EBSeqHMM: An R package for identifying gene-expression changes in ordered RNA-seq experiments

Ning Leng and Christina Kendziorski

April 27, 2020

Contents

1 Introduction

EBSeqHMM (as detailed in Leng *et al.* (2014)) is an empirical Bayesian approach that models a number of features observed in ordered RNA-seq experiments (time course, spatial course, etc.). In an ordered RNA-seq experiment, of primary interest is characterizing how genes are changing over time, space, gradient, etc. For example, an investigator may be interested in genes that are monotonically increasing (or decreasing),

that increase initially then decrease, that increase initially then remain unchanged, and so on. We refer to these types of changes in expression hereinafter as expression paths. We classify expression paths into three main categories: (i) constant paths: expression remains unchanged, or equally expressed (EE), over all conditions; (ii) sporadic paths: expression shows some change between at least one pair of adjacent conditions, but remains unchanged between at least one other pair; and (iii) dynamic paths: expression changes continuously.

EBSeqHMM provides functions for identifying differentially expressed (DE) genes (genes that are not in category i), characterizing their changes over conditions, and clustering genes with similar paths. In EBSeqHMM, an autoregressive hidden Markov model is implemented to accommodate dependence in gene expression across ordered conditions.

EBSeqHMM may also be used for inference regarding isoform expression. Importantly, for isoform level inference, EBSeqHMM directly accommodates isoform expression estimation uncertainty by modeling the differential variability observed in distinct groups of isoforms. In short, it is more challenging to estimate isoform expressions. There is decreased variability in some isoforms, 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 some groups of isoforms as well as increased false discoveries in the other groups. Similar to EBSeq, EBSeqHMM directly models differential uncertainty as a function of I_g providing a powerful approach for isoform level inference.

2 The model

2.1 EBSeqHMM model

EBSeqHMM requires estimates of gene or isoform expression collected over three or more ordered conditions. To simplify the presentation, we refer to ordered conditions as time points denoted by $t = 1, 2, \ldots, T$, noting that the method directly accommodates other ordered data structures (e.g. space, gradient, etc.).

Let $\mathbf{X_t}$ be a $G \times N_t$ matrix of expression values for G genes in N_t subjects at time t. The full set of observed expression values is then denoted by $\mathbf{X} = (\mathbf{X_1}, \mathbf{X_2}, \ldots, \mathbf{X_T})$. With a slight abuse of notation, let $\mathbf{X_g}$ denote one row of this matrix containing data for gene g over time; X_{gtn} denotes expression values for gene g at time t in sample n. Of interest are changes in the latent mean expression levels for gene g: $\mu_{g1}, \mu_{g2}, \ldots, \mu_{gT}$. We allow for three possibilities, or states, to describe such changes: Up, Down, EE. If $\mu_{t-1} < \mu_t$, we define state $S^{\Delta t}$ as Up; if $\mu_{t-1} > \mu_t$, $S^{\Delta t}$ is Down; and $\mu_{t-1} = \mu_t$ implies $S^{\Delta t}$ is EE.

The main goals in an ordered RNA-seq experiment - identifying genes that change over time, and specifying each genes' expression path - can be restated as questions about these underlying states. In short, for each gene g and each transition between $t-1$ and t, EBSeqHMM estimates the probability of each state at each transition.

2.2 Getting a false discovery rate (FDR) controlled list of DE genes or isoforms

In an RNA-seq experiment with ordered conditions, DE genes (non-constant genes) are defined as those showing significant change in at least one condition. To obtain a list of DE genes with false discovery rate (FDR) controlled at α, DE genes are the ones whose posterior probability (PP) of following the constant path is less than α. Examples are provided in Section 3.1.5. Isoform-based lists are obtained in the same way (examples are provided in Section 3.2.5).

2.3 Getting clusters of genes or isoforms following the same expression path

The most likely path of a DE gene is defined as the path with highest posterior probability. DE genes with confident assignments may be further grouped into gene clusters depending on their expression paths. By the default settings, a gene will be called as a confident assignment if the gene's maximum PP of belonging

to a certain non-constant path exceeds 0.5. Examples are provided in Section 3.1.6, And we note that the 0.5 threshold can be changed. This, too is discussed in Section 3.1.6. Isoform-based lists are obtained in the same way (examples are provided in Section 3.2.6).

3 Quick start

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

```
> library(EBSeq)
> library(EBSeqHMM)
```

3.1 Gene level analysis

3.1.1 Required inputs

Data: The object `Data` should be a $G - by - S$ matrix containing the expression values for each gene and each sample, where G is the number of genes and S is the number of samples. These values should exhibit estimates of gene expression, without normalization across samples. Counts of this nature may be obtained from RSEM (Li and Dewey (2011)), Cufflinks (Trapnell *et al.* (2012)), or a similar approach.

