EBSeq: An R package for differential expression analysis using RNA-seq data

Ning Leng, John A. Dawson, Christina Kendziorski

April 20, 2012

Contents

1 Introduction

EBSeq may be used to identify differentially expressed (DE) genes and isoforms in an RNA-Seq experiment. As detailed in Leng *et al.*, 2012 [3], EBSeq is an empirical Bayesian approach that models a number of features observed in RNA-seq data. Importantly, for isoform level inference, EBSeq directly accommodates isoform expression estimation uncertainty by modeling the differential variability observed in distinct groups of isoforms. Consider Figure 1, where we

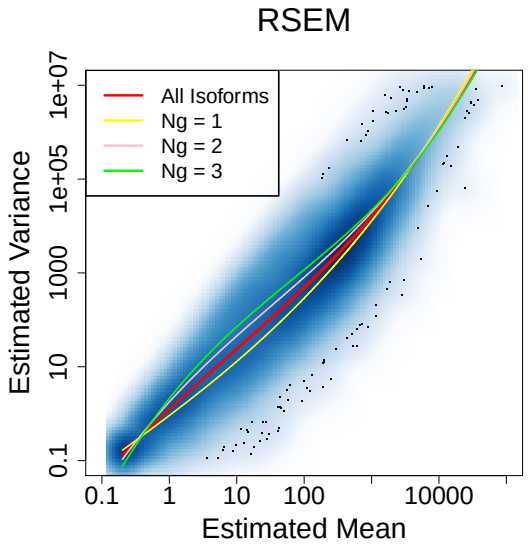

Figure 1: Empirical variance vs. mean for each isoform profiled in the mammary carcinoma experiment detailed in the Case Study section of Leng *et al.*, 2012 [3]. A spline fit to all isoforms is shown in red with splines fit within the $N_g = 1$, $N_g = 2$, and $N_g = 3$ isoform groups shown in yellow, pink, and green, respectively.

have plotted variance against mean for all isoforms using RNA-Seq expression data from Leng *et al.*, 2012 [3]. Also shown is the fit within three sub-groups of isoforms defined by the number of constituent isoforms of the parent gene. An isoform of gene g is assigned to the $N_g = k$ group, where $k = 1, 2, 3$, if the total number of isoforms from gene g is k (the $N_g = 3$ group contains all isoforms from genes having 3 or more isoforms). As shown in Figure 1, there is decreased variability in the $N_g = 1$ group, but increased variability in the others, due to the relative increase in uncertainty inherent in estimating isoform expression when multiple isoforms of a given gene are present. If this structure is not accommodated, there is reduced power for identifying isoforms in the $N_g = 1$ group (since the true variances in that group are lower, on average, than that derived from the full collection of isoforms) as well as increased false discoveries in the $N_g = 2$ and $N_g = 3$ groups (since the true variances are higher, on average, than those derived from the full collection). EBSeq directly models differential variability as a function of N_g providing a powerful approach for isoform level inference. As shown in Leng *et al.*, 2012 [3], the model is also useful for identifying DE genes. We will briefly detail the model in Section 2 and then describe the flow of analysis in Section 3 for both isoform and gene-level inference.

2 The Model

2.1 Two conditions

We let $X_{g_i}^{C1} = X_{g_i,1}, X_{g_i,2}, ..., X_{g_i,S_1}$ denote data from condition 1 and $X_{g_i}^{C2} = X_{g_i,(S_1+1)}, X_{g_i,(S_1+2)}, ..., X_{g_i,S}$ data from condition 2. We assume that counts within condition C are distributed as Negative Binomial: $X_{g_i,s}^{C} | r_{g_i,s}, q_{g_i}^{C} \sim NB(r_{g_i,s}, q_{g_i}^{C})$ where

$$P(X_{g_i,s}|r_{g_i,s}, q_{g_i}^{C}) = \binom{X_{g_i,s} + r_{g_i,s} - 1}{X_{g_i,s}} (1 - q_{g_i}^{C})^{X_{g_i,s}} (q_{g_i}^{C})^{r_{g_i,s}} \tag{1}$$

and $\mu_{g_i,s}^{C} = r_{g_i,s}(1 - q_{g_i}^{C})/q_{g_i}^{C}$; $(\sigma_{g_i,s}^{C})^2 = r_{g_i,s}(1 - q_{g_i}^{C})/(q_{g_i}^{C})^2$.

