baySeq: Empirical Bayesian analysis of patterns of differential expression in count data

Thomas J. Hardcastle

November 22, 2024

1 Introduction

This vignette is intended to give a rapid introduction to the commands used in implementing empirical Bayesian methods of evaluating differential expression in high-throughput sequencing data by means of the baySeq R package. For fuller details on the methods being used, consult Hardcastle & Kelly (2010) [1] and Hardcastle (2015) [2].

We assume that we have data from a set of sequencing or other high-throughput experiments, arranged in an array such that each column describes a sample and each row describes some genomic event for which data have been acquired. For example, the rows may correspond to the different sequences observed in a sequencing experiment. The data then consists of the number of times each sequence is observed in each sample. We wish to determine which, if any, rows of the data correspond to some patterns of differential expression across the samples.

baySeq uses empirical Bayesian methods to estimate the posterior likelihoods of each of a set of models that define patterns of differential expression for each row. This approach begins by considering a distribution for the row defined by a set of underlying parameters for which some prior distribution exists. By estimating this prior distribution from the data, we are able to assess, for a given model about the relatedness of our underlying parameters for multiple libraries, the posterior likelihood of the model.

In forming a set of models upon the data, we consider which patterns are biologically likely to occur in the data. For example, suppose we have count data from some organism in condition A and condition B. Suppose further that we have two biological replicates for each condition, and hence four libraries A_1, A_2, B_1, B_2, where A_1, A_2 and B_1, B_2 are the replicates. It is reasonable to suppose that at least some of the rows may be unaffected by our experimental conditions A and B, and the count data for each sample in these rows will be *equivalent*. These data need not in general be identical across each sample due to random effects and different library sizes, but they will share the same underlying parameters. However, some of the rows may be influenced by the different experimental conditions A and B. The count data for the samples A_1 and A_2 will then be equivalent, as will the count data for the samples B_1 and B_2. However, the count data between samples A_1, A_2, B_1, B_2 will not be equivalent. For such a row, the data from samples A_1 and A_2 will then share the same set of underlying parameters, the data from samples B_1 and B_2 will share the same set of underlying parameters, but, crucially, the two sets will not be identical. However, baySeq takes an alternative approach to analysis that allows more complicated patterns of differential expression than simple pairwise comparison, and thus is able to cope with more complex experimental designs (Section 5.5).

In this initial vignette, we consider RNA-seq type data assumed to follow a negative binomial distribution. Alternative scenarios are discussed in the vignette *Advanced analysis using baySeq; generic distribution definitions*.

2 Preparation

We begin by loading the baySeq package.

```
> library(baySeq)
```

Note that because the experiments that baySeq is designed to analyse are usually massive, we should use (if possible) parallel processing as implemented by the snow package. We use the parallel package (if it exists), and define a *cluster*. If parallel is not present, we can proceed anyway with a NULL cluster. Results may be slightly different depending on whether or not a cluster is used owing to the non-deterministic elements of the method.

```
> if(TRUE) { # set to FALSE if you don't want parallelisation
+   library(parallel)  # explicit use, assume installed because in "Imports"
+   cl <- makeCluster(4)
+ } else {
+   cl <- NULL
+ }
```

We load a simulated data set consisting of count data on one thousand counts.

```
> data(simData)
> simData[1:10,]
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    4    1    5    2    3    0    1    1    1     0
 [2,]    1    0    9    6    5    0    1    0    0     1
 [3,]    9    2    5    5   14    2    3    1    0     4
 [4,]    7    3    8    2    2    0    1    0    1     0
 [5,]    2    2    4    7    0    0    0    0    0     1
 [6,]    2    1    0    1    0    4    3    5    5     3
 [7,]    9    8    8    8    9    1    2    1    0     0
 [8,]    9    5    7    8    7    1    2    0    1     2
 [9,]    6    2    2    3    0    0    0    0    0     0
[10,]    1    0    2    0    1    3   17    2    2    10
```

The data are simulated such that the first hundred counts show differential expression between the first five libraries and the second five libraries. Our replicate structure, used to estimate the prior distributions on the data, can thus be defined as

```
>   replicates <- c("simA", "simA", "simA", "simA", "simA",
+                   "simB", "simB", "simB", "simB", "simB")
```

We can also establish two group structures for the data.

Each member (vector) contained within the 'groups' list corresponds to one model upon the data. In this setting, a model describes the patterns of data we expect to see at least some of the tags correspond to. In this simple example, we expect that some of the tags will be equivalently expressed between all ten libraries. This corresponds to the 'NDE' model, or vector `c(1,1,1,1,1,1,1,1,1,1)` - all libraries belong to the same group for these tags.

We also expect that some tags will show differential expression between the first five libraries and the second five libraries. For these tags, the two sets of libraries belong to different groups, and so we have the model 'DE', or vector `c(1,1,1,1,1,2,2,2,2,2)` - the first five libraries belong to group 1 and the second five libraries to group 2. We thus have the following group structure

```
> groups <- list(NDE = c(1,1,1,1,1,1,1,1,1,1),
+                DE = c(1,1,1,1,1,2,2,2,2,2))
```

In a more complex experimental design (Section **??**) we might have several additional models. The key to constructing vectors corresponding to a model is to see for which groups of libraries we expect equivalent expression of tags.