Conditions: The object `Conditions` should be a factor of length S that indicates to which condition each sample belongs. Note the order of levels in the factor should represent the order in the RNA-seq experiments. For example
The object `GeneExampleData` is a simulated data matrix containing 100 rows of genes and 15 columns of samples. The genes are named `Gene_1, Gene_2 ...`

```
> data(GeneExampleData)
> str(GeneExampleData)

 num [1:100, 1:15] 179 1063 435 170 195 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:100] "Gene_1" "Gene_2" "Gene_3" "Gene_4" ...
  ..$ : NULL
```

Here we simulated triplicates for 5 time points (conditions). To specify which condition each sample belongs, we define:

```
> CondVector <- rep(paste("t",1:5,sep=""),each=3)
> print(CondVector)

 [1] "t1" "t1" "t1" "t2" "t2" "t2" "t3" "t3" "t3" "t4" "t4" "t4" "t5" "t5" "t5"
```

Downstream analysis by EBSeqHMM requires the conditions to be specified as a factor. In particular, levels of the factor need to be sorted along the time/spatial course. For example, to generate a factor with ordered conditions from t1 to t5, we define:

```
> Conditions <- factor(CondVector, levels=c("t1","t2","t3","t4","t5"))
> str(Conditions)

 Factor w/ 5 levels "t1","t2","t3",..: 1 1 1 2 2 2 3 3 3 4 ...

> levels(Conditions)

[1] "t1" "t2" "t3" "t4" "t5"
```

3.1.2 Library size factor

EBSeqHMM requires library size factors to adjust for sequencing depth differences among different samples. Here, the library size factors may be obtained via the function `MedianNorm`, which reproduces the median normalization approach in DESeq (Anders and Huber, 2010).

```
> Sizes <- MedianNorm(GeneExampleData)
```

If quantile normalization is preferred, library size factors may be obtained via the function `QuantileNorm` (e.g. `QuantileNorm(GeneMat,.75)` for Upper-Quantile Normalization in Bullard *et al.* (2010)).

3.1.3 Visualizing genes of interest

A user may want to look at expression paths of genes of interest prior to the analysis. EBSeqHMM provides the function `GetNormalizedMat` to generate the normalized matrix and the function `PlotExp` to generate longitudinal plots over the ordered conditions.

The normalized matrix may be obtained by :

```
> GeneNormData <- GetNormalizedMat(GeneExampleData, Sizes)
```

Suppose we are particularly interested in `Gene_23`, we may apply:

```
> PlotExp(GeneNormData, Conditions, Name="Gene_23")
```

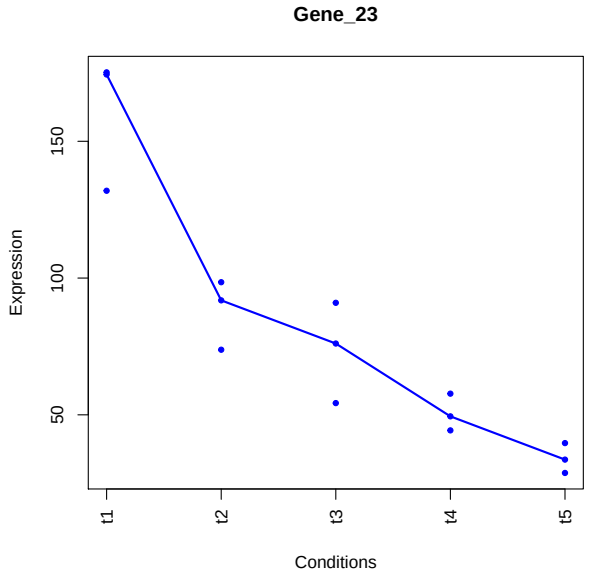

Figure 1: Expression profile of Gene 23

3.1.4 Running EBSeqHMM on gene expression estimates

Function `EBSeqHMMTest` may be used to estimate parameters and the posterior probability (PP) of being in each expression path. For example, here we run five iterations of the Baum-Welch algorithm by setting `UpdateRd=5`. Note that in practice, additional iterations are usually required; and please note this may take several minutes).

```
> EBSeqHMMGeneOut <- EBSeqHMMTest(Data=GeneExampleData, sizeFactors=Sizes, Conditions=Conditions,
+                                 UpdateRd=5)
```

