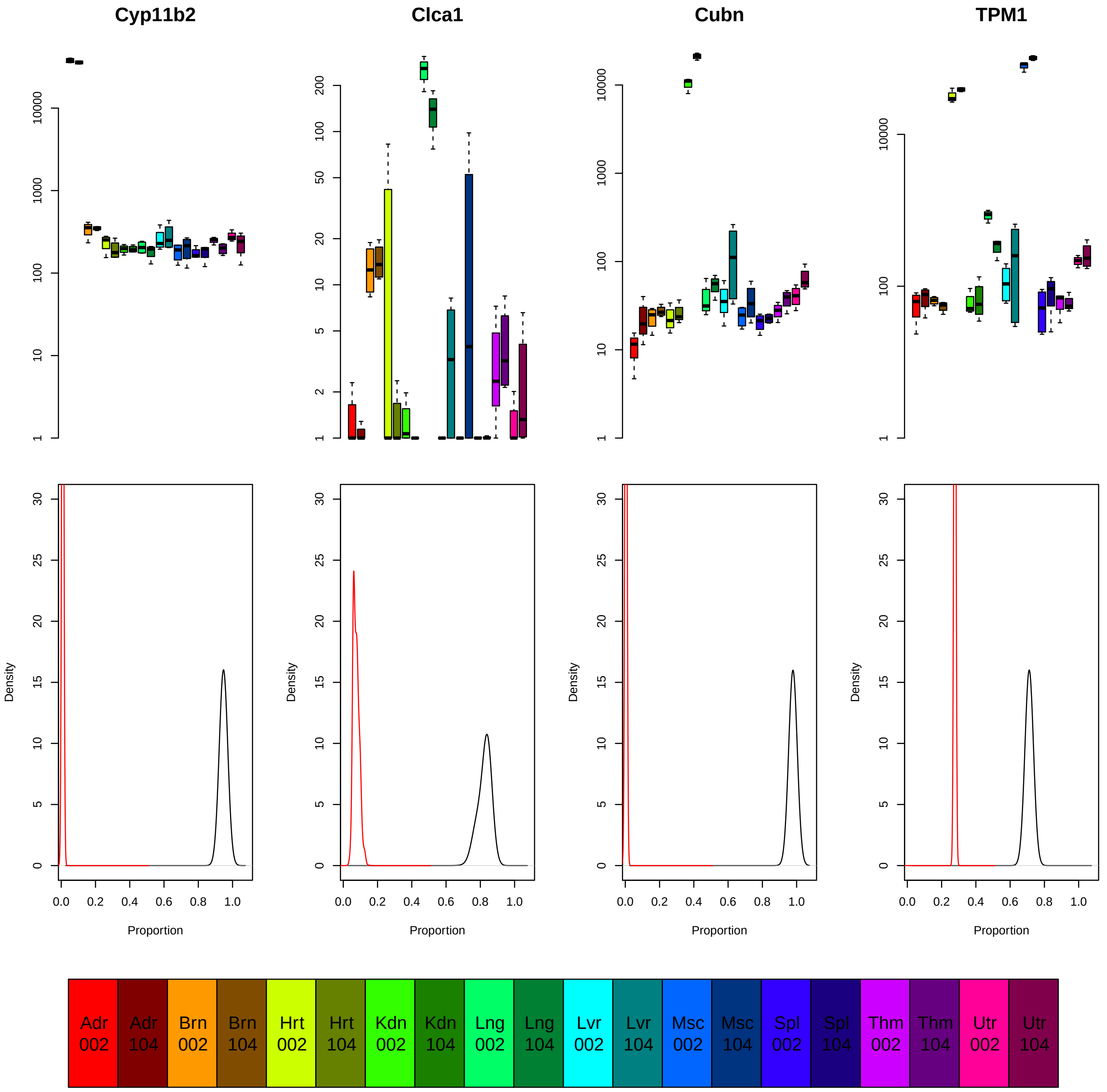

Figure S11: Gene expression plots and posterior distributions of the parameters for the proportion of expression in the tissue of highest (black) and second highest (red) expression. These four genes are the top ranked genes for a change in expression between tissue but not over time.

