baySeq (version 1.1.4)

Thomas J. Hardcastle

October 28, 2009

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].

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` [2] 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 hypotheses 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 hypothesis about the relatedness of our underlying parameters for multiple libraries, the posterior likelihood of the hypothesis.

In forming a set of hypotheses 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 hypothesis for each row of the data. We can do this by considering either a Poisson or negative-binomial distribution upon the sequencing count data. The Possion method is considerably faster as a closed form conjugate prior exists for this distribution. The negative-binomial solution is slower as it requires a numerical solution for the prior, but is probably a better model for 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. 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.

```
> if ("snow" %in% installed.packages()[, 1]) {
+     library(snow)
+     cl <- makeCluster(4, "SOCK")
+     clusterEvalQ(cl, library(baySeq))
+ } else 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 and library sizes for ten libraries.

```
> data(simCount)
> data(libsizes)
> simCount[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
```

```
> libsizes
```

```
 [1] 75373 40153 75403 34285 55975 53287 80477 37655 41171 77510
```

The data are simulated such that the first hundred counts show differential expression between the first five libraries and the second five libraries. We can therefore establish two groups.

```
> 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))
```

Each member (vector) contained within the 'groups' list corresponds to one hypothesis upon the data. In this setting, a hypothesis 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' hypothesis, 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 hypothesis 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.

In a more complex experimental design (Section 1) we might have several additional hypotheses. The key to constructing vectors corresponding to a hypothesis is to see for which groups of libraries we expect equivalent expression of tags.

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

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

```
> CD <- new("countData", data = simCount, libsizes = libsizes,
+     groups = groups)
```

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 Poisson-Gamma Approach

We first try to identify the posterior likelihoods of each hypothesis for each tag assuming a Poisson distribution on each tag with a rate that is Gamma distributed. That is, if Y_{ij} is an element of the data where i is the row of the data, and j is the sample, then

$$Y_{ij} \sim Poi(\theta_j l_j)$$

where the l_j is the library size of sample j (or some other suitable scaling factor) and

$$\theta_j \sim \Gamma(\alpha_j, \beta_j)$$

The relationships between the α_j, β_j for each j are determined by the hypothesis being investigated such that, if and only if samples X and Y belong to the same group, then $\alpha_X = \alpha_Y$ and $\beta_X = \beta_Y$.

We begin by trying to establish the parameters of the Gamma distribution by bootstrapping from the data and applying maximum likelihood methods. We are able to adjust the parameters of the bootstrapping; here we take twenty sets of count data, establish the maximum likelihood Gamma parameters, and iterate over 1000 cases. In general more than 5000 iterations is recommended but is used here for speed of calculation.

We then take the mean of the maximum likelihood estimates to acquire a prior on the rate distribution.

```
> CDP.Poi <- getPriors.Pois(CD, samplesize = 20, iterations = 1000,
+     takemean = TRUE)
```

```
...
```

The calculated priors are stored in the `@priors` slot of the `countData` object produced.

```
> CDP.Poi@priors
```

```
$type
[1] "Poi"

$priors
$priors$NDE
         [,1]     [,2]
[1,] 1.206407 25190.92

$priors$DE
         [,1]     [,2]
[1,] 1.029182 21161.97
[2,] 1.013052 21434.31
```

For each hypothesis, we get a set of priors. In the Poisson-Gamma model, we get, for each group in the hypothesis, a pair of parameters which define the Gamma distribution that we shall use as a prior distribution for the rates of the Poisson distributions that model how many counts we see in each row of the data. Thus, in the hypothesis of differential expression, there are two groups in the data and we find two sets of parameters.

Having acquired a set of prior distributions on the rate parameter of the Poisson distribution, we can calculate the posterior likelihoods of each hypothesis for each tag. We need to pass an initial prior likelihood on each hypothesis; the `prs` parameter. If `estimatePriors = TRUE` then the prior likelihood on each hypothesis will be iteratively updated.