3.1.5 Detection of DE genes and inference of gene's most likely path

Function `GetDECalls` may be used to detect DE genes under a target FDR. DE genes are defined as those showing significant change in at least one condition. Under a target FDR = 0.05, we call genes with PP(remain constant)< 0.05 as DE genes.

```
> GeneDECalls <- GetDECalls(EBSeqHMMGeneOut, FDR=.05)
> head(GeneDECalls)
```

```
        Most_Likely_Path    Max_PP
Gene_40 "Up-Up-Down-Down"   "0.9884"
Gene_1  "Down-Up-Down-Down" "0.7742"
Gene_15 "Down-Down-Up-Up"   "0.5733"
Gene_9  "Down-Down-Up-Up"   "0.5524"
Gene_2  "Up-Up-Down-Down"   "0.5321"
Gene_17 "Down-Down-Up-Up"   "0.5084"
```

```
> str(GeneDECalls)
```

```
 chr [1:53, 1:2] "Up-Up-Down-Down" "Down-Up-Down-Down" "Down-Down-Up-Up" ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:53] "Gene_40" "Gene_1" "Gene_15" "Gene_9" ...
  ..$ : chr [1:2] "Most_Likely_Path" "Max_PP"
```

The output `GeneDECalls` is a matrix with two columns. The first column shows a gene's most likely path (the path with highest PP). And the second column shows PP(most likely path) for each gene. The higher the PP is, the more likely that this gene is following the particular path. Rows are genes that are defined as DE. Here we identified 53 genes as DE under 5% target FDR. To check whether a particular gene is detected and its most likely path, a user may apply the codes below (we use Gene 23 as an example).

```
> "Gene_23"%in%rownames(GeneDECalls)
```

```
[1] TRUE
```

```
> GeneDECalls["Gene_23",]
```

```
     Most_Likely_Path                Max_PP
"Down-Down-Down-Down"                "0.5"
```

Here we can see the most likely path of Gene 23 is Down-Down-Down-Down, and the posterior probability of being in that path is 53.8%.

3.1.6 Clustering DE genes into expression paths

To cluster DE genes into expression paths, we consider DE genes with confident assignments. By default, a gene will be called as a confident assignment to its most likely path if its maximum PP is greater than 0.5. A user may change this threshold by specifying `cutoff`. By default, only dynamic paths are shown (recall that dynamic paths are those for which expression changes over all transitions). A user may specify `OnlyDynamic=FALSE` if sporadic paths are of interest as well.

```
> GeneConfCalls <- GetConfidentCalls(EBSeqHMMGeneOut, FDR=.05,cutoff=.5, OnlyDynamic=TRUE)
> #str(GeneConfCalls$EachPath)
> print(GeneConfCalls$EachPath[1:4])
```

```
$`Up-Up-Up-Up`
         [,1]          [,2]
Gene_37 "Up-Up-Up-Up" "0.5037"

$`Down-Up-Up-Up`
NULL

$`Up-Down-Up-Up`
NULL

$`Down-Down-Up-Up`
         [,1]               [,2]
Gene_15 "Down-Down-Up-Up" "0.5733"
Gene_9  "Down-Down-Up-Up" "0.5524"
Gene_17 "Down-Down-Up-Up" "0.5084"
Gene_35 "Down-Down-Up-Up" "0.5013"
Gene_12 "Down-Down-Up-Up" "0.5001"
Gene_22 "Down-Down-Up-Up" "0.5"
Gene_26 "Down-Down-Up-Up" "0.5"
```

`GeneConfCalls$EachPath` shows DE genes that are classified into each dynamic path. Different clusters are shown in different sub-lists. Columns here are defined similarly as that in section 3.1.5. Genes shown are the DE genes (FDR $\leq$5%) with PP(most likely path) $\geq$ 0.5 for each path. Here we print only 4 paths for demonstration purposes. It shows that only one gene is clustered into each of Up-Up-Up-Up, Down-Up-Up-Up and Up-Down-Up-Up. 7 genes are clustered into the Down-Down-Up-Up path.