We assume a prior distribution on $q_{g_i}^{C}$: $q_{g_i}^{C} | \alpha, \beta^{N_g} \sim Beta(\alpha, \beta^{N_g})$. The hyperparameter α is shared by all the isoforms and β^{N_g} is N_g specific (note this is an index, not a power). We further assume that $r_{g_i,s} = r_{g_i,0} l_s$, where $r_{g_i,0}$ is an isoform specific parameter common across conditions and $r_{g_i,s}$ depends on it through the sample-specific normalization factor l_s. Of interest in this two group comparison is distinguishing between two cases, or what we will refer to subsequently as two patterns of expression, namely equivalent expression (EE) and differential expression (DE):

$$H_0 \text{ (EE)} : q_{g_i}^{C1} = q_{g_i}^{C2} \text{ vs } H_1 \text{ (DE)} : q_{g_i}^{C1} \neq q_{g_i}^{C2}.$$

Under the null hypothesis (EE), the data $X_{g_i}^{C1,C2} = X_{g_i}^{C1}, X_{g_i}^{C2}$ arises from the prior predictive distribution $f_0^{N_g}(X_{g_i}^{C1,C2})$:

$$f_0^{N_g}(X_{g_i}^{C1,C2}) = \left[\prod_{s=1}^{S} \binom{X_{g_i,s} + r_{g_i,s} - 1}{X_{g_i,s}} \right] \frac{beta(\alpha + \sum_{s=1}^{S} r_{g_i,s}, \beta^{N_g} + \sum_{s=1}^{S} X_{g_i,s})}{beta(\alpha, \beta^{N_g})} \tag{2}$$

Alternatively (in a DE scenario), $X_{g_i}^{C1,C2}$ follows the prior predictive distribution $f_1^{N_g}(X_{g_i}^{C1,C2})$:

$$f_1^{N_g}(X_{g_i}^{C1,C2}) = f_0^{N_g}(X_{g_i}^{C1}) f_0^{N_g}(X_{g_i}^{C2}) \tag{3}$$

Let the latent variable Z_{g_i} be defined so that $Z_{g_i} = 1$ indicates that isoform g_i is DE and $Z_{g_i} = 0$ indicates isoform g_i is EE, and $Z_{g_i} \sim Bernoulli(p)$. Then, the marginal distribution of $X_{g_i}^{C1,C2}$ and Z_{g_i} is:

$$(1 - p) f_0^{N_g}(X_{g_i}^{C1,C2}) + p f_1^{N_g}(X_{g_i}^{C1,C2}) \tag{4}$$

The posterior probability of being DE at isoform g_i is obtained by Bayes' rule:

$$\frac{p f_1^{N_g}(X_{g_i}^{C1,C2})}{(1 - p) f_0^{N_g}(X_{g_i}^{C1,C2}) + p f_1^{N_g}(X_{g_i}^{C1,C2})} \tag{5}$$

2.2 More than two conditions

EBSeq naturally accommodates multiple condition comparisons. For example, in a study with 3 conditions, there are K=5 possible expression patterns (P1,...,P5), or ways in which latent levels of expression may vary across conditions:

$$\begin{aligned}
&\text{P1: } q_{g_i}^{C1} = q_{g_i}^{C2} = q_{g_i}^{C3} \\
&\text{P2: } q_{g_i}^{C1} = q_{g_i}^{C2} \neq q_{g_i}^{C3} \\
&\text{P3: } q_{g_i}^{C1} = q_{g_i}^{C3} \neq q_{g_i}^{C2} \\
&\text{P4: } q_{g_i}^{C1} \neq q_{g_i}^{C2} = q_{g_i}^{C3} \\
&\text{P5: } q_{g_i}^{C1} \neq q_{g_i}^{C2} \neq q_{g_i}^{C3} \text{ and } q_{g_i}^{C1} \neq q_{g_i}^{C3}
\end{aligned}$$