We note that the group for DE corresponds to the replicate structure. This will often be the case, but need not be in more complex experimental designs.

The ultimate aim of the baySeq package is to evaluate posterior likelihoods of each model for each row of the data.

We begin by combining the count data and user-defined groups into a `countData` object.

```
> CD <- new("countData", data = simData, replicates = replicates, groups = groups)
```

Library sizes can be inferred from the data if the user is not able to supply them.

```
> libsizes(CD) <- getLibsizes(CD)
> libsizes(CD)
 [1] 1424  944 1519  744 1277 1186 1588  818  870 1406
```

We can then plot the data in the form of an MA-plot, suitable modified to plot those data where the data are uniformly zero (and hence the log-ratio is infinite) (Figure 1). Truly differentially expressed data can be identified in the plot by coloring these data red, while non-differentially expressed data are colored black.

```
> plotMA.CD(CD, samplesA = "simA", samplesB = "simB",
+           col = c(rep("red", 100), rep("black", 900)))
```

We can also optionally add annotation details into the `@annotation` slot of the `countData` object.

```
> CD@annotation <- data.frame(name = paste("count", 1:1000, sep = "_"))
```

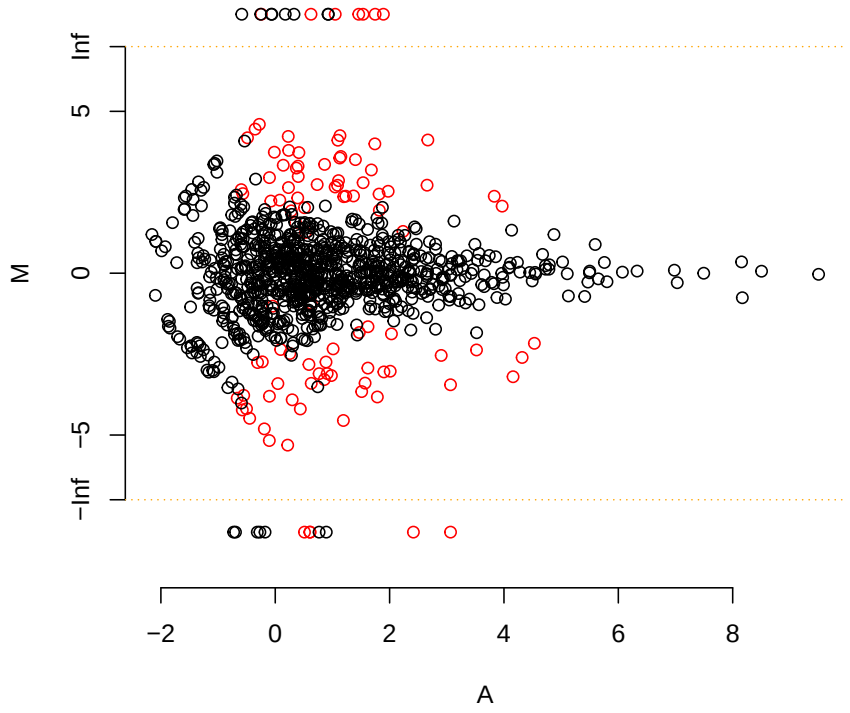

Figure 1: 'MA'-plot for count data. Where the log-ratio would be infinite (because the data in one of the sample groups consists entirely of zeros, we plot instead the log-values of the other group. Truly differentially expressed data are colored red, and non-differentially expressed data black.

3 Negative-Binomial Approach

We first estimate an empirical distribution on the parameters of the Negative Binomial distribution by bootstrapping from the data, taking individual counts and finding the quasi-likelihood parameters for a Negative Binomial distribution. By taking a sufficiently large sample, an empirical distribution on the parameters is estimated. A sample size of around 10000 iterations is suggested, depending on the data being used), but 1000 is used here to rapidly generate the plots and tables.

```
> CD <- getPriors.NB(CD, samplesize = 1000, estimation = "QL", cl = cl)
```

The calculated priors are stored in the `@priors` slot of the `countData` object produced as before. For the negative-binomial method, we are unable to form a conjugate prior distribution. Instead, we build an empirical prior distribution which we record in the list object `$priors` of the slot `@priors`. Each member of this list object corresponds to one of the models defined by the `group` slot of the `countData` object and contains the estimated parameters for each of the individual counts selected under the models. The vector `$sampled` contained in the slot `@priors` describes which rows were sampled to create these sets of parameters.

```
> CD@groups

$NDE
 [1] 1 1 1 1 1 1 1 1 1 1
Levels: 1

$DE
 [1] 1 1 1 1 1 2 2 2 2 2
Levels: 1 2

> sapply(names(CD@groups), function(group) lapply(CD@priors$priors[[group]], head, 5))

$NDE
$NDE[[1]]
                1
[1,] 0.0005535903 7.887362e-01
[2,] 0.0002547554 1.717391e-08
[3,] 0.0007454744 6.117239e-01
[4,] 0.0003396739 6.775168e-09
[5,] 0.0023315680 9.364410e-01


$DE
$DE[[1]]
                1
[1,] 0.0001263541 7.887362e-01
[2,] 0.0001263541 1.717391e-08
[3,] 0.0001263541 6.117239e-01
[4,] 0.0001263541 6.775168e-09
[5,] 0.0001263541 9.364410e-01

$DE[[2]]
                2
[1,] 0.0011571814 7.887362e-01
[2,] 0.0005112474 1.717391e-08
[3,] 0.0014791693 6.117239e-01
[4,] 0.0006816633 6.775168e-09
[5,] 0.0046124821 9.364410e-01
```