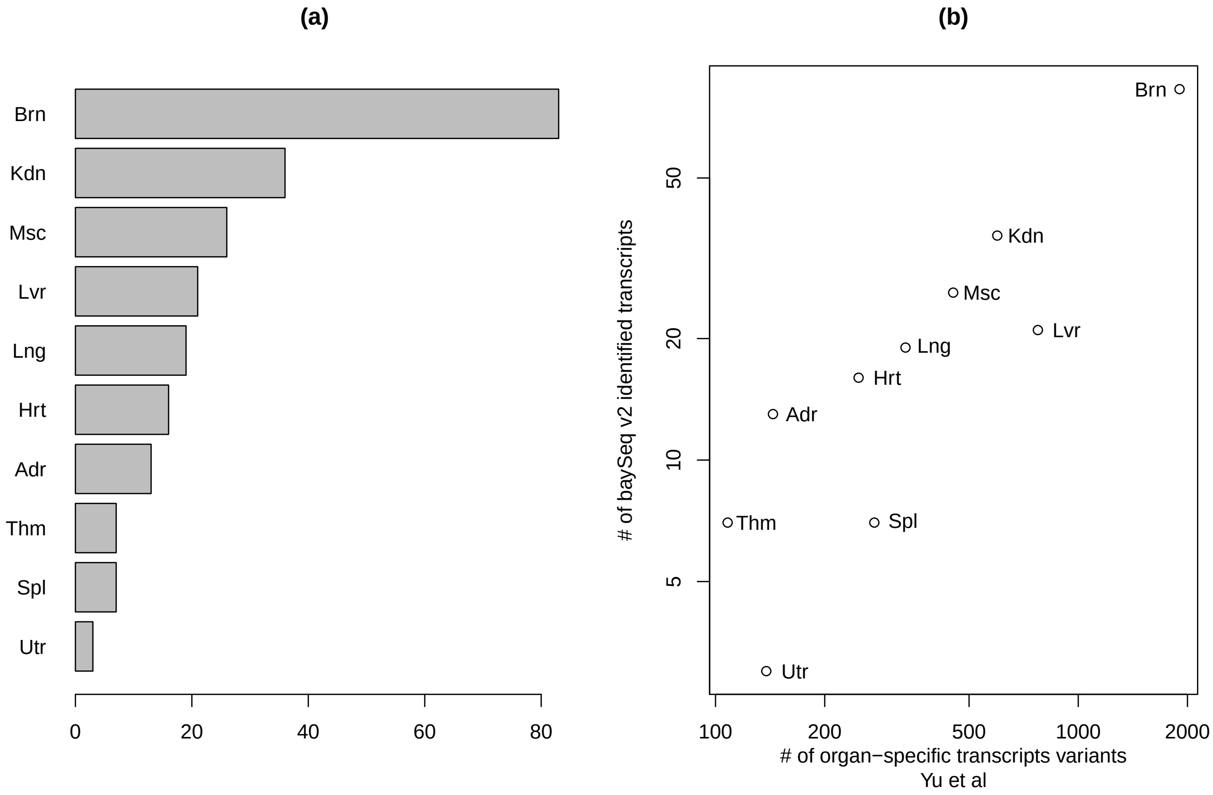

Figure S12: For the 231 genes (selected by controlling family-wise error rate at 10%) that are selected by `baySeq v2` as showing variability amongst tissues, and no differential behaviour between ages, the tissue in which the gene is maximally expressed can be identified. (a) shows the number of genes thus identified for each tissue, while (b) compares these numbers to those identified by [7] as being tissue-specific in an ANOVA analysis of tissue expression.

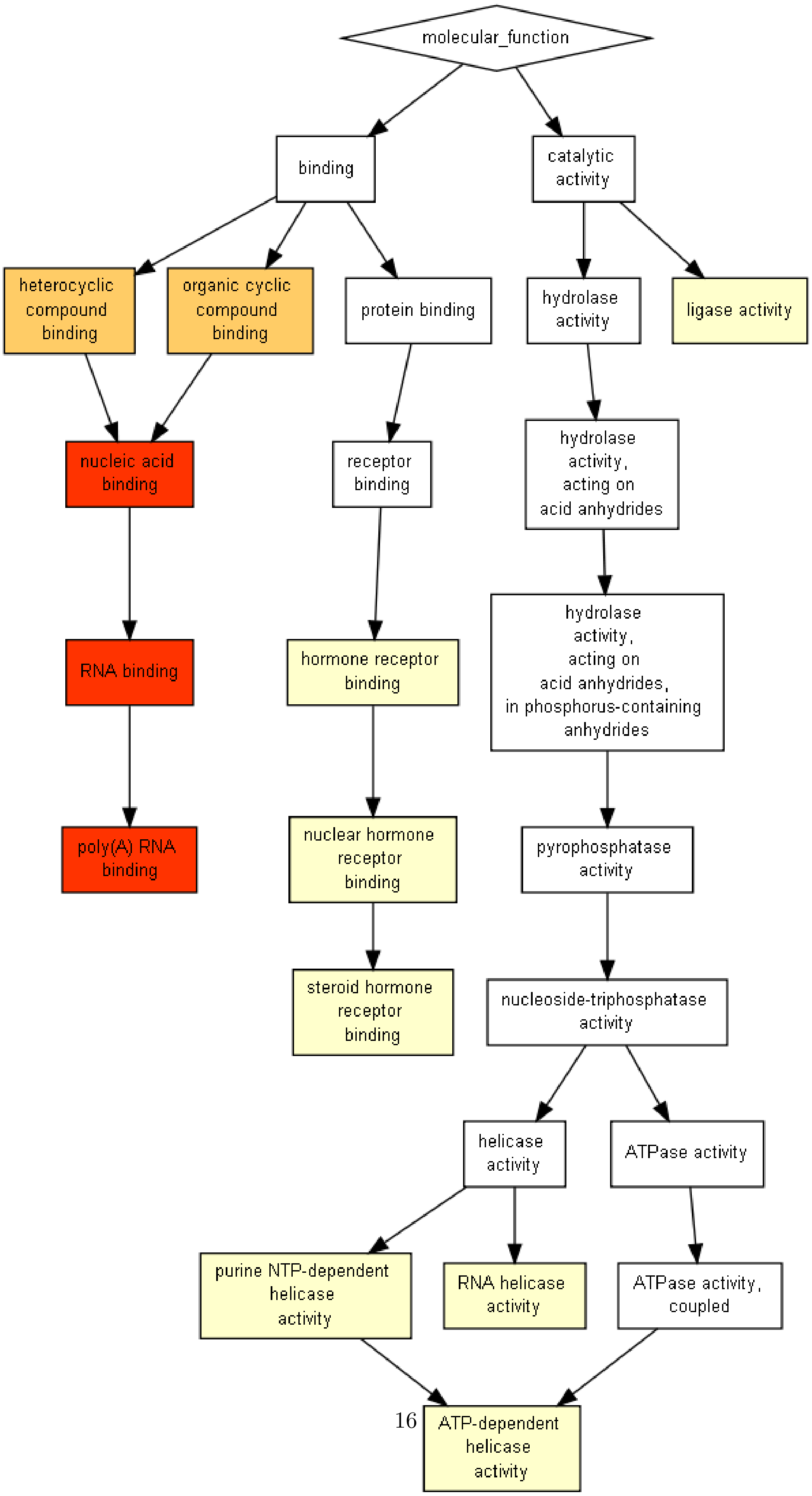
molecular_function
binding
catalytic activity
heterocyclic compound binding
organic cyclic compound binding
protein binding
hydrolase activity
ligase activity
nucleic acid binding
receptor binding
hydrolase activity, acting on acid anhydrides
RNA binding
hormone receptor binding
hydrolase activity, acting on acid anhydrides, in phosphorus-containing anhydrides
poly(A) RNA binding
nuclear hormone receptor binding
pyrophosphatase activity
steroid hormone receptor binding
nucleoside-triphosphatase activity
helicase activity
ATPase activity
purine NTP-dependent helicase activity
RNA helicase activity
ATPase activity, coupled
ATP-dependent helicase activity

Figure S13: Functional GO terms enriched in the gene set showing a decline in expression in thymus tissue relative to the expression across all other tissues.