The prior predictive distributions for these are given, respectively, by:

$$\begin{aligned}
g_1^{N_g}(X_{g_i}^{C1,C2,C3}) &= f_0^{N_g}(X_{g_i}^{C1,C2,C3}) \\
g_2^{N_g}(X_{g_i}^{C1,C2,C3}) &= f_0^{N_g}(X_{g_i}^{C1,C2}) f_0^{N_g}(X_{g_i}^{C3}) \\
g_3^{N_g}(X_{g_i}^{C1,C2,C3}) &= f_0^{N_g}(X_{g_i}^{C1,C3}) f_0^{N_g}(X_{g_i}^{C2}) \\
g_4^{N_g}(X_{g_i}^{C1,C2,C3}) &= f_0^{N_g}(X_{g_i}^{C1}) f_0^{N_g}(X_{g_i}^{C2,C3}) \\
g_5^{N_g}(X_{g_i}^{C1,C2,C3}) &= f_0^{N_g}(X_{g_i}^{C1}) f_0^{N_g}(X_{g_i}^{C2}) f_0^{N_g}(X_{g_i}^{C3})
\end{aligned}$$

where $f_0^{N_g}$ is the same as in equation 2. Then the marginal distribution in equation 4 becomes:

$$\sum_{k=1}^{5} p_k g_k^{N_g}(X_{g_i}^{C1,C2,C3}) \tag{6}$$

where $\sum_{k=1}^{5} p_k = 1$. Thus, the posterior probability of isoform g_i coming from pattern K is readily obtained by:

$$\frac{p_K g_K^{N_g}(X_{g_i}^{C1,C2,C3})}{\sum_{k=1}^{5} p_k g_k^{N_g}(X_{g_i}^{C1,C2,C3})} \tag{7}$$

3 Quick Start

Before analysis can proceed, the EBSeq package must be loaded into the working space:

```
> library(EBSeq)
```

3.1 Gene Level DE Analysis (Two Conditions)

3.1.1 Required input

Data: The object `Data` should be a $G - by - S$ matrix containing the expression values for each gene and each lane (sample), where G is the number of genes and S is the number of lanes. These values should exhibit raw counts, without normalization across samples. Counts of this nature may be obtained from RSEM [4], Cufflinks [7] or other such pre-processing approaches.
Conditions: The object `Conditions` should be a Factor vector of length S that indicates to which condition each sample belongs. For example, if there are two conditions and three samples in each, $S = 6$ and `Conditions` may be given by

```
as.factor(c("C1","C1","C1","C2","C2","C2"))
```

3.1.2 Simulating gene-level counts

The function `GeneSimu` may be used to simulate gene-level count data. As in [6] and [2], the function assumes counts are distributed as Negative Binomial with gene-specific mean in sample s and condition C given by $l_s\mu_g^C$ and variance $l_s\mu_g^C(1 + l_s\mu_g^C\phi_g)$. We first generate 10,000 genes and 5 samples for each of two conditions. Here we use `DEGeneProp = 0.1` to define the DE gene percentage, so that 10% of the genes will be generated as DE. The EE genes are simulated as $\mu_g^{C1} = \mu_g^{C2}$. The DE genes are simualted as half $\mu_g^{C1} = \delta_g\mu_g^{C2}$ and half $\mu_g^{C2} = \delta_g\mu_g^{C1}$. We could defined the DE genes to have constant δ_g equal to 4 by setting `DVDconstant = 4`; otherwise, the user may specify the `DVDqt1` and `DVDqt2` parameters to get non-constant δ_g from the corresponding lower and upper quantiles of the empirical δ_g values obtained from the data under investigation. (For example, let `DVDqt1=0.95` and `DVDqt2=0.97`. Then the δ_g will be randomly sampled from 95%-97% quantile of the empirical δ_g values.)

We will use ϕ_g values derived from the empirical data in the quantile range of 0.1 to 0.9; alternately, the user could set `Phiconstant` to a constant value in order to define a specific Negative Binomial distribution.

```
> GeneGenerate=GeneSimu(DVDconstant=4, DVDqt1=NULL, DVDqt2=NULL,
Conditions=rep(c("C1","C2"),each=5), NumofSample=10, NumofGene=10000,
DEGeneProp=.1, Phiconstant=NULL, Phi.qt1=.1, Phi.qt2=.9,
Meanconstant=NULL, OnlyData=T)
> GeneData=GeneGenerate$data
> GeneTrueDENames=GeneGenerate$TrueDE
```

The `GeneSimu` function is used to simulate a data matrix containing 10,000 rows of genes and 10 columns of samples. The genes are named `Gene_1, Gene_2 ...`

```
> str(GeneData)
 num [1:10000, 1:10] 1879 24 3291 97 485 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:10000] "Gene_1" "Gene_2" "Gene_3" "Gene_4" ...
  ..$ : NULL
```

where the first 10% of genes are simulated so that they are DE:

```
> str(GeneTrueDENames)
 chr [1:1000] "Gene_1" "Gene_2" "Gene_3" "Gene_4" "Gene_5" ...
```

3.1.3 Library size factor

As detailed in Section 2, EBSeq requires the library size factor l_s for each sample s. Here, l_s may be obtained via the function `MedianNorm`, which reproduces the median normalization approach to normalization used by DESeq [1].

```
> Sizes=MedianNorm(GeneData)
```

If quantile normalization is preferred, l_s may be obtained via the function `QuantileNorm`.