If we are interested in the 4th path particularly, we might select the cluster by

```
> Path4 <- GeneConfCalls$EachPath[["Down-Down-Up-Up"]]
> print(Path4)

         [,1]               [,2]
Gene_15 "Down-Down-Up-Up" "0.5733"
Gene_9  "Down-Down-Up-Up" "0.5524"
Gene_17 "Down-Down-Up-Up" "0.5084"
Gene_35 "Down-Down-Up-Up" "0.5013"
Gene_12 "Down-Down-Up-Up" "0.5001"
Gene_22 "Down-Down-Up-Up" "0.5"
Gene_26 "Down-Down-Up-Up" "0.5"
```

The combined list, number of genes in each cluster and list of gene names in each cluster may be obtained by `GeneConfCalls$Overall`, `GeneConfCalls$NumEach` and `GeneConfCalls$EachPathNames`:

```
> head(GeneConfCalls$Overall)

        Most_Likely_Path    Max_PP
Gene_40 "Up-Up-Down-Down"   "0.9884"
Gene_1  "Down-Up-Down-Down" "0.7742"
Gene_15 "Down-Down-Up-Up"   "0.5733"
Gene_9  "Down-Down-Up-Up"   "0.5524"
Gene_2  "Up-Up-Down-Down"   "0.5321"
Gene_17 "Down-Down-Up-Up"   "0.5084"

> print(GeneConfCalls$NumEach)
```

```
      Up-Up-Up-Up       Down-Up-Up-Up       Up-Down-Up-Up     Down-Down-Up-Up
                1                   0                   0                   7
    Up-Up-Down-Up     Down-Up-Down-Up     Up-Down-Down-Up   Down-Down-Down-Up
                1                   0                   0                   0
    Up-Up-Up-Down     Down-Up-Up-Down     Up-Down-Up-Down   Down-Down-Up-Down
                0                   0                   0                   0
  Up-Up-Down-Down   Down-Up-Down-Down   Up-Down-Down-Down Down-Down-Down-Down
                8                   2                   0                   3
```

```
> str(GeneConfCalls$EachPathNames)
```

```
List of 16
 $ Up-Up-Up-Up        : chr "Gene_37"
 $ Down-Up-Up-Up      : NULL
 $ Up-Down-Up-Up      : NULL
 $ Down-Down-Up-Up    : chr [1:7] "Gene_15" "Gene_9" "Gene_17" "Gene_35" ...
 $ Up-Up-Down-Up      : chr "Gene_6"
 $ Down-Up-Down-Up    : NULL
 $ Up-Down-Down-Up    : NULL
 $ Down-Down-Down-Up  : NULL
 $ Up-Up-Up-Down      : NULL
 $ Down-Up-Up-Down    : NULL
 $ Up-Down-Up-Down    : NULL
 $ Down-Down-Up-Down  : NULL
 $ Up-Up-Down-Down    : chr [1:8] "Gene_40" "Gene_2" "Gene_33" "Gene_24" ...
 $ Down-Up-Down-Down  : chr [1:2] "Gene_1" "Gene_18"
 $ Up-Down-Down-Down  : NULL
 $ Down-Down-Down-Down: chr [1:3] "Gene_29" "Gene_7" "Gene_23"
```

If we want to get names of the genes in Down-Down-Down-Down cluster, we may apply:

```
> GeneConfCalls$EachPathNames["Down-Down-Down-Down"]
```

```
$`Down-Down-Down-Down`
[1] "Gene_29" "Gene_7"  "Gene_23"
```

We can see that in addition to Gene 23, there are 2 other genes been clustered into the Down-Down-Down-Down path.

3.2 Isoform level analysis

3.2.1 Required inputs

Data: The object `Data` should be an $I - by - S$ matrix containing the expression values for each isoform and each sample, where I is the number of isoforms and S is the number of samples. As in the gene-level analysis, these values should exhibit estimates of isoform expression, without normalization across samples.

Conditions: The object `Conditions` should be a factor with length S to indicate the condition of each sample. Note the order of levels in the factor should represent the order in the RNA-seq experiment. An example may be found in Section 3.1.1.

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`).

IsoExampleList contains 200 simulated isoforms, in which IsoExampleList$IsoExampleData is a data matrix containing 200 rows of isoforms and 15 columns of samples; IsoExampleList$IsoNames contains the isoform names; IsoExampleList$IsosGeneNames contains the names of the genes to which the isoforms belongs to.

```
> data(IsoExampleList)
> str(IsoExampleList)
```

```
List of 3
 $ IsoExampleData: num [1:200, 1:15] 98 297 336 642 95 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:200] "Iso_1" "Iso_2" "Iso_3" "Iso_4" ...
  .. ..$ : NULL
 $ IsoNames      : chr [1:200] "Iso_1" "Iso_2" "Iso_3" "Iso_4" ...
 $ IsosGeneNames : chr [1:200] "Gene_1" "Gene_2" "Gene_3" "Gene_4" ...
```

```
> IsoExampleData <- IsoExampleList$IsoExampleData
> str(IsoExampleData)
```

```
 num [1:200, 1:15] 98 297 336 642 95 ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:200] "Iso_1" "Iso_2" "Iso_3" "Iso_4" ...
  ..$ : NULL
```

```
> IsoNames <- IsoExampleList$IsoNames
> IsosGeneNames <- IsoExampleList$IsosGeneNames
```