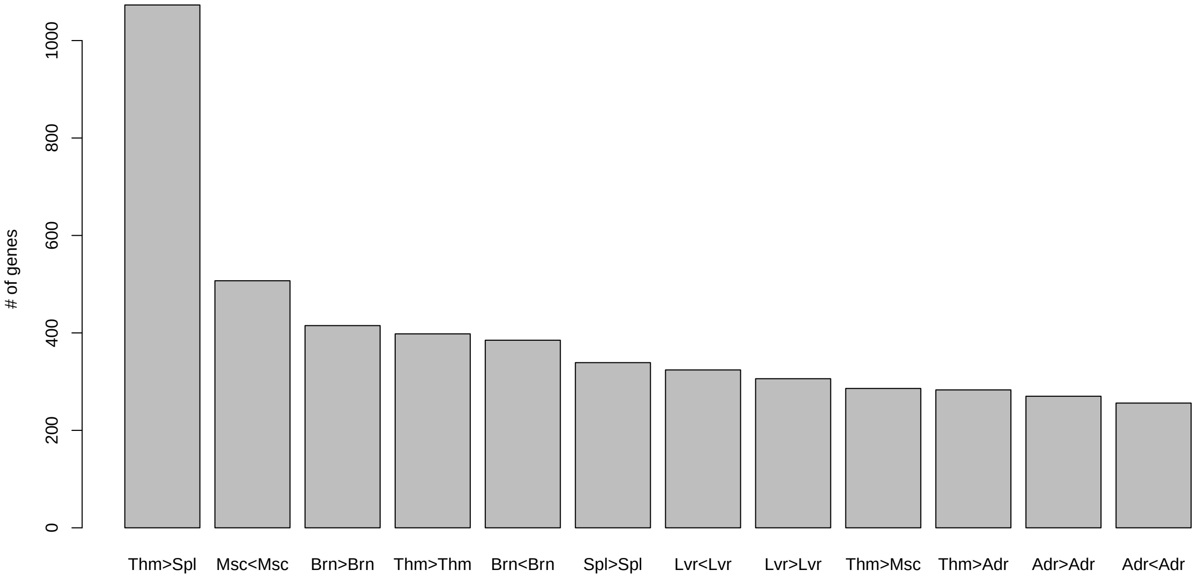

Figure S14: Numbers of genes for which there is a change in ratio of expression between tissues over time, split by tissues of maximum expression in the two time points. Bars are labelled as X> / <Y, such that the tissue with the highest proportion of expression in juvenile individuals is X, in aged individuals Y. The higher proportion of expression is indicated by the '> / <' symbol.

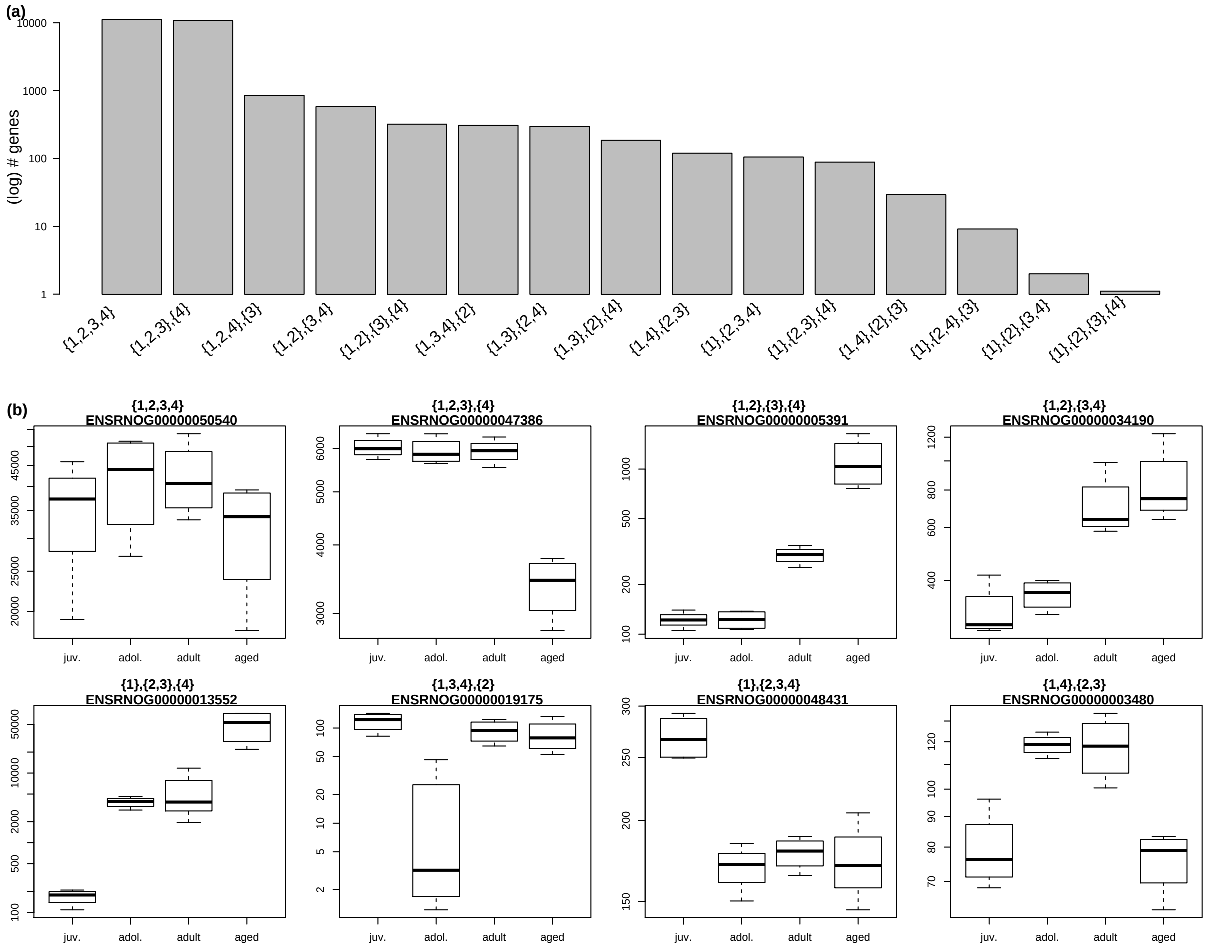

Figure S15: The expected number of genes belonging to each model (a). Normalised expression values of the top ranked gene for each of the eight models with highest expected number of genes, summarised by age group (b). Models are defined such that, e.g. {1,2,3}{4} indicates that the data from age groups 1,2 and 3 are equivalently distributed, and differ from age group 4. Age groups are identified in order, with 1 corresponding to juvenile individuals and 4 to aged individuals.