3.1.4 Running EBSeq on gene counts

The function `EBTest` is used to detect DE genes. For gene-level data, we don't need to specify the parameter `NgVector` since there are no differences in N_g structure among the different genes. Here, we simulated the first five lanes to be in condition 1 and the other five in condition 2, so define:

```
Conditions=as.factor(rep(c("C1","C2"),each=5))
```

`sizeFactors` is used to define the library size factor of each lane. It could be obtained by summing up the total number of reads within each lane, Median Normalization [1], scaling normalization [5], or some other such approach. These in hand, we run the EM, setting the number of iterations to five via `maxround=5`. Please note this may take several minutes:

```
> EBres=EBTest(Data=GeneData,
Conditions=as.factor(rep(c("C1","C2"),each=5)),sizeFactors=Sizes, maxround=5)
```

The posterior probabilities of being DE are obtained as follows, where PP is a vector containing the posterior probabilities of being DE for each of the 10000 simulated genes:

```
> PP=GetPP(EBres)
> str(PP)
> str(PP)
 Named num [1:10000] 1 1 1 1 1 ...
 - attr(*, "names")= chr [1:10000] "Gene_1" "Gene_2" "Gene_3" "Gene_4" ...
```

The vector PP may be used to form an FDR-controlled list of DE genes with a target FDR of 0.05 as follows:

```
> DEfound=names(PP)[which(PP>=.95)]
> str(DEfound)
 chr [1:991] "Gene_1" "Gene_2" "Gene_3" "Gene_4" "Gene_5" ...
> sum(DEfound%in%GeneTrueDENames)
[1] 959
```

EBSeq found 991 DE genes in total, and we see that 959 of them were true positives.

3.1.5 Checking the model fit and other diagnostics

As noted in Leng *et al.*, 2012 [3], EBSeq relies on parametric assumptions that should be checked following each analysis. The `QQP` function may be used to check the QQ plot of the empirical q's vs the simulated q's from the fitted beta prior distribution (see Figure 2). We can check the fit using the data from condition 1 in this way:

```
> QQP(QList=EBres$QList1, AlphaResult=EBres[[1]][5,1],
BetaResult=EBres[[2]][5,1], name="Gene Simulation", AList="F", GroupName=NULL)
```

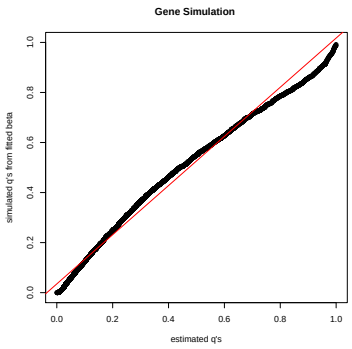

Figure 2: The QQ plot for checking the model fitting

Here we see no violation of the Beta assumption. Likewise, the `DenNHist` function may be used to check the density plot of empirical q's vs the simulated q's from the fitted beta prior distribution (see Figure 3).

```
> DenNHist(QList=EBres$QList1, Alpha=EBres[[1]][5,1], Beta=EBres[[2]][5,1],
name="Gene Simulation", AList="F", GroupName=NULL)
```

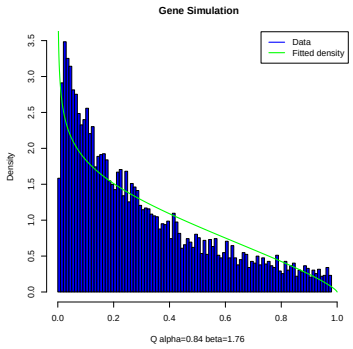

Figure 3: The density plot for checking the model fitting

3.2 Isoform Level DE Analysis (Two Conditions)

3.2.1 Required inputs

Data: The object `Data` should be a $I - by - S$ matrix containing the expression values for each isoform and each lane, where I is the number of isoforms and S is the number of lanes. Again, these values should exhibit raw data, without normalization across samples.
Conditions: The object `Conditions` should be a vector with length S to indicate the condition of each sample.
IsoformNames: The object `IsoformNames` should be a vector with length I to indicate the isoform names.
IsosGeneNames: The object `IsosGeneNames` should be a vector with length I to indicate the gene name of each isoform. (in the same order as `IsoformNames`.)