```
> CDPost.Poi <- getLikelihoods.Pois(CDP.Poi, prs = c(0.5, 0.5),
+     estimatePriors = TRUE, cl = cl)
> CDPost.Poi@estProps

      NDE        DE
0.6423602 0.3576398

> CDPost.Poi@posteriors[1:10, ]

             NDE            DE
 [1,]  -2.717427 -6.832647e-02
 [2,]  -7.204147 -7.437726e-04
 [3,]  -5.463705 -4.246830e-03
 [4,]  -7.789025 -4.143422e-04
 [5,]  -5.299484 -5.006681e-03
 [6,]  -3.607388 -2.749721e-02
 [7,] -16.188893 -9.316505e-08
 [8,]  -9.959837 -4.726156e-05
 [9,]  -6.085022 -2.279310e-03
[10,] -10.648984 -2.372522e-05

> CDPost.Poi@posteriors[101:110, ]

                NDE            DE
 [1,] -0.2654691450  -1.456056544
 [2,]  0.0000000000 -35.988199601
 [3,] -0.1811294853  -1.797741236
 [4,] -0.1394140979  -2.039203988
 [5,] -0.0004020790  -7.819063118
 [6,] -0.2093248234  -1.666705429
 [7,] -0.4325892455  -1.046476145
 [8,] -6.0744635473  -0.002303531
 [9,] -0.1964945482  -1.723759629
[10,] -0.0836090919  -2.523116302
```

The estimated posterior likelihoods for each hypothesis are stored in the natural logarithmic scale in the `@posteriors` slot of the `countDataPosterior`. The nth column of the posterior likelihoods matrix corresponds to the nth hypothesis as listed in the `group` slot of `CDPost.Poi`.

Here the assumption of a Poisson distribution gives an estimate of

```
       DE
0.3576398
```

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

4 Negative-Binomial Approach

We next try the same analysis assuming a negative binomial distribution on the data. 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 maximum 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.

```
> CDP.NBML <- getPriors.NB(CD, samplesize = 1000, estimation = "ML",
+     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 `$priors` of the slot `@priors`. Each member of this list object corresponds to one of the hypotheses defined by the `group` slot of the `countData` object and contains the estimated parameters for each of the individual counts selected under the hypotheses. The vector `$sampled` contained in the slot `@priors` describes which rows were sampled to create these sets of parameters.

We then acquire posterior likelihoods as before, estimating the proportions of differentially expressed counts. We can repeatedly bootstrap the prior estimatation to improve accuracy; here two bootstraps are used.

```
> CDPost.NBML <- getLikelihoods.NBboot(CDP.NBML, prs = c(0.5, 0.5),
+     estimatePriors = TRUE, bootStraps = 2, cl = cl)

[1] 0.8178803 0.1821197

> CDPost.NBML@estProps

      NDE        DE 
0.8178803 0.1821197 

> CDPost.NBML@posteriors[1:10, ]

             NDE           DE
 [1,] -1.3390060 -0.303955183
 [2,] -1.1987882 -0.358905184
 [3,] -1.4800102 -0.258298505
 [4,] -3.5398487 -0.029447058
 [5,] -0.8093841 -0.589025260
 [6,] -1.5809847 -0.230385164
 [7,] -6.1829954 -0.002066369
 [8,] -5.2022032 -0.005519629
 [9,] -1.6634897 -0.210075079
[10,] -2.3348771 -0.101836049

> CDPost.NBML@posteriors[101:110, ]
```

```
               NDE        DE
 [1,] -0.1216578759 -2.166755
 [2,] -0.0003962438 -7.833679
 [3,] -0.0780614305 -2.589036
 [4,] -0.0502453948 -3.015854
 [5,] -0.0090864870 -4.705507
 [6,] -0.1060212856 -2.296658
 [7,] -0.1254268417 -2.138091
 [8,] -0.0632335548 -2.792370
 [9,] -0.1269297712 -2.126915
[10,] -0.0210204438 -3.872752
```

Here the assumption of a negative binomial distribution with priors estimated by maximum likelihood gives an estimate of

```
       DE
0.1821197
```