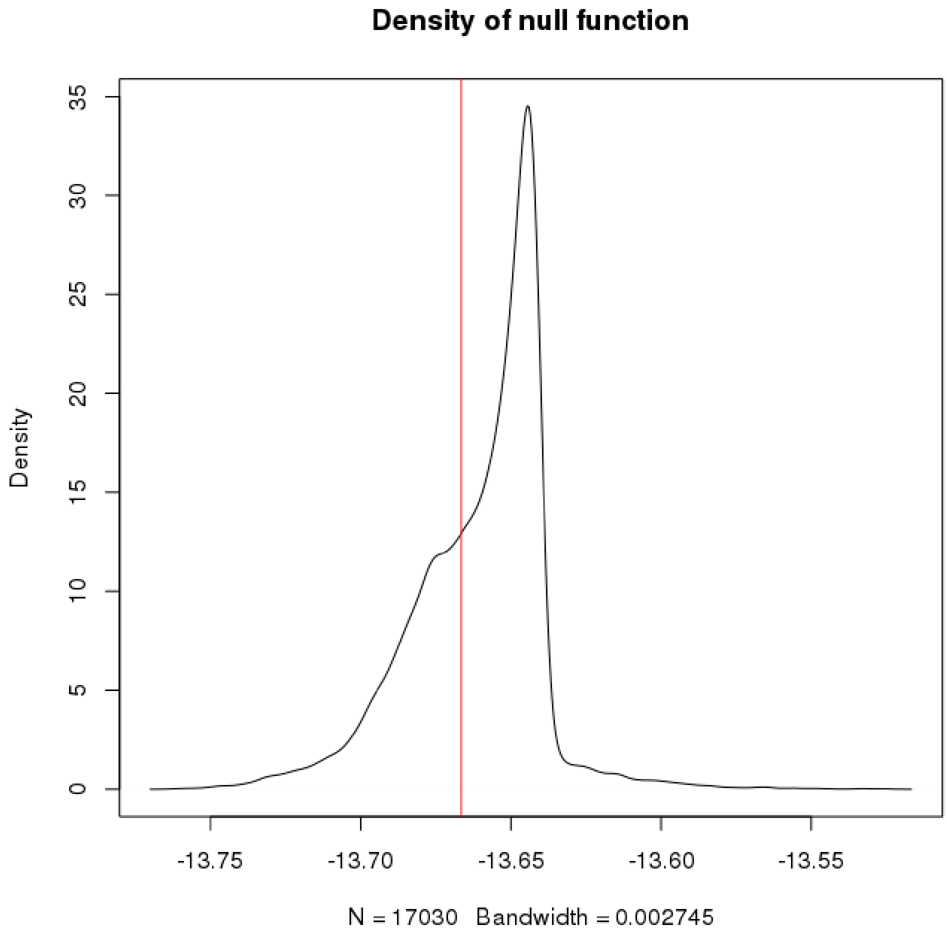

Figure S16: Distribution of the log of the parameter associated with the mean expression (scaled by library scaling factor and gene length) in RNA-seq data derived from rat thymus in juvenile female individuals [7]. The tail of data to the left of the modal peak may be considered to represent non-expressed genes. The red line indicates the threshold level which minimises the intra-class variance.

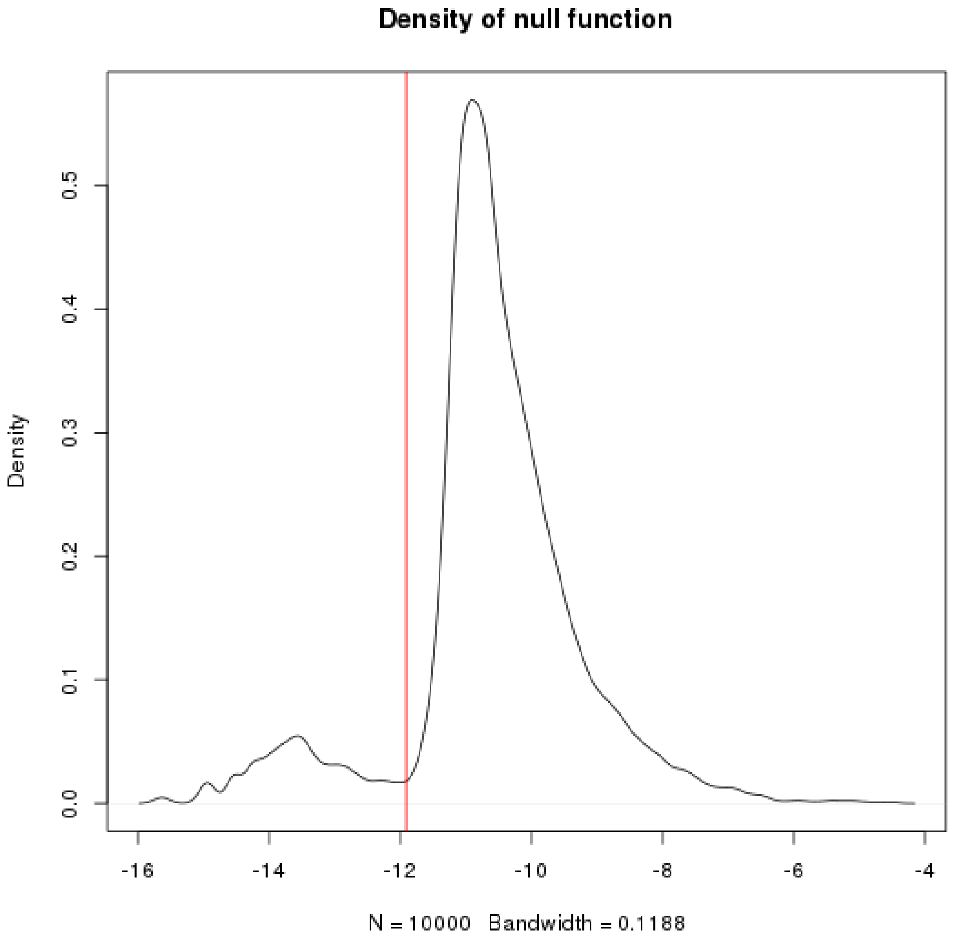

Figure S17: Distribution of the log of the parameter associated with the mean expression (scaled for library scaling factor) in a set of simulated RNA-Seq data. The tail of data to the left of the modal peak may be considered to represent non-expressed genes. The red line indicates the threshold level which minimises the intra-class variance.

