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

Thomas J. Hardcastle

October 1, 2012

1 Introduction

This vignette is intended to give a rapid introduction to the commands used in implementing two methods of evaluating differential expression in Solexa-type, or *count* data by means of the `baySeq` `R` package. For fuller details on the methods being used, consult Hardcastle & Kelly [1]. The major improvement made in this release is the option to include region length in evaluating differential expression between genomic regions (e.g. genes). See Section **??** for more details.

We assume that we have discrete data from a set of sequencing or other high-throughput experiments, arranged in a matrix such that each column describes a sample and each row describes some entity for which counts exist. 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. This problem has been addressed for pairwise differential expression by the `edgeR` [3] package.

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. We also observe that the methods implemented in `baySeq` perform at least as well, and in some circumstances considerably better than those implemented in `edgeR` [1].

`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.

Our task is thus to determine the posterior likelihood of each model for each row of the data.

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 therefore need to load the `snow` package (if it exists), define a *cluster* and load the `baySeq` library onto each member of the cluster.

```
>   library(snow)
> cl <- makeCluster(4, "SOCK")
```

If `snow` 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.

```
>   cl <- NULL
```

Here we have (if the `snow` package is installed) defined a cluster of four processors on sockets; that is to say, on the local machine. If the local machine has multiple processors this may be a valid method of accelerating `baySeq`, but if very large data sets are being analysed we may wish to consider some other form of parallelisation (e.g. LAM/MPI) that allows processors on multiple nodes to be used. See the `snow` documentation for details on how to achieve this.

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)
```

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)))
```

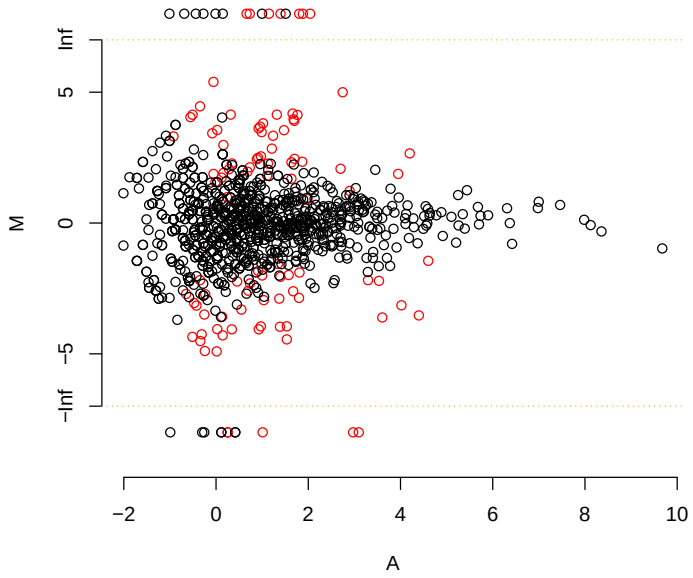

Figure 1: 'MA'-plot for count data. Where the log-ratio would be infinite (because the data in one of the sample groups consists entirely of zeros, we plot instead the log-values of the other group. Truly differentially expressed data are colored red, and non-differentially expressed data black.

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 = "_"))
```

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.

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

```
> CD <- getLikelihoods.NB(CD, pET = 'BIC', cl = cl)
```

.

```
> CD@estProps
```

```
[1] 0.8765202 0.1234798
```

```
> CD@posteriors[1:10,]
```

```
             NDE           DE
 [1,] -0.6379573 -0.751561830
 [2,] -0.9002858 -0.521639691
 [3,] -0.7711396 -0.620799949
 [4,] -2.2055580 -0.116746225
 [5,] -0.6307673 -0.759678681
 [6,] -1.1445501 -0.383264073
 [7,] -5.5615869 -0.003850076
 [8,] -3.8695783 -0.021087963
 [9,] -0.8983994 -0.522933531
[10,] -1.6797002 -0.206323140
```

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

```
                NDE        DE
 [1,] -6.066791e-02 -2.832521
 [2,] -7.991582e-05 -9.434577
 [3,] -4.640167e-02 -3.093531
 [4,] -1.701222e-02 -4.082318
 [5,] -4.290052e-03 -5.453601
 [6,] -5.109052e-02 -2.999593
 [7,] -7.850153e-02 -2.583631
 [8,] -4.885146e-02 -3.043297
 [9,] -6.376703e-02 -2.784233
[10,] -8.884361e-03 -4.727902
```

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