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

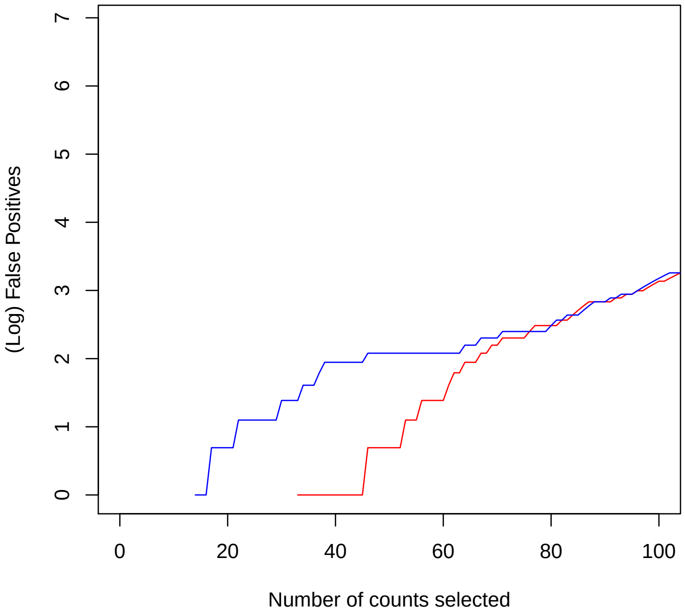
5 Results

We can ask for the top differentially expressed tags using the `topCounts` function.

```
> topCounts(CDPost.NBML, group = 2)
```

```
       name X1 X2 X3 X4 X5 X6 X7 X8 X9 X10          logP
1  count_80  1  1  0  1  1 13 21  8  6  20 -0.0004456254
2  count_78  1  1  0  1  1  8 13  7  9  10 -0.0008054183
3  count_26 13  4 11  5  7  1  1  1  0   0 -0.0012617713
4  count_66  0  0  0  0  0 15 10  4  4  10 -0.0016237465
5  count_21  2  0  1  1  0 15 15  6  5  11 -0.0017893779
6   count_7  9  8  8  8  9  1  2  1  0   0 -0.0020663689
7  count_72  0  0  1  0  0  7  6  4  3   8 -0.0030075749
8  count_83 14  6  9  2  9  1  0  0  1   1 -0.0032450746
9  count_64  6  6  8 11  9  1  1  0  0   1 -0.0042637985
10 count_27  5  3  6  4  7  0  0  0  1   0 -0.0050292472
```

We can compare the accuracy of the methods by considering the false positive rates.



False positive rates for the bootstrapped negative binomial approach with maximum likelihood priors (red) are much lower than for the Poisson-Gamma conjugacy approach (blue). This approach is therefore significantly more accurate, although potentially computationally more intensive and thus slower than the Poisson-Gamma conjugacy.

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

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

6 More Complex Experimental Designs

To illustrate the way in which a model is specified for more complex experimental designs, we consider a factorial design containing eight libraries. Table 1 describes the experimental design in more detail. Samples 1, 2, 3 & 4 are from condition A while samples 5, 6, 7 & 8 are from condition B. However, samples 1, 2, 5 & 6 have also been subjected to some condition C, while samples 3, 4, 7 & 8 have been subjected to some condition D.

	Condition A	Condition B
Condition C	Samples 1, 2	Samples 5, 6
Condition D	Samples 3, 4	Samples 7, 8

Table 1: An example factorial design experiment in which samples 1 and 2 are subjected to experimental conditions A and C, samples 3 and 4 are subjected to conditions B and C, samples 5 and 6 are subjected to conditions A and C and samples 7 and 8 are subjected to conditions B and D.