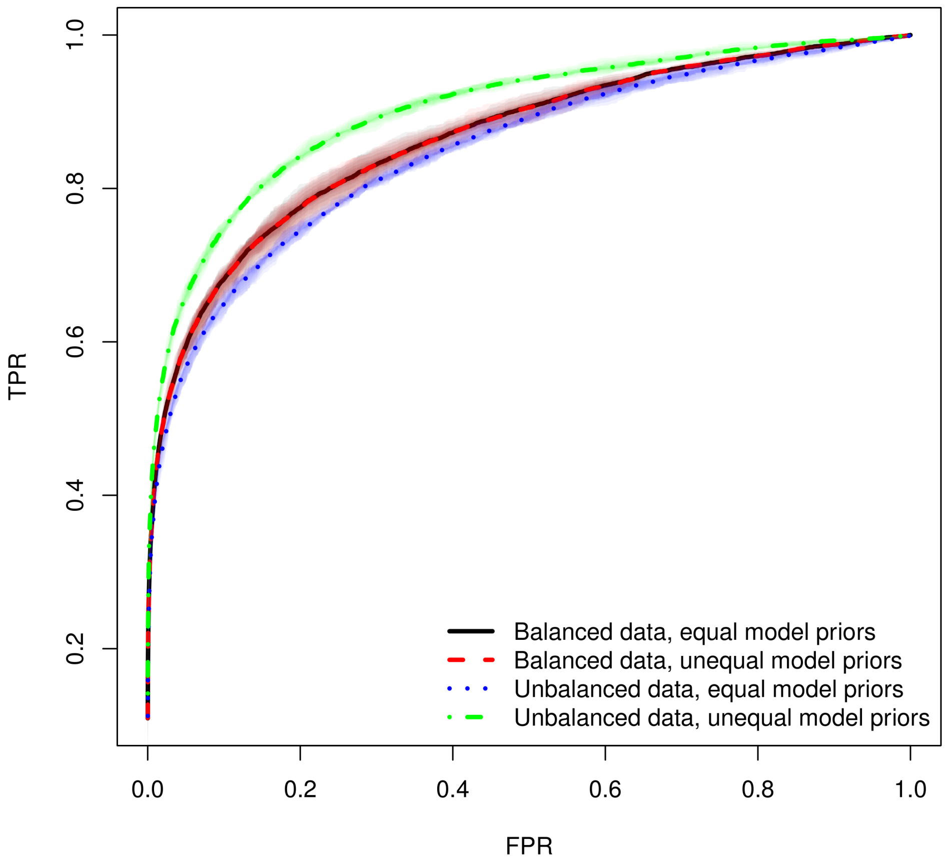

Figure S18: Average ROC curves showing performance of `baySeq v2` on balanced and unbalanced differentially expressed data. Allowing unequal model priors for different sets of the data increases performance for unbalanced data. Percentiles of true positive rates across samplings are shown as transparent areas around curves.

# S6 Supplementary Tables

| GO term | Description | P-value | FDR q-value |
|---|---|---|---|
| GO:0003723 | RNA binding | 3.17E-13 | 4.14E-10 |
| GO:0044822 | poly(A) RNA binding | 1.66E-10 | 1.08E-7 |
| GO:0003676 | nucleic acid binding | 2.69E-10 | 1.17E-7 |
| GO:1901363 | heterocyclic compound binding | 5.69E-6 | 1.86E-3 |
| GO:0097159 | organic cyclic compound binding | 7.25E-6 | 1.89E-3 |
| GO:0008026 | ATP-dependent helicase activity | 2.12E-4 | 4.61E-2 |
| GO:0070035 | purine NTP-dependent helicase activity | 2.12E-4 | 3.95E-2 |
| GO:0051427 | hormone receptor binding | 2.86E-4 | 4.66E-2 |
| GO:0035257 | nuclear hormone receptor binding | 3.05E-4 | 4.42E-2 |
| GO:0035258 | steroid hormone receptor binding | 3.91E-4 | 5.09E-2 |
| GO:0016874 | ligase activity | 7.28E-4 | 8.63E-2 |
| GO:0003724 | RNA helicase activity | 8.34E-4 | 9.07E-2 |

Table S1: Functional GO terms enriched in the gene set showing a decline in expression in thymus tissue relative to the expression across all other tissues.

# S7 Code Examples

## S7.1 Affymetrix Latin Square/Normally Distributed Data Code Example

This script fragment is intended to be run from the command line; it identifies differential expression in one set of spike-in replicates with another (both defined on the command-line) in the Affymetrix Latin Square HGU133 data.

The required libraries, including the HGU133 cdf with spike-in data are loaded.

```
require(affy)
require(limma)
require(baySeq)
require(hgu133atagcdf)
```

Processing of command line arguments; the degree of parallelisation, and the two experiment numbers.

```
args <- as.numeric(commandArgs(TRUE))
cl <- makeCluster(args[1])
eA <- args[2]
eB <- args[3]
```

Pre-processing of the Affymetrix data. The data are processed through the RMA algorithm. True positives have been identified previously.

```
ab <- ReadAffy(filenames = c(
     dir(".", pattern = paste("Expt", eA, "_.*CEL", sep = ""),
      full.names = TRUE),
     dir(".", pattern = paste("Expt", eB, "_.*CEL", sep = ""),
      full.names = TRUE)),
    cdfname = "hgu133atagcdf")
nvals <- rma(ab)

## Background correcting
## Normalizing
## Calculating Expression

ned <- exprs(nvals)
```

We discard spike-in data that are not in the set of true positives (previously identified), as these are highly variable and produce misleading results.

```
tp <- read.delim("latinsquare_TP.txt",
     sep = "\t", header = FALSE, as.is = TRUE)[,1]
spikes <- rownames(ned)[grep("AFFX", rownames(ned))]
spikes <- spikes[!(spikes %in% tp)]
ned <- ned[!rownames(ned) %in% spikes,]
```

Linear model fitting on the expression data. Since this is a pairwise comparison, the design matrix is a single vector.

```
fit <- lmFit(ned, design = c(1,1,1,-1,-1,-1))
eb <- eBayes(fit)
```