```
[1] 0.1234798
```

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
1  count_80      1      1      0      1      1     13     21      8      6     20
2  count_78      1      1      0      1      1      8     13      7      9     10
3  count_66      0      0      0      0      0     15     10      4      4     10
4  count_21      2      0      1      1      0     15     15      6      5     11
5   count_7      9      8      8      8      9      1      2      1      0      0
6  count_26     13      4     11      5      7      1      1      1      0      0
7  count_72      0      0      1      0      0      7      6      4      3      8
8  count_64      6      6      8     11      9      1      1      0      0      1
9  count_83     14      6      9      2      9      1      0      0      1      1
10 count_27      5      3      6      4      7      0      0      0      1      0
   Likelihood         FDR
1   0.9988118 0.001188193
2   0.9985328 0.001327718
3   0.9977532 0.001634062
4   0.9968802 0.002005505
5   0.9961573 0.002372939
6   0.9949830 0.002813616
7   0.9938530 0.003289813
8   0.9918470 0.003897714
9   0.9891786 0.004667009
10  0.9861448 0.005585832
```

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)))
```

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

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

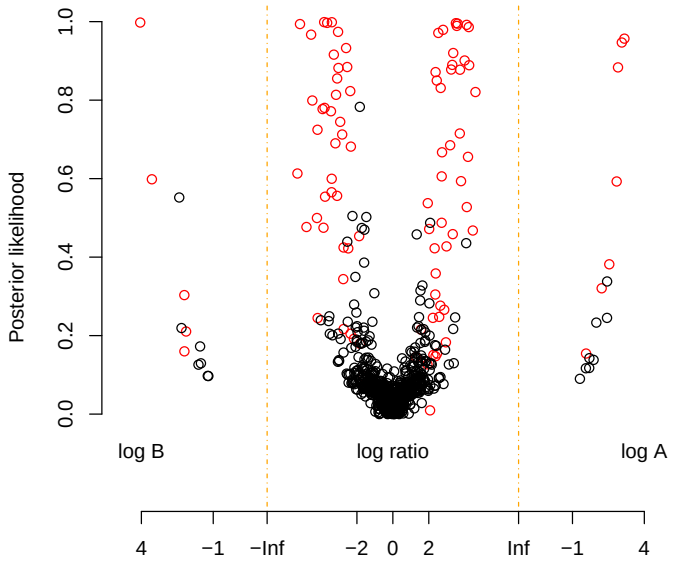

Figure 2: Posterior likelihoods of differential expression against log-ratio (where this would be non-infinite) or log values (where all data in the other sample group consists of zeros). Truly differentially expressed data are colored red, and non-differentially expressed data black.

5 Paired Data Analysis

There now exists functionality to analyse paired data through a similar process, using the beta-binomial distribution. The process for analysing paired data follows approximately the same steps as for analysing unpaired data. However, two different types of differential expression can exist within paired data. Firstly, we can find differential expression between replicate groups, as before. However, we can also find (consistent) differential expression between pairs; this would occur when for a single row of data, the first member of each pair differs from the second member of each pair. `baySeq` can identify both these types of differential expression simultaneously, and we implement this proceedure below.

We begin by loading a simulated dataset containing counts for four paired datasets.

```
>   data(pairData)
```

The first four columns in these data are paired with the second four columns. We construct a pairedData in a similar fashion to the countData object.

```
>    pairCD <- new("pairedData", data = pairData[,1:4], pairData = pairData[,5:8],
+                  replicates = c(1,1,2,2),
+                  groups = list(NDE = c(1,1,1,1), DE = c(1,1,2,2)))
```

We can find the library sizes for the data with the `getLibsizes` function.

```
>   libsizes(pairCD) <- getLibsizes(pairCD)
```

We estimate an empirical distribution on the parameters of a beta-binomial distribution by bootstrapping from the data, taking individual counts and finding the maximum likelihood parameters for a beta-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.

```
>   pairCD <- getPriors.BB(pairCD, samplesize = 1000, cl = cl)
```

We then acquire posterior likelihoods as before. The 'nullProps' parameter indicates that the proportion of counts observed in the first member of a non-differentially expressed pair is 0.5.