We prepare the baySeq library and cluster as before.

```
> library(baySeq)
> if ("snow" %in% installed.packages()[, 1]) {
+     library(snow)
+     cl <- makeCluster(4, "SOCK")
+     clusterEvalQ(cl, library(baySeq))
+ } else cl <- NULL
```

We load a simulated data set corresponding to the factorial design described. The first hundred cases show differential expression caused by differences between condition A and condition B, while the second hundred cases show differential expression caused by differences between condition C and condition D.

```
> data(factData)
> data(factlibsizes)
```

We establish three group structures on the data.

```
> factgroups <- list(NDE = c(1, 1, 1, 1, 1, 1, 1, 1), DE.A.B = c(1,
+     1, 1, 1, 2, 2, 2, 2), DE.C.D = c(1, 1, 2, 2, 1, 1, 2, 2))
```

The first group assumes no differential expression between samples. The second group assumes differential expression between samples experiencing condition A and samples experiencing condition B. The third group assumes differential expression between samples experiencing condition C and samples experiencing condition D.

We could also consider the possibility of interactions between effects, by considering a group c(1,1,2,2,3,3,4,4). However, in this simulated data set, no such data exists and so we need not consider this group. It should be noted, however, that such a group would only find that an interaction effect takes place in some elements of the data. Like an ANOVA test, it is necessary to examine the data to determine what form the effect takes.

Having established a group structure, we proceed as before.

```
> CDfact <- new("countData", data = factCount, libsizes = factlibsizes,
+     groups = factgroups)
> CDfact@annotation <- data.frame(name = paste("count", 1:1000,
+     sep = "_"))
> CDfactP.NBML <- getPriors.NB(CDfact, samplesize = 1000, estimation = "ML",
+     cl = cl)
> CDfactPost.NBML <- getLikelihoods.NBboot(CDfactP.NBML, prs = c(0.2,
+     0.3, 0.5), estimatePriors = TRUE, bootStraps = 2, cl = cl)

[1] 0.6312575 0.1877423 0.1810002

> CDfactPost.NBML@estProps

      NDE    DE.A.B    DE.C.D
0.6312575 0.1877423 0.1810002
```

We can then ask for the tags showing most differential expression caused by the difference between conditions A and B

```
> topCounts(CDfactPost.NBML, group = 2)
```

```
       name X1 X2 X3 X4  X5  X6  X7  X8          logP
1  count_94 19 41 19 22   3   2   3   4 -0.0006165511
2  count_41  2  5  4  1  29  11  29  23 -0.0045514750
3  count_81  6 12  5 10   0   0   0   0 -0.0049621226
4  count_88  0  0  0  0  10   8   6   8 -0.0059169326
5  count_13  2  6  6  6  32  32  59  37 -0.0060594547
6  count_22 31 50 34 26 446 113 275 194 -0.0070713930
7  count_60  5 14  6  6   0   0   1   0 -0.0073585387
8  count_49  5 19 14 12   2   1   2   1 -0.0099316842
9   count_7  0  1  1  0  19  16   9   8 -0.0153614010
10 count_75  1  4  1  0  30   7  33  14 -0.0177234017
```

And for those tags showing most differential expression caused by the difference between conditions C and D

```
> topCounts(CDfactPost.NBML, group = 3)
```

```
        name X1 X2 X3 X4 X5 X6 X7 X8          logP
1  count_138  3  1 21 23  2  1 29 26 -0.0002084306
2  count_126 53 78 10  3 47 39  7  6 -0.0018854110
3  count_161 15 27  1  1 10 15  1  0 -0.0024901288
4  count_200 18 16  0  1 11 10  1  2 -0.0045706901
5  count_180  1  2  6 11  0  1 14 15 -0.0053242908
6  count_125  2  1 11  8  2  0 19  8 -0.0085263061
7  count_155 10 14 40 56  5 12 81 73 -0.0109477317
8  count_105  0  3  8  5  0  0  9  8 -0.0191101292
9  count_166  6 18  1  0 24 15  0  3 -0.0268582083
10 count_186  9 58  1  3 24 15  3  4 -0.0277058862
```

References

[1] Thomas J. Hardcastle and Krystyna A. Kelly. *Empirical Bayesian methods for differential expression in count data*. In submission. 2009.

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