3.2.2 Simulating isoform-level counts

In order to simulate isoform-level data, the function `IsoSimu` may be used, where `NumofIso` defines the number of isoforms in each N_g group:

```
> IsoGenerate=IsoSimu(DVDconstant=NULL, DVDqt1=.97, DVDqt2=.98,
Conditions=as.factor(rep(c("C1","C2"),each=5)), NumofSample=10,
NumofIso=c(1000,2000,3000), DEIsoProp=.1, Phiconstant=NULL,
Phi.qt1=.25, Phi.qt2=.75, OnlyData=T )
> str(IsoGenerate)
List of 2
 $ data  :List of 3
  ..$ : num [1:1000, 1:10] 4824 210 3374 871 8917 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:1000] "Iso_1_1" "Iso_1_2" "Iso_1_3" "Iso_1_4" ...
  .. .. ..$ : NULL
```

```
  ..$ : num [1:2000, 1:10] 35 249 219 4 48 12 134 76 85 29 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:2000] "Iso_2_1" "Iso_2_2" "Iso_2_3" "Iso_2_4" ...
  .. .. ..$ : NULL
  ..$ : num [1:3000, 1:10] 1567 1368 125 77 307 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : chr [1:3000] "Iso_3_1" "Iso_3_2" "Iso_3_3" "Iso_3_4" ...
  .. .. ..$ : NULL
 $ TrueDE: chr [1:600] "Iso_1_1" "Iso_1_2" "Iso_1_3" "Iso_1_4" ...
```

Here, we simulated 6,000 isoforms in total. The number of isoforms in $N_g = 1, 2, 3$ groups are 1,000, 2,000 and 3,000, respectively. `TrueDENames` is a vector containing all the isoforms that are truly DE. Since `EBTest` requires a matrix that contains all of the isoform expressions, we need to convert the list `IsoGenerate$data` into a matrix:

```
> IsoMat=do.call(rbind,IsoGenerate$data)
> str(IsoMat)
 num [1:6000, 1:10] 4824 210 3374 871 8917 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:6000] "Iso_1_1" "Iso_1_2" "Iso_1_3" "Iso_1_4" ...
  ..$ : NULL
```

3.2.3 Library size factor

Similar to the gene-level analysis presented above, we may obtain the isoform-level library size factors via `MedianNorm`:

```
> IsoSizes=MedianNorm(IsoMat)
```

3.2.4 The N_g vector

Since EBSeq fits rely on N_g, we need to obtain the N_g for each isoform. This can be done using the function `GetNg`. The required inputs of `GetNg` are the isoforms names (`IsoformNames`) and their corresponding gene names (`IsosGeneNames`), described above. In the simulated data, we assume that the isoforms in the $N_g = 1$ group belong to genes `Gene_1, ... , Gene_1000`; The isoforms in the $N_g = 2$ group belong to genes `Gene_1001, ..., Gene_2000`; and isoforms in the $N_g = 3$ group belong to `Gene_2001, ..., Gene_3000`.

```
> IsoNames=rownames(IsoMat)
> str(IsoNames)
 chr [1:6000] "Iso_1_1" "Iso_1_2" "Iso_1_3" "Iso_1_4" "Iso_1_5" ...
> GeneNames=paste("Gene",c(1:3000),sep="_")
> IsosGeneNames=c(GeneNames[1:1000],rep(GeneNames[1001:2000],each=2),
rep(GeneNames[2001:3000],each=3))
> NgList=GetNg(IsoNames, IsosGeneNames)
> IsoNgTrun=NgList$IsoformNgTrun
```

```
> IsoNgTrun[c(1:3,1001:1003,3001:3003)]
Iso_1_1 Iso_1_2 Iso_1_3 Iso_2_1 Iso_2_2 Iso_2_3 Iso_3_1 Iso_3_2 Iso_3_3
      1       1       1       2       2       2       3       3       3
```