We then acquire posterior likelihoods, estimating the proportions of differentially expressed counts.

```
> CD <- getLikelihoods(CD, cl = cl, bootStraps = 3, verbose = FALSE)

...

> CD@estProps

     NDE       DE
0.881738 0.118262

> CD@posteriors[1:10,]

             NDE           DE
 [1,] -0.7182509 -0.668658238
```

```
 [2,] -1.1044953 -0.402536515
 [3,] -1.0081651 -0.453953779
 [4,] -2.8017825 -0.062622239
 [5,] -0.7890052 -0.605679478
 [6,] -1.0411667 -0.435474740
 [7,] -6.4331196 -0.001608722
 [8,] -4.4751954 -0.011453337
 [9,] -1.1732792 -0.370123310
[10,] -1.8065428 -0.179390932
```

```
> CD@posteriors[101:110,]
```

```
                NDE         DE
 [1,] -3.509570e-02  -3.367173
 [2,] -3.456222e-10 -21.785679
 [3,] -4.959439e-02  -3.028572
 [4,] -2.507815e-02  -3.698271
 [5,] -2.022689e-05 -10.808508
 [6,] -2.900704e-02  -3.554685
 [7,] -5.219352e-02  -2.978780
 [8,] -1.773987e-02  -4.040798
 [9,] -4.623394e-02  -3.097069
[10,] -1.502412e-02  -4.205601
```

Here the assumption of a Negative Binomial distribution with priors estimated by maximum likelihood gives an estimate of

```
      DE
0.118262
```

as the proportion of differential expressed counts in the simulated data, where in fact the proportion is known to be 0.1.

4 Results

We can ask for the top candidates for differential expression using the `topCounts` function.

```
> topCounts(CD, group = "DE")
```

```
       name simA.1 simA.2 simA.3 simA.4 simA.5 simB.1 simB.2 simB.3 simB.4 simB.5
80 count_80      1      1      0      1      1     13     21      8      6     20
78 count_78      1      1      0      1      1      8     13      7      9     10
66 count_66      0      0      0      0      0     15     10      4      4     10
21 count_21      2      0      1      1      0     15     15      6      5     11
7   count_7      9      8      8      8      9      1      2      1      0      0
26 count_26     13      4     11      5      7      1      1      1      0      0
64 count_64      6      6      8     11      9      1      1      0      0      1
72 count_72      0      0      1      0      0      7      6      4      3      8
83 count_83     14      6      9      2      9      1      0      0      1      1
27 count_27      5      3      6      4      7      0      0      0      1      0
       likes  DE       FDR.DE      FWER.DE
80 0.9995791 2>1 0.0004208781 0.0004208781
```

```
78 0.9992829 2>1 0.0005689902 0.0011376785
66 0.9990943 2>1 0.0006812318 0.0020423630
21 0.9985733 2>1 0.0008675879 0.0034661054
7  0.9983926 1>2 0.0010155560 0.0050679624
26 0.9980266 1>2 0.0011752048 0.0070314101
64 0.9962283 1>2 0.0015461322 0.0107765864
72 0.9957691 2>1 0.0018817294 0.0149619013
83 0.9954451 1>2 0.0021787486 0.0194486538
27 0.9941751 1>2 0.0025433681 0.0251603097
```

We can plot the posterior likelihoods against the log-ratios of the two sets of samples using the `plotPosteriors` function, coloring the truly differentially expressed data red and the non-differentially expressed data black (Figure 2).

```
> plotPosteriors(CD, group = "DE", col = c(rep("red", 100), rep("black", 900)))
```