The density function class is specified as follows. This function is defined within baySeq and so need not be re-defined here, but is added to demonstrate the construction of such a class. Many of the parameters defined here are optional and offer only marginal performance improvements under specific conditions; the key element is the specification of the density in terms of observed data, sets of recorded observables, and variable parameters.

```
normDensity <-
 new("densityFunction",
  description = "A density function based on the normal distribution.",
          # The core density function
  density = function(dat, observables, parameters) {
          # return NA if any parameters are invalid
  if(any(sapply(parameters, function(par) any(par < 0))))
  return(NA)
          # calculate relative (log) likelihood of the
          # data under an assumed normal distribution
          # based on the parameters supplied
  dnorm(dat, parameters[[1]] * observables$libsizes,
  parameters[[2]], log = TRUE)},
          # Initiating values supplied to the numerical
          # estimation of maximum likelihood parameter
          # values for a given set of data. Avoiding the
          # boundaries of the domain of the density
          # function is usually advantageous for numerical
          # estimation, and so zeros are prevented from
          # occuring in these initial estimates.
  initiatingValues = function(dat, observables)
  c(mean = max(mean(dat / observables$libsizes),
   min(0.1 / observables$libsizes)),
  sd = max(sd(dat), 1e-4)),

          # The first parameter (mean expression) is
          # assumed to vary across experimental conditions
          # when estimating priors. The second parameter
          # (standard deviation) is assumed to be fixed
          # across experimental conditions.
  equalOverReplicates = c(FALSE, TRUE),
          # Bounds on the domain of the function for the
          # single non-fixed parameter.
  lower = function(dat) 0, upper = function(dat) 1 + max(dat) * 2,
          # Stratification function and number of strata
          # (if subsampling)
  stratifyFunction = rowMeans, stratifyBreaks = 10,
          # Function on the data which will likely exhibit
          # bimodality in the presence of 'null' data.
     nullFunction = function(pars) pars[,1])
```

A "countData" object can then be constructed from the data to carry out a pairwise comparison between the two experiments as for the limma analysis. Prior estimation and posterior likelihood estimation is then straightforward.

```
normCD <- new("countData", data = ned,
    replicates = c("1", "1", "1", "2", "2", "2"),
    groups = list(NDE = c(1,1,1,1,1,1), DE = c(1,1,1,2,2,2)))
densityFunction(normCD) <- normDensity
libsizes(normCD) <- c(1,1,1,1,1,1)
```

```
normCD <- getPriors(normCD, cl = cl)
normCD <- getLikelihoods(normCD, cl = cl)
```

## S7.2 Zero-Inflated Negative Binomial Code Example

This script fragment is intended to be run from the command line; it identifies differential expression in simulated zero-inflated count data using `baySeq v2` and ShrinkBayes.

### S7.2.1 Data simulation

Sampled mean values from SAGE set used to simulate count data

```
load("zhanglambdas.RData")
```

Simulation function for zero-inflated data

```
simZI <- function(muscale, zprop, libs, b = 8, n = 10000, perDE = 0.1) {
 # Sample library sizes and dispersions
 libsizes <- round(runif(libs, 30000, 90000)) * muscale
 dispersions <- rgamma(n, shape = 0.85, scale = 0.5)

 # Create empty matrix
 y <- matrix(0, nrow = n, ncol = libs)

 # simulate DE genes
 if(n * perDE > 1)
  for(jj in 1:(n * perDE))
   {
    samlam <- sample(lambdas, 1)
    rcoin <- rep(sample(c(-1, 1)), each = (libs / 2))
    while(sum(y[jj,]) < 1)
     y[jj,] <- rnbinom(libs, size = 1/dispersions[jj],
              mu = (sqrt(b) ^ rcoin) * samlam * libsizes)
   }

 #simulate NDE genes
 for(jj in (n*perDE+1):n)
  {
   samlam <- sample(lambdas, 1)
   while(sum(y[jj,]) < 1)
    y[jj,] <- rnbinom(libs, size = 1/dispersions[jj], mu = samlam * libsizes)
  }

 ziDat <- y

 # zero inflate data
 p <- runif(nrow(ziDat), 0, zprop)
 for(ii in 1:nrow(ziDat))
  ziDat[ii, sample(c(TRUE, FALSE), size = ncol(ziDat), replace = TRUE,
           prob = c(p[ii], 1-p[ii]))] <- 0
 ziDat
}
```

Required libraries are loaded:

```
require(baySeq)
require(ShrinkBayes)
```

Processing command-line arguments; the degree of parallelisation, the scaling parameter on library sizes, the proportion of zero inflation, the number of libraries, and the proportion of differential expression.

```
args <- as.numeric(commandArgs(TRUE))
print(args)

## [1]  8.0  5.0  0.5 10.0  0.1

muscale <- args[2]; zprop = args[3]; libs = args[4]; perDE = args[5]
cl <- makeCluster((args[1]))
```

Simulation of data:

```
ziDat <- simZI(muscale, zprop, libs = libs, perDE = perDE)
```

### S7.2.2 baySeq fitting

Model specification for baySeq

```
groups <- list(NDE = rep(1, libs), DE = rep(1:2, each = libs / 2))
```

The density function class is specified as follows. This function is defined within baySeq and so need not be re-defined here, but is added to demonstrate the construction of such a class. Many of the parameters defined here are optional and offer only marginal performance improvements under specific conditions; the key element is the specification of the density in terms of observed data, sets of recorded observables, and variable parameters. The constraint on the domain of the likelihood function such that $\zeta \geq \max_r \{1 - 2^{-1|D_{cF_r}|}\}$, i.e., $\zeta$ is greater or equal to that proportion of zero-inflation which gives a 50% chance of seeing no zeros within the smallest replicate group is introduced in the core density function.

```
# The core density function.
.dZINB <- function(dat, observables, parameters) {
 if(any(sapply(parameters, function(par) any(par < 0)))) return(NA)
 zinf <- parameters[[3]]
 # Zero-inflation cannot exceed 1
 if(any(zinf >= 1)) return(NA)

 # Zero-inflation should be greater than the proportion of
 # zero-inflation which gives a 50\% chance of seeing no zeros.
 if(any(zinf < 1-0.5^(1/observables$dim[2]))) return(NA)

 # probability of data without zero inflation
 nzero <- dnbinom(dat, mu = parameters[[1]] * observables$libsizes *
                  observables$seglens, size = 1 / parameters[[2]],
                  log = TRUE) + log(1 - zinf)
 # probability of zeros under zero inflation
 zero <- c(-Inf, log(zinf))[as.integer(dat == 0) + 1]

 # combined probabilities
 pmz <- pmax(nzero, zero)
 log(exp(zero - pmz) + exp(nzero - pmz)) + pmz
}


ZINBDensity <- new("densityFunction",
          description = "A density function based on the zero-inflated
                         negative-binomial distribution.",
          density = .dZINB,
                    # Initiating values supplied to the numerical
                    # estimation of maximum likelihood parameter
                    # values for a given set of data. Avoiding the
                    # boundaries of the domain of the density
                    # function is usually advantageous for numerical
                    # estimation, and so zeros are prevented from
                    # occuring in these initial estimates.
          initiatingValues = function(dat, observables)
          c(mean(pmax(dat,0.1) / observables$libsizes / observables$seglens), 0.1,
           min(0.9, max(1 - 0.5^(1/observables$dim[2]), sum(dat == 0)/length(dat)))),
                    # The first parameter (mean expression) is
                    # assumed to vary across experimental conditions
                    # when estimating priors. The second and third
                    # parameters (dispersion and zero-inflation) are
                    # assumed to be fixed across experimental conditions.
          equalOverReplicates = c(FALSE, TRUE, TRUE),
                    # Bounds on the domain of the function for the
                    # single non-fixed parameter.
          lower = function(dat) 0, upper = function(dat) 1 + max(dat) * 2,
                    # Stratification function and number of strata
                    # (if subsampling)
          stratifyFunction = rowMeans, stratifyBreaks = 10)
```