```
>   pairCD <- getLikelihoods.BB(pairCD, pET = 'BIC', nullProps = 0.5, cl = cl)
```

.

We can ask for the top candidates for differential expression between replicate groups using the `topCounts` function as before.

```
>   topCounts(pairCD, group = 2)
```

```
    rowID   X1.1  X1.2     X2.1   X2.2 Likelihood         FDR
1   row_5 159:73 44:24     0:49   0:68  0.9975207 0.002479339
2  row_53  709:0 895:0  373:191 124:60  0.9942650 0.004107194
3  row_35  53:12  19:7     0:77    0:6  0.9935244 0.004896654
```

```
4  row_96   25:0  73:0      8:3  36:13  0.9913705 0.005829873
5  row_65   80:0  48:0    36:50   12:3  0.9864768 0.007368533
6  row_24   63:0  21:0    47:80   6:13  0.9811678 0.009279149
7  row_90  268:0  39:0   74:107  98:36  0.9786982 0.010996675
8  row_71    8:0  15:0    21:16    2:1  0.9587300 0.014780844
9  row_68 123:63 38:36 1198:179 350:18  0.9535254 0.018302370
10 row_28  196:1  55:1    56:52  27:16  0.9529789 0.021174243
```

However, we can also look for consistent differential expression between the pairs. Since we set the 'nullProps' variable to 0.5, the first grouping model describes pairs which show no differential expression between replicate groups, but does show deviation from a one-to-one ratio of data between pairs.

```
>   topCounts(pairCD, group = 1)
```

```
     rowID    X1.1  X1.2    X2.1  X2.2 Likelihood        FDR
1  row_116   17:70  1:40   9:117  3:45  0.9785444 0.02145562
2  row_166 1027:27 835:8 1155:29 138:0  0.9693996 0.02602800
3  row_146    1:38  0:68    0:28  0:26  0.9597737 0.03076076
4  row_123     1:4  1:11     0:5  1:14  0.9510090 0.03531832
5  row_193    69:1  10:1  119:17  53:5  0.9437790 0.03949886
6  row_180     1:2  1:16    2:41   0:2  0.9423653 0.04252150
7  row_101    0:30   0:5    0:60  0:24  0.9372627 0.04540947
8  row_138    0:12   0:4     0:4  0:13  0.9351786 0.04783596
9  row_144     0:4  0:21     0:2  0:12  0.9342545 0.04982591
10 row_127     0:3  0:12    0:15   0:4  0.9330513 0.05153819
```

6 Case Study: Analysis of sRNA-Seq Data

6.1 Introduction

We will look at data sequenced from small RNAs acquired from six samples of root stock from *Arabidopsis thaliana* in a grafting experiment [2]. 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.

6.2 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)
```

6.3 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.

6.3.1 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")
```

6.4 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 = NULL)
```

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.

6.4.1 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.

```
> plotPriors(cDPair, group = "NDE")
```

We can use the `nullData = TRUE` option in the `getLikelihoods.NB` 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.NB(cDPair, nullData = TRUE, cl = NULL)
```

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"
[1] D3/D3 D3/D3 WT/D3 WT/D3
Levels: D3/D3 WT/D3

Slot "libsizes"
SL236 SL260 SL237 SL238
 7648 14708 10194  8372
```