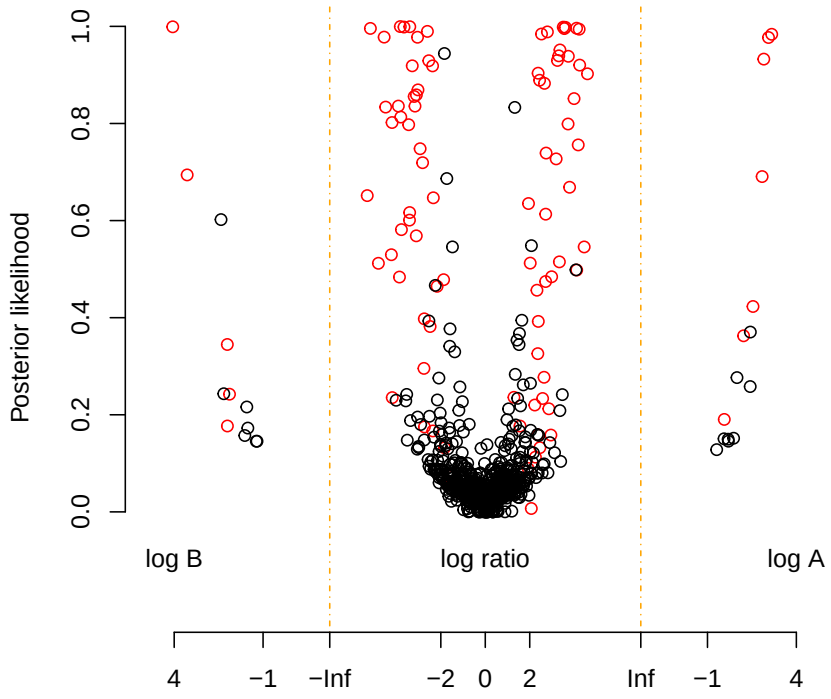

Figure 2: Posterior likelihoods of differential expression against log-ratio (where this would be non-infinite) or log values (where all data in the other sample group consists of zeros). Truly differentially expressed data are colored red, and non-differentially expressed data black.

5 Case Study: Analysis of sRNA-Seq Data

5.1 Introduction

We will look at data sequenced from small RNAs acquired from six samples of root stock from *Arabidopsis thaliana* in a grafting experiment [3]. Three different biological conditions exist within these data; one in which a Dicer 2,3,4 triple mutant shoot is grafted onto a Dicer 2,3,4 triple mutant root (**SL236** & **SL260**), one in which a wild-type shoot is grafted onto a wild-type root (**SL239** & **SL240**), and one in which a wild-type shoot is grafted onto a Dicer 2,3,4 triple mutant root (**SL237** & **SL238**). Dicer 2,3,4 is required for the production of 22nt and 24nt small RNAs, as well as some 21nt ones. Consequently, if we detect differentially expressed sRNA loci in the root stock of the grafts, we can make inferences about the mobility of small RNAs.

5.2 Reading in data

The data and annotation are stored in two text files. We can read them in using **R**'s standard functions.

```
> data(mobData)
> data(mobAnnotation)
```

5.3 Making a countData object

We can create a `countData` object containing all the information we need for a first attempt at a differential expression analysis.

5.3.1 Including lengths

If two genes are expressed at the same level, but one is twice the length of the other, then (on average) we will sequence twice as many reads from the longer gene. The same is true for sRNA loci, and so in these analyses it is often useful to include the lengths of each feature. The lengths can be derived from the annotation of each feature, but we need to explicitly declare them within the 'countData' object.

```
> seglens <- mobAnnotation$end - mobAnnotation$start + 1
> cD <- new("countData", data = mobData, seglens = seglens, annotation = mobAnnotation)
```

Determining the best library scaling factor to use is a non-trivial task. The simplest approach would be to use the total number of sequenced reads aligning to the genome. However, this approach meas that a few sequences that appear at very high levels can drastically skew the size of the scaling factor. Bullard *et al* suggest that good results can be obtained by taking the sum of the reads below the nth percentile of the data.

```
> libsizes(cD) <- getLibsizes(cD, estimationType = "quantile")
```

5.4 Pairwise Differential Expression

We start by looking at a pairwise differential expression analysis between two of the sample types. The analysis between samples 'SL236', 'SL260' and 'SL237', 'SL238' should be a first step in discovering sRNA loci associated with mobility.

We begin by selecting a subset of the available data:

```
> cDPair <- cD[,1:4]
```

We then need to define the replicate structure of the `countData` object. We do this by creating a vector that defines the replicate group that each sample belongs to.

```
> replicates(cDPair) <- as.factor(c("D3/D3", "D3/D3", "WT/D3", "WT/D3"))
```

We next need to define each of the models applicable to the data. In the first case, it is reasonable to suppose that at least some of the loci will be unaffected by the different experimental conditions prevailing in our replicate groups, and so we create one model of no differential expression.

We do this by defining a vector `NDE`.

```
> NDE <- c(1,1,1,1)
```

Each member of the `NDE` vector represents one sample in our experiment. By giving each item in the `NDE` vector the same number, we indicate that, under the hypothesis of no differential expression, all the samples belong to the same group.

We may also conjecture that some of the loci will be affected depending on whether the shoot is a Dicer mutant or a wild-type *Arabidopsis* sample.

```
> mobile <- c("non-mobile","non-mobile","mobile","mobile")
```

This vector indicates that the third and fourth samples, which consist of the wild-type shoot samples, are in a separate expression group to the first and second samples, corresponding to the Dicer 2,3,4 mutant shoot.

We can now add these models to the locus data by modfiying the `@groups` slot

```
> groups(cDPair) <- list(NDE = NDE, mobile = mobile)
```

Now that we have defined our models, we need to establish prior distributions for the data. We do this using the `getPriors.NB` function.

```
> cDPair <- getPriors.NB(cDPair, samplesize = 1e4, cl = cl)
```

The accuracy of the distribution is determined by the number of data points used to estimate the distribution; the 'samplesize'. Here we've used a small sample size to reduce the computational effort required, but higher values will give more accurate results (the default is 1e5).