3.2.2 Library size factor

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

```
> IsoSizes <- MedianNorm(IsoExampleData)
```

3.2.3 The I_g vector

While working on isoform level data, EBSeqHMM fits different prior parameters for different uncertainty groups (defined as I_g groups). The default setting to define the uncertainty groups consists of using the number of isoforms the host gene contains (N_g) for each isoform. The default settings will provide three uncertainty groups:

$I_g = 1$ group: Isoforms with $N_g = 1$;
$I_g = 2$ group: Isoforms with $N_g = 2$;
$I_g = 3$ group: Isoforms with $N_g \geq 3$.

The N_g and I_g group assignment can be obtained using the function GetNg. The required inputs of GetNg are the isoform names (IsoformNames) and their corresponding gene names (IsosGeneNames). If RSEM is used for quantification, such information may be found in file sample_name.isoforms.results.

```
> NgList <- GetNg(IsoNames, IsosGeneNames)
> IsoNgTrun <- NgList$IsoformNgTrun
> IsoNgTrun[c(1:3,101:103,161:163)]
```

```
Iso_1   Iso_2   Iso_3 Iso_101 Iso_102 Iso_103 Iso_161 Iso_162 Iso_163
    1       1       1       2       2       2       3       3       3
```

A user may group the isoforms into more than 3 groups by defining the parameter TrunThre in the GetNg function.

3.2.4 Running EBSeqHMM on isoform expression estimates

Here we have 5 time points with triplicates for each. The condition factor may be defined as in section 3.1.1:

```
> CondVector <- rep(paste("t",1:5,sep=""),each=3)
> Conditions <- factor(CondVector, levels=c("t1","t2","t3","t4","t5"))
> str(Conditions)

 Factor w/ 5 levels "t1","t2","t3",..: 1 1 1 2 2 2 3 3 3 4 ...
```

Function `EBSeqHMMTest` may be used to estimate parameters and PP's on isoform level data as well. In isoform level analysis, we need to specify `NgV` to define the uncertainty groups of isoforms. Here we run five iterations of the Baum-Welch algorithm by setting `UpdateRd=5`.

```
> EBSeqHMMIsoOut <- EBSeqHMMTest(Data=IsoExampleData,
+                              NgVector=IsoNgTrun,
+                              sizeFactors=IsoSizes, Conditions=Conditions,
+                              UpdateRd=5)
```

3.2.5 Detection of DE isoforms and inference of isoform's most likely path

Similar to the gene level analyses, Function `GetDECalls` may be used to detect DE isoforms under a target FDR as well. DE isoforms are defined as those showing significant change in at least one condition. Under a target FDR = 0.05, we call isoforms with PP(remain constant)$<$ 0.05 as DE isoforms.

```
> IsoDECalls <- GetDECalls(EBSeqHMMIsoOut, FDR=.05)
> str(IsoDECalls)

 chr [1:106, 1:2] "Up-Up-Down-Down" "Up-Up-Down-Down" "Up-Up-Down-Up" ...
 - attr(*, "dimnames")=List of 2
  ..$ : chr [1:106] "Iso_10" "Iso_20" "Iso_112" "Iso_19" ...
  ..$ : chr [1:2] "Most_Likely_Path" "Max_PP"

> head(IsoDECalls)

        Most_Likely_Path  Max_PP
Iso_10  "Up-Up-Down-Down" "1"
Iso_20  "Up-Up-Down-Down" "1"
Iso_112 "Up-Up-Down-Up"   "1"
Iso_19  "Up-Up-Down-Down" "0.9994"
Iso_23  "Down-Down-Up-Up" "0.9993"
Iso_122 "Down-Down-Up-Up" "0.9945"
```

The output `IsoDECalls` is a matrix with two columns. The first column shows an isoform's most likely path (the path with highest PP). And the second column shows PP(most likely path) for each isoform. Rows are isoforms that are defined as DE. Here we identified 106 isoforms as DE under 5% target FDR.