A “countData” object can then be constructed from the data to carry out a pairwise comparison between the two experiments as for the limma analysis. Prior estimation and posterior likelihood estimation is then straightforward.

```
CDZI <- new("countData", data = ziDat, replicates = rep(1:2, each = libs / 2),
            groups = groups, densityFunction = ZINBDensity)
libsizes(CDZI) <- getLibsizes(CDZI)
```

```
CDZI <- getPriors(CDZI, cl = cl)
CDZI <- getLikelihoods(CDZI, cl = cl)
```

### S7.2.3 ShrinkBayes fitting

Parameters for `ShrinkBayes`

```
group <- as.factor(rep(1:2, each = 5))
group <- BaselineDef("group", baselinegroup = "1")
```

Fit for differential expression

```
form = y ~ 1 + group
shrinkDE <- ShrinkSeq(form=form, dat=ziDat,shrinkfixed="group",mixtdisp=TRUE,ncpus=1)
fitgDE <- FitAllShrink(form,dat=ziDat,fams="zinb",shrinksimul = shrinkDE,ncpus=1)
```

Fit for non-differential expression

```
form <- y ~ 1
shrinkNDE <- ShrinkSeq(form=form, dat=ziDat,mixtdisp=TRUE,ncpus=1)
fitgNDE <- FitAllShrink(form,dat=ziDat,fams="zinb",shrinksimul = shrinkNDE,ncpus=1)
```

Non-parametric prior and posterior estimation; calculation of Bayesian false discovery rates.

```
cpprior <- NonParaUpdatePrior(fitall=fitgDE, fitall0 = fitgNDE, modus="fixed",
               shrinkpara="group", shrinklc=NULL,ncpus=1, maxiter=3,
               includeP0 = FALSE, symmetric=TRUE, logconcave=FALSE, allow2modes=FALSE)
cppostshr <- NonParaUpdatePosterior(fitgDE,cpprior, fitall0 = fitgNDE, ncpus=1)

lfdr <- SummaryWrap(cppostshr, thr = 0.1)
lBFDR <- BFDR(lfdr)
```

Mixture model prior and posterior estimation; calculation of Bayesian false discovery rates.

```
cmprior <- MixtureUpdatePrior(fitall=fitgDE, fitall0 = fitgNDE,
               shrinkpara = "group", ncpus = 1)
cmpost <- MixtureUpdatePosterior(fitall=fitgDE, cmprior, fitall0 = fitgNDE, ncpus = 1)
lmfdr <- SummaryWrap(cmpost, thr = 0.1)
lmBFDR <- BFDR(lmfdr)
```

### S7.3 Dirchlet-Multinomial Code Example

This script fragment is intended to be run from the command line; it identifies differential expression in matched sample data using a (restricted) Dirichlet-multinomial distribution implemented in `baySeq v2`

Load (eXpress) processed count data in the form of a list of matrices (object `cdat`), each matrix containing counts for one tissue.

```
load("female_rat_data.RData")
```

Create ’countData’ object. When the data slot is specified as a list of matrices, this is turned into a three-dimensional array.

```
require(baySeq)
cD <- new("countData", data = cdat[1:10],
          replicates = c(1,1,1,1,2,2,2,2),
          groups = list(NDE = c(1,1,1,1,1,1,1,1),
            DE = c(1,1,1,1,2,2,2,2)))
```

The core density function for calculating likelihoods of data given the model from a restricted Dirichlet-multinomial distribution function, in which the two highest proportion values in the multinomial fit are evaluated separately, and the remaining proportions are assumed to be equal. This function is defined within `baySeq` and so need not be re-defined here, but is added to demonstrate the construction of such a class.

```
.multiDirichletFunction2 <- function(dat, observables, parameters) {
  # No parameter may lie outside the range [0,1]
  if(any(sapply(parameters, function(par) any(par < 0) | any(par > 1))))
    return(NA)

  # Force ordering on proportion parameters
  if(!all(parameters[[2]] >= parameters[[3]])) return(NA)

  # Calculation of Dirichlet-multinomial densities for each row
  # of matrix x given matrix of proportions and dispersion parameter
  ddirmult <- function(x, prop, disp) {
    if(nrow(x) != nrow(prop))
      stop("Dimension of x does not match dimension of proportions")
    ps <- rep(NA, (nrow(x)))
    alphas = (1/disp - 1) * prop
    ps <- lfactorial(rowSums(x)) - rowSums(lfactorial(x)) +
      lgamma(rowSums(alphas)) - lgamma(rowSums(alphas) + rowSums(x)) +
        rowSums(lgamma(x + alphas) - lgamma(alphas))
    return(ps)
  }

  x <- dat[,,,drop = TRUE]
  # find indices of two maximum proportions
  maxp <-
    order(colSums(x / observables$libsizes), decreasing = TRUE)[1:2]
  #extract dispersion parameter
  disp <- parameters[[1]]
  disp <- array(disp, dim = dim(x))

  # construct proportion matrix from indices of maximum proportions
  props <- matrix(NA, nrow = nrow(x), ncol = observables$dim[3])
  props[,maxp] <- do.call("cbind", parameters[-1])
  props[is.na(props)] <-
    (1 - rowSums(props, na.rm = TRUE)) / (observables$dim[3] - 2)

  # enforce domain of function to avoid illegal values
  if(any(props >= 1) | any(props[,maxp[1]] < 1/observables$dim[3])
     | any(props <= 0) | any(disp <= 2e-15)) return(NA)

  # modify proportions by observed library sizes
  propmod <- props * observables$libsizes
  propmod <- propmod / rowSums(propmod)

  # compute likelihoods of data
  ddirmult(x = x, prop = propmod, disp = disp)
}
```

```
md2Density <- new("densityFunction",
  description = "A density function based on an
R[2]-multinomial-Dirichlet distribution, in which the two highest
proportion values in the multinomial fit are evaluated separately,
and the remaining proportions are assumed to be equal. This
density function is designed to assess genomic events that
generate multiple counts (e.g. RNA-Seq from multiple tissue types
taken from the same organism).",
# use the core density function specified above
   density = .multiDirichletFunction2,
# initiating values for maximum likelihood estimation estimated from data
# fudgeFactor prevents zeros in inital values
  initiatingValues = function(dat, observables) {
    dat <- dat / array(observables$libsizes,
                       dim = c(1, dim(observables$libsizes)))
    fudgeFactor <-
      0.1/observables$dim[3] /observables$dim[2] /max(observables$libsizes)
    props <- rep(1/observables$dim[3], observables$dim[3])
    if(sum(dat) > 0) {
      sumdat <- apply(dat, 3, sum)
      scaledat <- apply(dat, 2, sum)
      props[sumdat == 0] <- min(fudgeFactor / scaledat)
      props[sumdat > 0] <- apply(dat[,,sumdat > 0,drop = FALSE], 3,
        function(x) mean(x / scaledat, na.rm = TRUE)) -
                     min(fudgeFactor / scaledat) *
                       sum(sumdat == 0) / sum(sumdat != 0)
    }
    c(0.01, sort(props, decreasing = TRUE)[1:2])
  },
   equalOverReplicates = function(datdim) c(TRUE, FALSE, FALSE),
   lower = function(data) 0,
   upper = function(data) 1,
   stratifyFunction = function(data)
                  rowMeans(data[,,1] / rowSums(data), na.rm = TRUE),
   stratifyBreaks = 10,
# null function specified on the distribution of the largest proportion
   nullFunction = function(pars) pars[,2],
# modify estimated priors for the 'null' data to reflect consistent expression
   modifyNullPriors = function(x, datdim) {x[[1]][,2:3] <- 1/datdim[2]; x})
```