Having found prior distributions for the data, we can identify posterior likelihoods for the data using the `getLikelihoods` function. Before we do this, however, it is worth considering the possibility that some loci will not be expressed at all in our data.

5.4.1 Null Data

We first examine the priors to see if any 'null' data, consisting of un-expressed sRNA loci, are present. If the distribution of priors for the non-differentially expressed group is bimodal, it is likely that some of the loci are expressed at substantially lower levels than others.

```
> plotNullPrior(cDPair)
```

There is some evidence for bimodality, with a small peak of lowly expressed data to the left of the distribution.

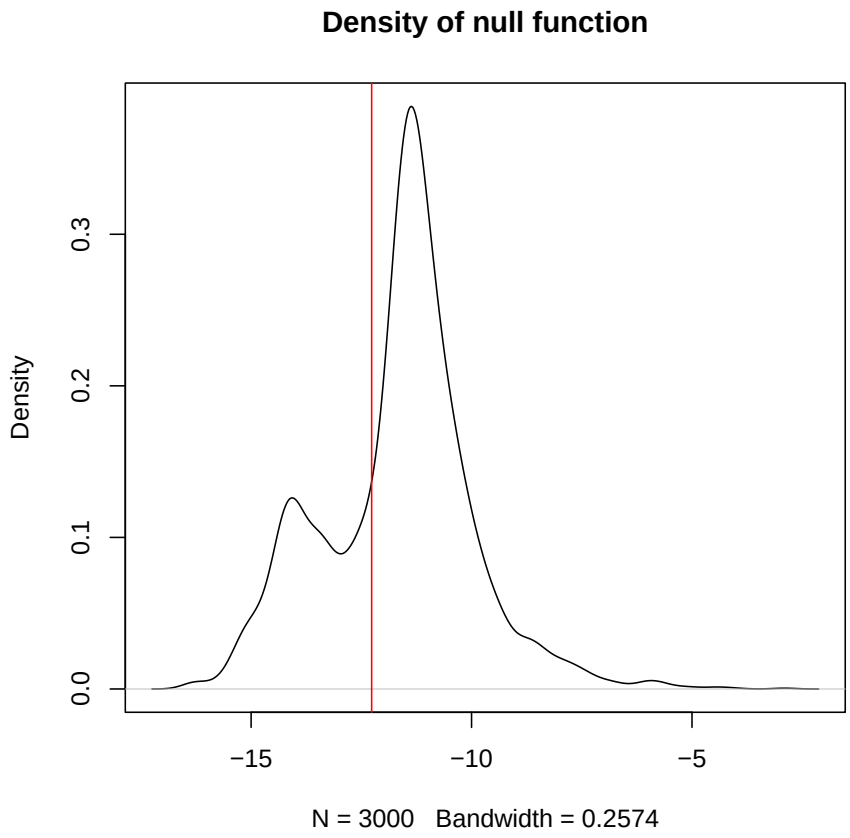

Figure 3: Distribution of μ_{ij}. Bimodality suggests the presence of 'null', or un-expressed, data.

We can use the `nullData = TRUE` option in the `getLikelihoods` function to allow for the possibility that some of the loci are miscalled in our locus map, and should properly be identified as nulls.

```
> cDPair <- getLikelihoods(cDPair, nullData = TRUE, cl = cl)
```

If we now look at the `cDPair` object, we can see that we have acquired posterior likelihoods for the data

```
> cDPair

An object of class "countData"
3000 rows and 4 columns

Slot "replicates"
D3/D3 D3/D3 WT/D3 WT/D3
```

```
Slot "groups":
$NDE
[1] 1 1 1 1
Levels: 1

$mobile
[1] non-mobile non-mobile mobile     mobile
Levels: mobile non-mobile


Slot "data":
     SL236 SL260 SL237 SL238
[1,]     0     0     0     0
[2,]    18    21     1     5
[3,]     1     2     2     3
[4,]    68    87   270   184
[5,]    68    87   270   183
2995 more rows...

Slot "annotation":
  chr start   end
1   1   789   869
2   1  8641  8700
3   1 10578 10599
4   1 17041 17098
5   1 17275 17318
2995 more rows...
Slot "posteriors":
             NDE    mobile
[1,] 0.002100924 0.0235143
[2,] 0.132668688 0.8672543
[3,] 0.839641267 0.1406516
[4,] 0.277694727 0.7223053
[5,] 0.365355942 0.6346441
2995 more rows...

Slot "estProps":
      NDE    mobile
0.4452675 0.2997288
```

The estimated posterior likelihoods for each model are stored in the natural logarithmic scale in the `@posteriors` slot of the `countDataPosterior` object. The nth column of the posterior likelihoods matrix corresponds to the nth model as listed in the `group` slot of `CDPair`. In general, what we would like to do with this information is form a ranked list in which the loci most likely to be differentially expressed are at the top of the list.

Try looking at the proportions of data belonging to each group. Note that these no longer sum to 1, as some data are now classified as 'null'.

```
> summarisePosteriors(cD)

numeric(0)
```

The value contained in the `@estProps` slot is a best-guess figure for the proportion of data belonging to each model defined by the `@groups` slot. In this case, it is is estimated that approximately 65% of the loci are not differentially expressed, while 35% are differentially expressed. These estimates should not be relied upon absolutely, but are a useful indicator of the global structure of the data.