3.2.6 Clustering DE isoforms into expression paths

Similar to the gene level analyses, function `GetConfidentCalls` may be used to cluster isoforms into clusters with similar expression profiles.

```
> IsoConfCalls <- GetConfidentCalls(EBSeqHMMIsoOut, FDR=.05)
> head(IsoConfCalls$Overall)
```

```
        Most_Likely_Path  Max_PP
Iso_10  "Up-Up-Down-Down" "1"
Iso_20  "Up-Up-Down-Down" "1"
Iso_112 "Up-Up-Down-Up"   "1"
Iso_19  "Up-Up-Down-Down" "0.9994"
Iso_23  "Down-Down-Up-Up" "0.9993"
Iso_122 "Down-Down-Up-Up" "0.9945"
```

```
> str(IsoConfCalls$EachPath[1:4])
```

```
List of 4
 $ Up-Up-Up-Up    : NULL
 $ Down-Up-Up-Up  : NULL
 $ Up-Down-Up-Up  : chr [1:3, 1:2] "Up-Down-Up-Up" "Up-Down-Up-Up" "Up-Down-Up-Up" "0.5387" ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:3] "Iso_30" "Iso_12" "Iso_27"
  .. ..$ : NULL
 $ Down-Down-Up-Up: chr [1:14, 1:2] "Down-Down-Up-Up" "Down-Down-Up-Up" "Down-Down-Up-Up" "Down-Down-Up-
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr [1:14] "Iso_23" "Iso_122" "Iso_8" "Iso_2" ...
  .. ..$ : NULL
```

```
> str(IsoConfCalls$EachPathNames)
```

```
List of 16
 $ Up-Up-Up-Up        : NULL
 $ Down-Up-Up-Up      : NULL
 $ Up-Down-Up-Up      : chr [1:3] "Iso_30" "Iso_12" "Iso_27"
 $ Down-Down-Up-Up    : chr [1:14] "Iso_23" "Iso_122" "Iso_8" "Iso_2" ...
 $ Up-Up-Down-Up      : chr [1:5] "Iso_112" "Iso_39" "Iso_28" "Iso_33" ...
 $ Down-Up-Down-Up    : NULL
 $ Up-Down-Down-Up    : NULL
 $ Down-Down-Down-Up  : NULL
 $ Up-Up-Up-Down      : NULL
 $ Down-Up-Up-Down    : NULL
 $ Up-Down-Up-Down    : chr "Iso_104"
 $ Down-Down-Up-Down  : chr "Iso_173"
 $ Up-Up-Down-Down    : chr [1:20] "Iso_10" "Iso_20" "Iso_19" "Iso_124" ...
 $ Down-Up-Down-Down  : chr [1:6] "Iso_165" "Iso_170" "Iso_7" "Iso_17" ...
 $ Up-Down-Down-Down  : NULL
 $ Down-Down-Down-Down: chr [1:2] "Iso_118" "Iso_174"
```

4 More detailed examples

4.1 Working on a subset of paths

In the clustering step, a user may define a subset of paths of primary interest to simplify the output list of `GetConfidentCalls` function. By doing so, `GetConfidentCalls` function will only identify genes following the selected paths.

To get paths of interest, function `GetAllPaths` may be used to get all possible patterns in the experiment:

```
> AllPaths <- GetAllPaths(EBSeqHMMGeneOut)
> print(AllPaths)
```

```
 [1] "Up-Up-Up-Up"          "Down-Up-Up-Up"        "Up-Down-Up-Up"
 [4] "Down-Down-Up-Up"      "Up-Up-Down-Up"        "Down-Up-Down-Up"
 [7] "Up-Down-Down-Up"      "Down-Down-Down-Up"    "Up-Up-Up-Down"
[10] "Down-Up-Up-Down"      "Up-Down-Up-Down"      "Down-Down-Up-Down"
[13] "Up-Up-Down-Down"      "Down-Up-Down-Down"    "Up-Down-Down-Down"
[16] "Down-Down-Down-Down"
```