Assign the restricted Dirichlet-multinomial density class to the ’countData’ object.

```
densityFunction(cD) <- md2Density
```

Prior estimation and posterior likelihood estimation is carried out as before. ’Null’ data allows the differentiation of genes showing equal expression across tissue types and ages from those with equal expression across tissue types but variable across ages.

```
cD <- getPriors(cD, cl = cl)
cD <- getLikelihoods(cD, nullData = TRUE,
                     bsNullOnly = FALSE, bootStraps = 5, cl = cl)
```

## S8 Complex Modelling Code Example

This script fragment is intended to be run from the command line; it identifies multiple patterns of differential expression in RNA-seq data from four different age groups of female rat thymus tissue.

Processing of command line arguments; the degree of parallelisation used.

```
args <- as.numeric(commandArgs(TRUE))
cl <- makeCluster(args[1])
```

Load (eXpress) processed count data as a matrix (object `cdat`) containing counts for each gene/sample, and annotation for the genes (object `annot`).

```
load("female_thymus_all_ages.RData")
```

Construct `countData` object, specifying replicate structure and standard negative-binomial distribution

```
require(baySeq)
cD <- new("countData", data = cdat,
          replicates = c(1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4), annotation = annot)
libsizes(cD) <- getLibsizes(cD)
densityFunction(cD) <- nbinomDensity
```

Specify groups for a reduced model fit.

```
groups(cD) <- list(NDE = c(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1),
                  LDE = c(1,1,1,1,1,1,1,1,1,1,1,1,2,2,2,2),
                  MDE = c(1,1,1,1,1,1,1,1,2,2,2,2,2,2,2,2),
                  EDE = c(1,1,1,1,2,2,2,2,2,2,2,2,2,2,2,2),
                  cat = c(1,1,1,1,2,2,2,2,3,3,3,3,4,4,4,4))
```

Reduced model fit; prior and posterior likelihood estimation

```
cD <- getPriors(cD, cl = cl)
cD <- getLikelihoods(cD, cl = cl)
```

Reduced model fit; prior and posterior likelihood estimation with consensus priors

```
cDc <- getPriors(cD, consensus = TRUE, cl = cl)
cDc <- getLikelihoods(cDc, cl = cl)
```

All models generated from replicate structure of data.

```
cDa <- allModels(cD)
```

All model fit; prior and posterior likelihood estimation.

```
cDa <- getPriors(cDa, cl = cl)
cDa <- getLikelihoods(cDa, cl = cl)
```

All model fit; prior and posterior likelihood estimation with consensus priors.

```
cDac <- getPriors(cDa, cl = cl, consensus = TRUE)
cDac <- getLikelihoods(cDac, cl = cl)
```

# References

[1] T. J. Hardcastle and K. A. Kelly. baySeq: empirical Bayesian methods for identifying differential expression in sequence count data. *BMC Bioinformatics*, 11(1):422, Jan. 2010.

[2] T. J. Hardcastle and K. A. Kelly. Empirical Bayesian analysis of paired high-throughput sequencing data with a beta-binomial distribution. *BMC Bioinformatics*, 14(1):135, Jan. 2013.

[3] N. Otsu. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.

[4] F. Rapaport, R. Khanin, Y. Liang, M. Pirun, A. Krek, P. Zumbo, C. E. Mason, N. D. Socci, and D. Betel. Comprehensive evaluation of differential gene expression analysis methods for RNA-seq data. *Genome Biology*, 14(9):R95, Jan. 2013.

[5] M. D. Robinson and G. K. Smyth. Moderated statistical tests for assessing differences in tag abundance. *Bioinformatics*, 23(21):2881–7, Nov. 2007.

[6] C. Soneson and M. Delorenzi. A comparison of methods for differential expression analysis of RNA-seq data. *BMC Bioinformatics*, 14(1):91, Jan. 2013.

[7] Y. Yu, J. C. Fuscoe, C. Zhao, C. Guo, M. Jia, T. Qing, D. I. Bannon, L. Lancashire, W. Bao, T. Du, H. Luo, Z. Su, W. D. Jones, C. L. Moland, W. S. Branham, F. Qian, B. Ning, Y. Li, H. Hong, L. Guo, N. Mei, T. Shi, K. Y. Wang, R. D. Wolfinger, Y. Nikolsky, S. J. Walker, P. Duerksen-Hughes, C. E. Mason, W. Tong, J. Thierry-Mieg, D. Thierry-Mieg, L. Shi, and C. Wang. A rat RNA-Seq transcriptomic BodyMap across 11 organs and 4 developmental stages. *Nature Communications*, 5:3230, Jan. 2014.