We can ask for the rows most likely to be differentially expressed under our different models using the `topCounts` function. If we look at the second model, or grouping structure, we see the top candidates for differential expression. Because the library sizes of the different libraries differ, it can be unclear as to why some loci are identified as differentially expressed unless the data are normalised.

```
> topCounts(cDPair, group = 2, normaliseData = TRUE)

     chr    start      end SL236 SL260 SL237 SL238     likes            mobile
74     1   447231   447298     0     0   174   146 0.9999530 mobile>non-mobile
1111   1  8287590  8287674     0     0   107    83 0.9994780 mobile>non-mobile
2500   1 13463357 13463459    12    20   165   140 0.9992603 mobile>non-mobile
1334   1  9254068  9254167     0     0   101    81 0.9990406 mobile>non-mobile
625    1  5056092  5056161    65   184     1     0 0.9989713 non-mobile>mobile
2962   1 14188044 14188079     1     0    47    33 0.9987427 mobile>non-mobile
266    1  2157113  2157287    26    40   751   405 0.9985358 mobile>non-mobile
1696   1 11140107 11140158     0     0    60    38 0.9983326 mobile>non-mobile
1212   1  8766946  8767133   171   487    12     8 0.9982605 non-mobile>mobile
901    1  6880517  6880553     0     0    41    26 0.9981053 mobile>non-mobile
       FDR.mobile  FWER.mobile
74   4.695759e-05 4.695759e-05
1111 2.844576e-04 5.688906e-04
2500 4.361965e-04 1.308144e-03
1334 5.670011e-04 2.266304e-03
625  6.593351e-04 3.292644e-03
2962 7.590004e-04 4.545831e-03
266  8.597445e-04 6.003384e-03
1696 9.607056e-04 7.660807e-03
1212 1.047233e-03 9.386936e-03
901  1.131984e-03 1.126389e-02
```

Observe how the data change in the normalised results; the effect is particularly noticable in the SL236 and SL260 datasets, in which the normalised data is much less variable between these two samples.

We can also use `topCounts` to examine the data identified as 'null'.

```
> topCounts(cDPair, group = NULL, number = 500)
```

We can visualise the data in a number of ways. We can first examine the posterior likelihoods against log-ratio values.

```
> plotPosteriors(cDPair, group = 2, samplesA = 1:2, samplesB = 3:4)
```

Also informative is the MA-plot. We can color the data by the posterior likelihoods of differential expression.

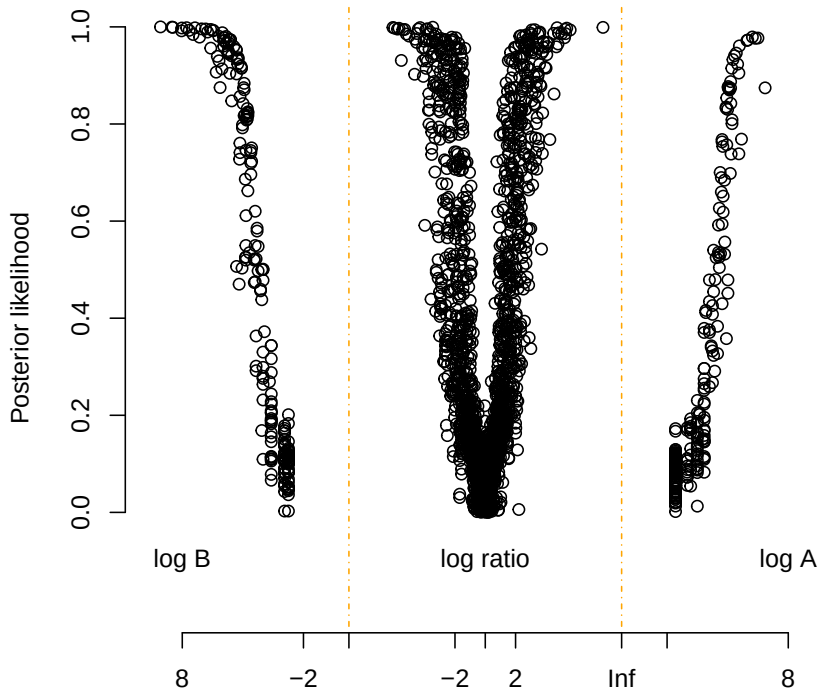

Figure 4: Posterior likelihoods of differential expression against log-ratios of the data. Where the data in one of the sample groups consists entirely of zeros, the log-ratio would be infinite. In this case, we plot instead the log-values of the non-zero group. Note the skew in the data; there are many more loci with a high-likelihood of differential expression over-expressed in the WT/D3 graft compared to the D3/D3 graft than vice versa.

```
> plotMA.CD(cDPair, samplesA = c(1,2), samplesB = c(3,4),
+           col = rgb(red = exp(cDPair@posteriors[,2]), green = 0, blue = 0))
```

5.5 Multiple Group Comparisons

We next examine all three experimental conditions simultaneously. We first need to define the replicate structure of the data.

```
> cD@replicates <- as.factor(c("D3/D3", "D3/D3", "WT/D3", "WT/D3", "WT/WT", "WT/WT"))
```

As before, we begin by supposing that at least some of the loci will be unaffected by the different experimental conditions prevailing in our replicate groups, and so we create one model of no differential expression.