The default settings in `GetAllPaths` will return only dynamic paths (16 paths here). To obtain a list for all paths (including constant path and sporadic paths, for a total of 81 paths here), a user may define `OnlyDynamic=FALSE`:

```
> AllPathsWithEE <- GetAllPaths(EBSeqHMMGeneOut, OnlyDynamic=FALSE)
> print(AllPathsWithEE)

 [1] "Up-Up-Up-Up"          "Down-Up-Up-Up"        "Up-Down-Up-Up"
 [4] "Down-Down-Up-Up"      "Up-Up-Down-Up"        "Down-Up-Down-Up"
 [7] "Up-Down-Down-Up"      "Down-Down-Down-Up"    "Up-Up-Up-Down"
[10] "Down-Up-Up-Down"      "Up-Down-Up-Down"      "Down-Down-Up-Down"
[13] "Up-Up-Down-Down"      "Down-Up-Down-Down"    "Up-Down-Down-Down"
[16] "Down-Down-Down-Down" "EE-Up-Up-Up"          "Up-EE-Up-Up"
[19] "EE-EE-Up-Up"          "Down-EE-Up-Up"        "EE-Down-Up-Up"
[22] "Up-Up-EE-Up"          "EE-Up-EE-Up"          "Down-Up-EE-Up"
[25] "Up-EE-EE-Up"          "EE-EE-EE-Up"          "Down-EE-EE-Up"
[28] "Up-Down-EE-Up"        "EE-Down-EE-Up"        "Down-Down-EE-Up"
[31] "EE-Up-Down-Up"        "Up-EE-Down-Up"        "EE-EE-Down-Up"
[34] "Down-EE-Down-Up"      "EE-Down-Down-Up"      "Up-Up-Up-EE"
[37] "EE-Up-Up-EE"          "Down-Up-Up-EE"        "Up-EE-Up-EE"
[40] "EE-EE-Up-EE"          "Down-EE-Up-EE"        "Up-Down-Up-EE"
[43] "EE-Down-Up-EE"        "Down-Down-Up-EE"      "Up-Up-EE-EE"
[46] "EE-Up-EE-EE"          "Down-Up-EE-EE"        "Up-EE-EE-EE"
[49] "EE-EE-EE-EE"          "Down-EE-EE-EE"        "Up-Down-EE-EE"
[52] "EE-Down-EE-EE"        "Down-Down-EE-EE"      "Up-Up-Down-EE"
[55] "EE-Up-Down-EE"        "Down-Up-Down-EE"      "Up-EE-Down-EE"
[58] "EE-EE-Down-EE"        "Down-EE-Down-EE"      "Up-Down-Down-EE"
[61] "EE-Down-Down-EE"      "Down-Down-Down-EE"    "EE-Up-Up-Down"
[64] "Up-EE-Up-Down"        "EE-EE-Up-Down"        "Down-EE-Up-Down"
[67] "EE-Down-Up-Down"      "Up-Up-EE-Down"        "EE-Up-EE-Down"
[70] "Down-Up-EE-Down"      "Up-EE-EE-Down"        "EE-EE-EE-Down"
[73] "Down-EE-EE-Down"      "Up-Down-EE-Down"      "EE-Down-EE-Down"
[76] "Down-Down-EE-Down"    "EE-Up-Down-Down"      "Up-EE-Down-Down"
[79] "EE-EE-Down-Down"      "Down-EE-Down-Down"    "EE-Down-Down-Down"
```

Assume we are only interested in the monotone increasing path and monotone decreasing path (1st and 16th in the list of dynamic paths), we may define `Paths=AllPaths[c(1,16)]` in `GetConfidentCalls` function to obtain the simplified list:

```
> GeneConfCallsTwoPaths <- GetConfidentCalls(EBSeqHMMGeneOut, FDR=.05, Paths=AllPaths[c(1,16)])
> print(GeneConfCallsTwoPaths)

$Overall
        Most_Likely_Path      Max_PP
Gene_37 "Up-Up-Up-Up"         "0.5037"
Gene_29 "Down-Down-Down-Down" "0.5021"
```

```
Gene_7  "Down-Down-Down-Down" "0.5"
Gene_23 "Down-Down-Down-Down" "0.5"

$EachPath
$EachPath$`Up-Up-Up-Up`
        [,1]          [,2]
Gene_37 "Up-Up-Up-Up" "0.5037"

$EachPath$`Down-Down-Down-Down`
        [,1]                  [,2]
Gene_29 "Down-Down-Down-Down" "0.5021"
Gene_7  "Down-Down-Down-Down" "0.5"
Gene_23 "Down-Down-Down-Down" "0.5"


$NumEach
        Up-Up-Up-Up Down-Down-Down-Down
                   1                   3

$EachPathNames
$EachPathNames$`Up-Up-Up-Up`
[1] "Gene_37"

$EachPathNames$`Down-Down-Down-Down`
[1] "Gene_29" "Gene_7"  "Gene_23"
```

The output object only contains information of clusters associated with these two paths. We have 1 and 4 genes clustered into the monotone increasing and decreasing cluster, respectively.