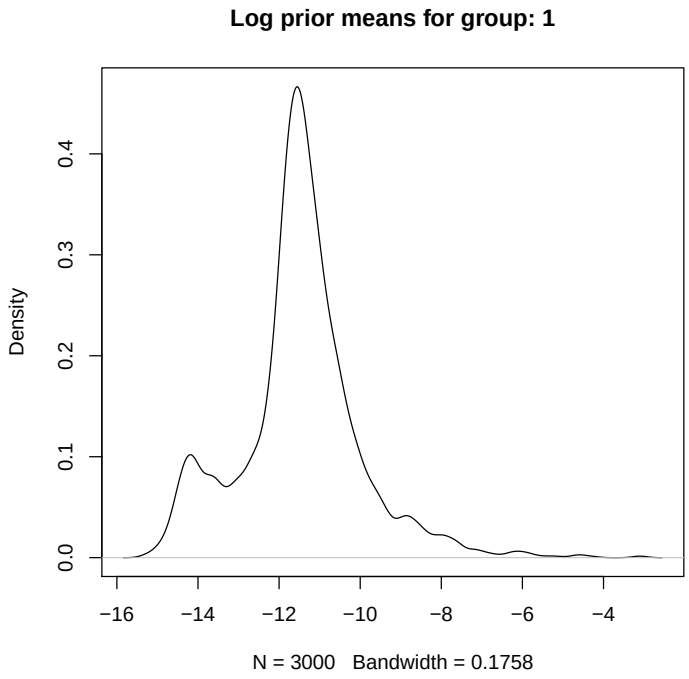

Figure 3: Distribution of μ_{ij} estimated as priors. Bimodality suggests the presence of 'null', or un-expressed, data.

```
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,]    21    52     4     4
[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,] 0.01554777 0.9844522
[2,] 0.13419341 0.8658066
[3,] 0.80158762 0.1983238
[4,] 0.29583616 0.7041638
[5,] 0.39458980 0.6054102
2995 more rows...

Slot "estProps":
[1] 0.5100932 0.3723167
```

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'.

```
> cDPair@estProps

[1] 0.5100932 0.3723167
```

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.

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

   chr    start      end SL236 SL260 SL237 SL238 Likelihood          FDR
1    1 11212437 11212516     1     2   397   299  0.9999682 3.176699e-05
2    1   447231   447298     0     0   174   146  0.9999283 5.175523e-05
3    1 13075806 13075879     2     6   145   114  0.9998582 8.177468e-05
4    1 13544660 13544713   114   227     5     4  0.9998098 1.088831e-04
5    1 13463357 13463459    12    20   165   140  0.9995180 1.835158e-04
6    1 10314530 10314578     4     3    91    71  0.9995124 2.342019e-04
```

```
7    1  8287590  8287674     0     0   107    83  0.9994666 2.769376e-04
8    1  9174281  9174537    24    43   351   245  0.9994328 3.132215e-04
9    1  5056092  5056161    65   184     1     0  0.9991752 3.700645e-04
10   1  8766946  8767133   171   487    12     8  0.9990798 4.250771e-04
```

Looking at the data in this way can be misleading. Because the library sizes of the different libraries differ, it can be unclear as to why some loci are identified as differentially expressed.

Try viewing the normalised results.

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

```
   chr    start      end SL236 SL260 SL237 SL238 Likelihood          FDR
1    1 11212437 11212516     1     1   385   354  0.9999682 3.176699e-05
2    1   447231   447298     0     0   169   173  0.9999283 5.175523e-05
3    1 13075806 13075879     3     4   141   135  0.9998582 8.177468e-05
4    1 13544660 13544713   148   153     5     5  0.9998098 1.088831e-04
5    1 13463357 13463459    16    13   160   166  0.9995180 1.835158e-04
6    1 10314530 10314578     5     2    88    84  0.9995124 2.342019e-04
7    1  8287590  8287674     0     0   104    98  0.9994666 2.769376e-04
8    1  9174281  9174537    31    29   341   290  0.9994328 3.132215e-04
9    1  5056092  5056161    84   124     1     0  0.9991752 3.700645e-04
10   1  8766946  8767133   221   328    12     9  0.9990798 4.250771e-04
```

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.

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

6.5 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`.

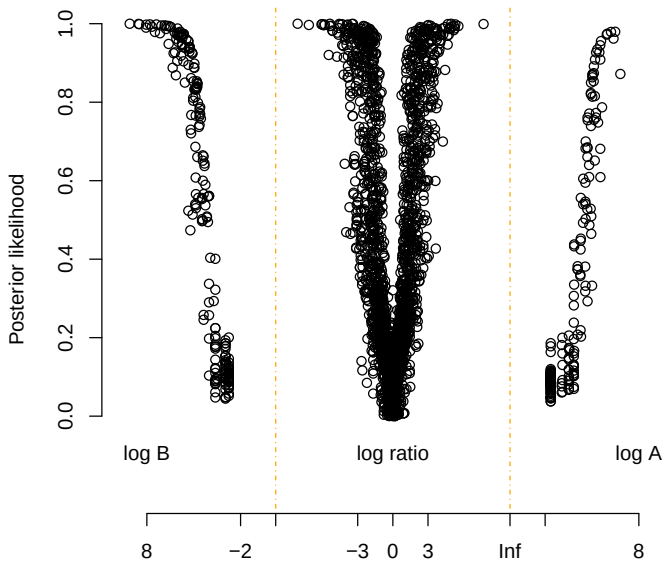

Figure 4: Posterior likelihoods of differential expression against log-ratios of the data. Where the data in one of the sample groups consists entirely of zeros, the log-ratio would be infinite. In this case, we plot instead the log-values of the non-zero group. Note the skew in the data; there are many 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.