3.2.5 Running EBSeq on isoform counts

The `EBTest` function is also used to run EBSeq on isoform-level data:

```
> IsoEBres=EBTest(Data=IsoMat, NgVector=IsoNgTrun,
Conditions=as.factor(rep(c("C1","C2"),each=5)),sizeFactors=IsoSizes, maxround=5)
> IsoPP=GetPP(IsoEBres)
> str(IsoPP)
 Named num [1:6000] 1 1 1 1 1 ...
 - attr(*, "names")= chr [1:6000] "Iso_1_1" "Iso_1_2" "Iso_1_3" "Iso_1_4" ...
> IsoDE=IsoPP[which(IsoPP>=.95)]
> str(IsoDE)
 Named num [1:550] 1 1 1 1 1 ...
 - attr(*, "names")= chr [1:550] "Iso_1_1" "Iso_1_2" "Iso_1_3" "Iso_1_4" ...
> sum(names(IsoDE)%in%IsoGenerate$TrueDE)
[1] 511
```

We see that EBSeq found 550 DE isoforms at the target FDR of 0.05 and that 511 of those were true positives.

3.2.6 Checking the model fit and other diagnostics

If it is of interest to check differences among isoform groups defined by N_g (such as those shown in Figure 1), the function `PolyFitValue` may be used. The following code generates the first three panels shown in Figure 4:

```
> par(mfrow=c(2,2))
> PolyFitValue=vector("list",3)
> for(i in 1:3)
>           PolyFitValue[[i]]=PolyFitPlot(IsoEBres$C1Mean[[i]],
              IsoEBres$C1EstVar[[i]],5)
```

Superimposing all N_g groups using the code below will generate the figure shown in the lower right panel of Figure 4:

```
> PolyAll=PolyFitPlot(unlist(IsoEBres$C1Mean), unlist(IsoEBres$C1EstVar),5)
> lines(log10(IsoEBres$C1Mean[[1]][PolyFitValue[[1]]$sort]),
PolyFitValue[[1]]$fit[PolyFitValue[[1]]$sort],col="yellow")
> lines(log10(IsoEBres$C1Mean[[2]][PolyFitValue[[2]]$sort]),
PolyFitValue[[2]]$fit[PolyFitValue[[2]]$sort],col="pink")
> lines(log10(IsoEBres$C1Mean[[3]][PolyFitValue[[3]]$sort]),
PolyFitValue[[3]]$fit[PolyFitValue[[3]]$sort],col="green")
> legend("topleft",c("All Isoforms","Ng = 1","Ng = 2","Ng = 3"),
col=c("red","yellow","pink","green"),lty=1,lwd=3,box.lwd=2)
```

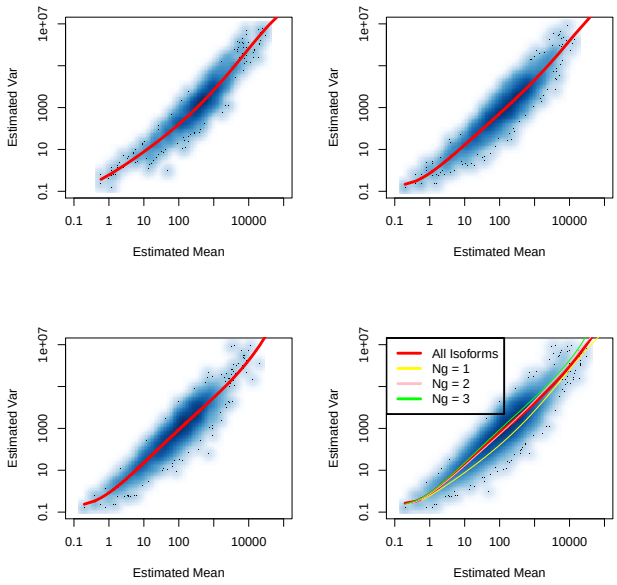

Figure 4: The Mean-Variance fitting plot for each Ng group

To generate a QQ plot of the fitted beta prior distribution and the estimated q-values within condition 1, a user may do the following (as in the gene-level analysis):

```
> par(mfrow=c(2,2))
> QQP(QList=IsoEBres$QList1, AlphaResult=IsoEBres[[1]][5,],
 BetaResult=IsoEBres[[2]][5,],
name="Isoforms", AList="F", GroupName=paste("Ng = ",c(1:3),sep=""))
```