We do this by defining a vector `NDE`.

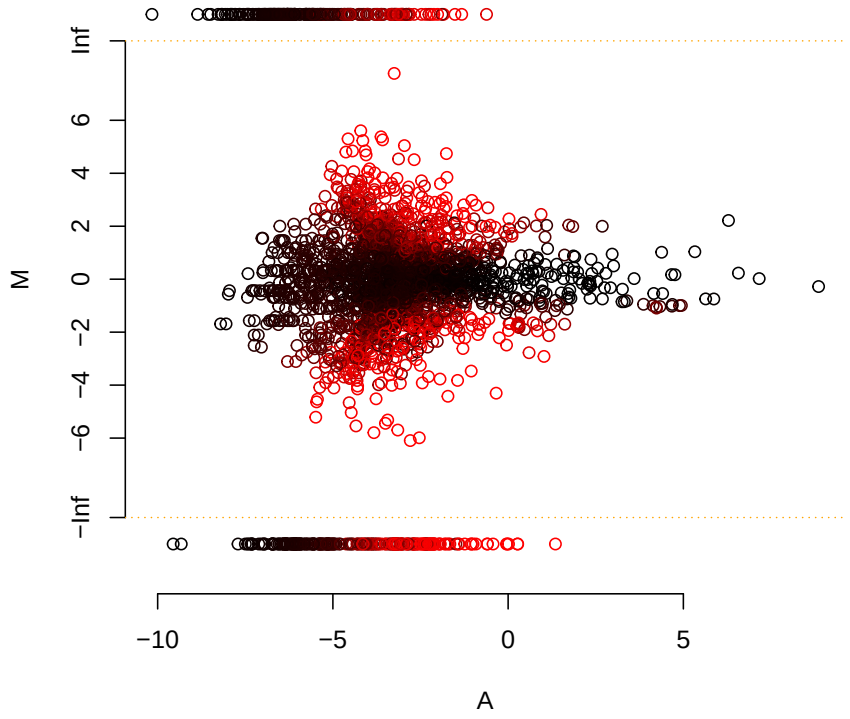

Figure 5: 'MA'-plot for count data. Where the data in one of the sample groups consists entirely of zeros, the log-ratio would be infinite. In this case, we plot instead the log-values of the non-zero group. Differentially expressed data are colored red, and non-differentially expressed data black.

```
> NDE <- factor(c(1,1,1,1,1,1))
```

Each member of the `NDE` vector represents one sample in our experiment. By giving each item in the `NDE` vector the same number, we indicate that, under the hypothesis of no differential expression, all the samples belong to the same group.

We may also conjecture that some of the loci that are present in the wild-type root will not be present in the Dicer 2,3,4 mutant roots. We represent this conjecture with the vector

```
> d3dep <- c("wtRoot","wtRoot","wtRoot","wtRoot","dicerRoot","dicerRoot")
```

This vector indicates that the fifth and sixth samples, which consist of the wild-type root samples, are in a separate expression group to the other samples, corresponding to the Dicer 2,3,4 mutant.

Finally, we hypothesise that some of the small RNAs generated in the wild-type shoot will move to the root. We represent this hypothesis with the vector

```
> mobile <- c("dicerShoot","dicerShoot","wtShoot","wtShoot","wtShoot","wtShoot")
```

This vector shows that all samples with a wild-type shoot are distinct from those samples with a Dicer 2,3,4 shoot.

We can now add these models to the locus data by modfiying the `@groups` slot

```
> groups(cD) <- list(NDE = NDE, d3dep = d3dep, mobile = mobile)
```

Note that in this case the replicate structure does not correspond to any biologically plausible model; we do not expect that any loci will be different between all three experimental groups.

We can now find the priors and likelihoods for this analysis as before.

```
> cD <- getPriors.NB(cD, cl = cl)
> cD <- getLikelihoods(cD, nullData = TRUE, cl = cl)
```

We can see if there are any potential candidates for mobile sRNA loci by using the 'topCounts' function.

```
> topCounts(cD, group = "mobile", normaliseData = TRUE)

     chr    start      end SL236 SL260 SL237 SL238 SL239 SL240     likes
74     1   447231   447298     0     0   174   146   226   220 0.9999997
1111   1  8287590  8287674     0     0   107    83   101   156 0.9999980
2962   1 14188044 14188079     1     0    47    33    66    59 0.9999980
764    1  6127755  6127808     0     0    81    42   108    68 0.9999931
901    1  6880517  6880553     0     0    41    26    43    45 0.9999905
1334   1  9254068  9254167     0     0   101    81   111    83 0.9999655
1696   1 11140107 11140158     0     0    60    38    64    43 0.9999573
2298   1 13042720 13042777     2     4    46    31    58    67 0.9999542
1212   1  8766946  8767133   171   487    12     8    28    29 0.9999260
427    1  3440406  3440437     0     0    13    15    19    19 0.9998933
                mobile   FDR.mobile  FWER.mobile
74   wtShoot>dicerShoot 3.056093e-07 3.056093e-07
1111 wtShoot>dicerShoot 1.175909e-06 2.351817e-06
2962 wtShoot>dicerShoot 1.466957e-06 4.400866e-06
764  wtShoot>dicerShoot 2.825973e-06 1.130386e-05
901  wtShoot>dicerShoot 4.151618e-06 2.075795e-05
1334 wtShoot>dicerShoot 9.213774e-06 5.528178e-05
1696 wtShoot>dicerShoot 1.399326e-05 9.794960e-05
2298 wtShoot>dicerShoot 1.796906e-05 1.437447e-04
1212 dicerShoot>wtShoot 2.418928e-05 2.176851e-04
427  wtShoot>dicerShoot 3.243789e-05 3.243373e-04
```