```
> 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")
```

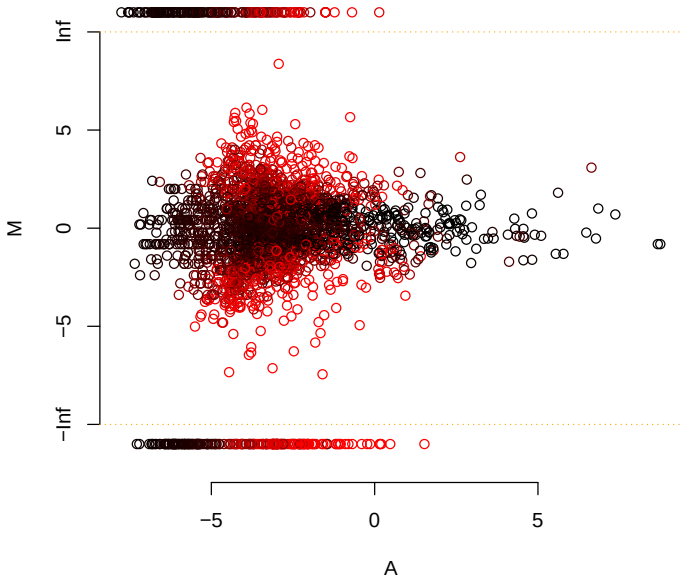

Figure 5: 'MA'-plot for count data. Where the data in one of the sample groups consists entirely of zeros, the log-ratio would be infinite. In this case, we plot instead the log-values of the non-zero group. Differentially expressed data are colored red, and non-differentially expressed data black.

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

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 Likelihood          FDR
1    1    447231    447298     0     0   202   206   163   151  0.9999994 6.347174e-07
```

```
2    1 14188044 14188079     2     0    54    47    48    40  0.9999970 1.817495e-06
3    1 10314530 10314578     6     2   105   100   117   132  0.9999948 2.931728e-06
4    1  8287590  8287674     0     0   124   117    73   107  0.9999946 3.551614e-06
5    1  6127755  6127808     0     0    94    59    78    47  0.9999843 5.973534e-06
6    1  6880517  6880553     0     0    48    37    31    31  0.9999822 7.946428e-06
7    1 12548300 12548396     0     1    65    51    53    45  0.9999731 1.065995e-05
8    1 11212437 11212516     2     2   460   422   233   223  0.9999719 1.284326e-05
9    1 13042720 13042777     3     3    53    44    42    46  0.9999552 1.638972e-05
10   1 13463357 13463459    19    16   191   198   305   258  0.9999437 2.038478e-05
```

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 Likelihood          FDR
1    1  3980795  3980853     2     4     1     3   215   213  0.9999958 4.182288e-06
2    1  9013965  9014013     5     5     5     8    35    34  0.9986507 6.767171e-04
3    1 12726934 12726976     5     4     5     8    35    34  0.9986385 9.049651e-04
4    1 14154618 14154660    20    28    14    17   163   207  0.9984781 1.059196e-03
5    1  8741412  8741466     6     4     1     0    40    48  0.9980139 1.244585e-03
6    1 13689324 13689396     9     7     7    10    42    34  0.9977336 1.414879e-03
7    1  6173399  6173503     0     0     2     1    27    18  0.9929428 2.220922e-03
8    1 14206419 14206455    25    23    34    24     6     9  0.9925952 2.868910e-03
9    1 12824336 12824400     0     1     0     0     9     6  0.9921380 3.423702e-03
10   1  8238064  8238106     6     4     7     4    24    18  0.9881460 4.266736e-03
```

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

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] 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).

[3] Mark Robinson `edgeR`*:' Methods for differential expression in digital gene expression datasets.* Bioconductor.