And in order to produce the plot of the densities of fitted beta prior distribution and the histograms of estimated q-values within Condition 1 (see Figure 6), the following would be used:

```
> DenNHist(QList=IsoEBres$QList1, Alpha=IsoEBres[[1]][5,],
Beta=IsoEBres[[2]][5,],
name="Isoforms", AList="F", GroupName=paste("Ng = ",c(1:3),sep=""))
```

3.3 Working with more than two conditions

In analyses where the data are spread over more than two conditions, the set of possible patterns for each gene is more complicated than simply EE and DE. As noted in Section 2, when we have 3 conditions, there are 5 expression patterns to consider. Suppose, for example, that we have 6 samples, 2 in each of

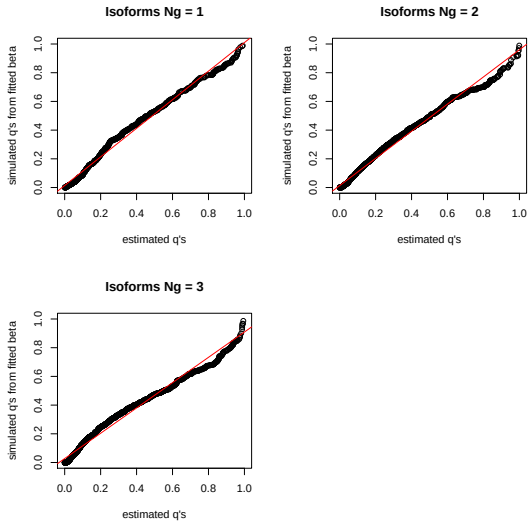

Figure 5: The QQ plot of the priore distribution fitting within each Ng group

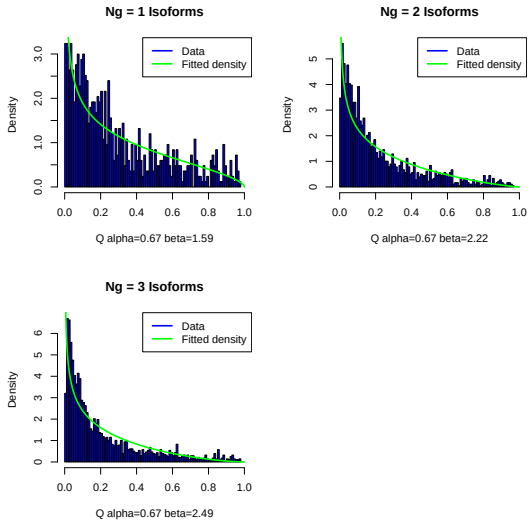

Figure 6: The prior distribution fitting within each Ng group

3 conditions. The function `GetPatterns` allows the user to generate all possible patterns given the conditions. For example:

```
> Conditions=c("C1","C1","C2","C2","C3","C3")
> PosParti=GetPatterns(Conditions)
```

```
> PosParti
                         C1 C2 C3
                Pattern1  1  1  1
                Pattern2  1  1  2
                Pattern3  1  2  1
                Pattern4  1  2  2
                Pattern5  1  2  3
```

where the first row means all three conditions have the same latent mean expression level; the second row means C1 and C2 have the same latent mean expression level but that of C3 is different; and the last row corresponds to the case where the three conditions all have different latent mean expression levels. The user may use all or only some of these possible patterns as an input to `EBMultiTest` (more on this function presently). For example, if we were interested in Patterns 1, 2, 4 and 5 only, we'd define:

```
> Parti=PosParti[-3,]
> Parti
                         C1 C2 C3
                Pattern1  1  1  1
                Pattern2  1  1  2
                Pattern4  1  2  2
                Pattern5  1  2  3
```

This established, we simulate 1,000 genes with 6 samples. The proportions of genes in each of our four patterns are $(0.7, 0.1, 0.1, 0.1)$:

```
> MultiData=GeneMultiSimu(Conditions=Conditions,AllParti=Parti,
          NumofSample=6,NumofGene=1000,DEGeneProp=c(.7,.1,.1,.1),
          DVDqt1=.98,DVDqt2=.99,Phi.qt1=.25,Phi.qt2=.75)
> str(MultiData)
List of 2
 $ data    : num [1:1000, 1:6] 127 85 2231 323 28 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:1000] "Gene_1" "Gene_2" "Gene_3" "Gene_4" ...
  .. ..$ : NULL
 $ Patterns: Named chr [1:1000] "Pattern2" "Pattern2" "Pattern2" "Pattern2" ...
  ..- attr(*, "names")= chr [1:1000] "Gene_1" "Gene_2" "Gene_3" "Gene_4" ...
```

`MultiData$data` provides the expression matrix. `MultiData$Patterns` provides the true pattern each gene belongs to.