We can also identify dicer-dependent root specific small RNA loci by examining our alternative model for differential expression.

```
> topCounts(cD, group = "d3dep", normaliseData = TRUE)

     chr    start      end SL236 SL260 SL237 SL238 SL239 SL240     likes           d3dep
2166   1 12726934 12726976     3     5     4     6    48    49 0.9987420 dicerRoot>wtRoot
1292   1  9013965  9014013     3     6     4     6    48    49 0.9987387 dicerRoot>wtRoot
1202   1  8741412  8741466     4     5     1     0    55    70 0.9981772 dicerRoot>wtRoot
```

```
2934   1 14154618 14154660    13    35    12    12   226   302 0.9981501 dicerRoot>wtRoot
2637   1 13689324 13689396     6     9     6     7    59    50 0.9979523 dicerRoot>wtRoot
2200   1 12824336 12824400     0     1     0     0    12     9 0.9944024 dicerRoot>wtRoot
1095   1  8238064  8238106     4     5     6     3    34    27 0.9887337 dicerRoot>wtRoot
795    1  6263246  6263343     0     2     3     0    23    24 0.9870628 dicerRoot>wtRoot
2982   1 14206419 14206455    16    29    29    17     9    13 0.9868232 wtRoot>dicerRoot
1160   1  8472567  8472649     0     3     1     2    25    20 0.9841345 dicerRoot>wtRoot
       FDR.d3dep  FWER.d3dep
2166 0.001257957 0.001257957
1292 0.001259616 0.002517646
1202 0.001447342 0.004335851
2934 0.001547978 0.006177717
2637 0.001647927 0.008212790
2200 0.002306212 0.013764456
1095 0.003586222 0.024875663
795  0.004755095 0.037491047
2982 0.005690838 0.050173812
1160 0.006708302 0.065243260
```

By including more experimental conditions in our analyses, increasingly complex patterns of expression can be detected from sequencing data.

Finally, we shut down the cluster (assuming it was started to begin with).

```
> if(!is.null(cl)) stopCluster(cl)
```

Session Info

```
> sessionInfo()

R Under development (unstable) (2024-11-20 r87352)
Platform: x86_64-apple-darwin20
Running under: macOS Monterey 12.7.6

Matrix products: default
BLAS:   /Library/Frameworks/R.framework/Versions/4.5-x86_64/Resources/lib/libRblas.0.dylib
LAPACK: /Library/Frameworks/R.framework/Versions/4.5-x86_64/Resources/lib/libRlapack.dylib;  LAPACK version

locale:
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: America/New_York
tzcode source: internal

attached base packages:
[1] parallel  stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] baySeq_2.41.0
```

```
loaded via a namespace (and not attached):
 [1] httr_1.4.7              cli_3.6.3               knitr_1.49
 [4] rlang_1.1.4             xfun_0.49               UCSC.utils_1.3.0
 [7] generics_0.1.3          jsonlite_1.8.9          statmod_1.5.0
[10] S4Vectors_0.45.2        BiocStyle_2.35.0        htmltools_0.5.8.1
[13] stats4_4.5.0            locfit_1.5-9.10         rmarkdown_2.29
[16] grid_4.5.0              abind_1.4-8             evaluate_1.0.1
[19] fastmap_1.2.0           yaml_2.3.10             IRanges_2.41.1
[22] GenomeInfoDb_1.43.1     BiocManager_1.30.25     compiler_4.5.0
[25] limma_3.63.2            edgeR_4.5.0             XVector_0.47.0
[28] lattice_0.22-6          digest_0.6.37           R6_2.5.1
[31] GenomeInfoDbData_1.2.13 GenomicRanges_1.59.1    tools_4.5.0
[34] zlibbioc_1.53.0         BiocGenerics_0.53.3
```

References

[1] Thomas J. Hardcastle and Krystyna A. Kelly. *baySeq: Empirical Bayesian Methods For Identifying Differential Expression In Sequence Count Data.* BMC Bioinformatics (2010).

[2] Thomas J. Hardcastle. *Generalised empirical Bayesian methods for discovery of differential data in high-throughput biology.* bioRχv preprint (2015).

[3] Attila Molnar and Charles W. Bassett and Thomas J. Hardcastle and Ruth Dunn and David C. Bauclombe *Small silencing RNAs in plants are mobile and direct epigenetic modification in recipient cells.* Science (2010).