Moving on to the analysis, `MedianNorm` or one of its competitors should be used to determine the normalization factors. Once this is done, the formal test is performed by `EBMultiTest`.

```
> MultiSize=MedianNorm(MultiData$data)
```

```
> MultiRes=EBMultiTest(MultiData$data,NgVector=NULL,Conditions=Conditions,
           AllParti=Parti, sizeFactors=MultiSize, maxround=5)
```

The posterior probobility of being in each pattern for every gene is obtained by using the function `GetMultiPP`:

```
> MultiPP=GetMultiPP(MultiRes)
> names(MultiPP)
[1] "PP"       "MAP"      "Patterns"
> MultiPP$PP[1:10,]
             Pattern1  Pattern2      Pattern4     Pattern5
Gene_1   1.510967e-74 1.0000000  2.266077e-73 4.279187e-19
Gene_2   9.297752e-12 0.7597022  5.929096e-13 2.402978e-01
Gene_3   5.335801e-82 0.9107056  1.284363e-50 8.929443e-02
Gene_4   5.751977e-75 0.8692036  1.769043e-46 1.307964e-01
Gene_5   5.894841e-04 0.8182709  3.059064e-04 1.808337e-01
Gene_6  5.609515e-237 1.0000000 3.110408e-191 2.569921e-19
Gene_7   6.277301e-02 0.7066661  8.666257e-03 2.218946e-01
Gene_8   1.553140e-29 0.9452813  7.932557e-26 5.471867e-02
Gene_9   0.000000e+00 1.0000000  0.000000e+00 1.117557e-19
Gene_10  1.247370e-07 0.8670365  5.324147e-06 1.329580e-01
> MultiPP$MAP[1:10]
 [1] "Pattern2" "Pattern2" "Pattern2" "Pattern2" "Pattern2" "Pattern2"
 [7] "Pattern2" "Pattern2" "Pattern2" "Pattern2"
> MultiPP$Patterns
          C1 C2 C3
Pattern1  1  1  1
Pattern2  1  1  2
Pattern4  1  2  2
Pattern5  1  2  3
```

where `MultiPP$PP` provides the posterior probobility of being in each pattern for every gene. `MultiPP$MAP` provides the most likely pattern of each gene based on the posterior probabilities. `MultiPP$Patterns` provides the details of the patterns.

```
> sum(MultiPP$MAP==MultiData$Patterns)
[1] 900
```

EBSeq made 900 correct calls out of 1,000 genes.

References

[1] S Anders and W Huber. Differential expression analysis for sequence count data. *Genome Biology*, 11:R106, 2010.

[2] T J Hardcastle and K A Kelly. bayseq: empirical bayesian methods for identifying differential expression in sequence count data. *BMC Bioinformatics*, 11:422, 2010.

[3] N. Leng, J.A. Dawson, J.A Thomson, V Ruotti, R. A. Rissman, B.M.G Smits, J.D. Hagg, M.N. Gould, R.M. Stwart, and C. Kendziorski. Ebseq: An empirical bayes hierarchical model for inference in rna-seq experiments. *BMI technical report, University of Wisconsin Madison*, 226, 2012.

[4] B Li and C N Dewey. Rsem: accurate transcript quantification from rna-seq data with or without a reference genome. *BMC Bioinformatics*, 12:323, 2011.

[5] M D Robinson and Oshlack A. A scaling normalization method for differential expression analysis of rna-seq data. *Genome Biology*, 11:R25, 2010.

[6] M D Robinson and G K Smyth. Moderated statistical tests for assessing differences in tag abundance. *Bioinformatics*, 23(21):2881–2887, 2007.

[7] C Trapnell, A Roberts, L Goff, G Pertea, D Kim, D R Kelley, H Pimentel, S L Salzberg, J L Rinn, and L Pachter. Differential gene and transcript expression analysis of rna-seq experiments with tophat and cufflinks. *Nature Protocols*, 7(3):562–578, 2012.