4.2 Data visualization

As introduced in section 3.1.3, EBSeqHMM also provides functions to visualize genes/isoforms of interest. Prior to generating the plots, we first need to obtain a normalized expression matrix that adjusts for library sizes. The normalized matrix may be obtained by the `GetNormalizedMat` function:

```
> GeneNormData <- GetNormalizedMat(GeneExampleData, Sizes)
```

Then function `PlotExp` may be used to visualize gene/isoform expression patterns across ordered conditions. For example, suppose we are interested in genes following the same path as Gene 23 (Down-Down-Down-Down) in our simulated data. To get genes that were clustered into this path, we may use the codes below:

```
> print(GeneConfCallsTwoPaths$EachPath[["Down-Down-Down-Down"]])

        [,1]                  [,2]
Gene_29 "Down-Down-Down-Down" "0.5021"
Gene_7  "Down-Down-Down-Down" "0.5"
Gene_23 "Down-Down-Down-Down" "0.5"

> GeneOfInterest <- GeneConfCallsTwoPaths$EachPathNames[["Down-Down-Down-Down"]]
> print(GeneOfInterest)

[1] "Gene_29" "Gene_7"  "Gene_23"
```

```
> par(mfrow=c(2,2))
> for(i in 1:3)PlotExp(GeneNormData, Conditions, Name=GeneOfInterest[i])
```

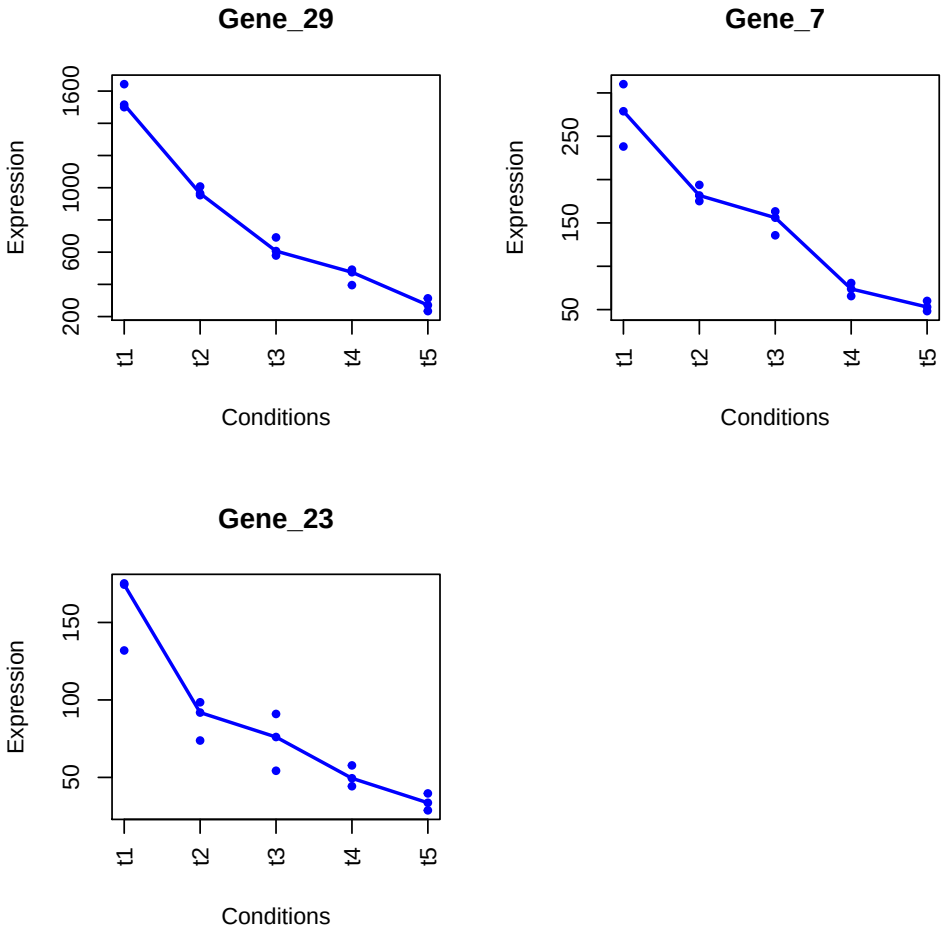

Figure 2: Genes in monotone decreasing cluster

4.3 Diagnostic plots

As in EBSeq, EBSeqHMM also relies on parametric assumptions that should be checked following analysis. Two functions from the EBSeq package may be used to check the fit of the EBSeqHMM model. The `QQP` function generates Q-Q plot of the empirical q's vs. simulated q's from the Beta prior distribution with estimated hyper-parameters. Figure 3 shows an example on gene level data. Note in Figure 3, only a small set of genes are considered here for demonstration purposes. Much better fitting should be expected while using the full set of genes. For examples of reasonable diagnostics, see Figure 7.

```
> par(mfrow=c(3,2))
> QQP(EBSeqHMMGeneOut,GeneLevel=TRUE)
```

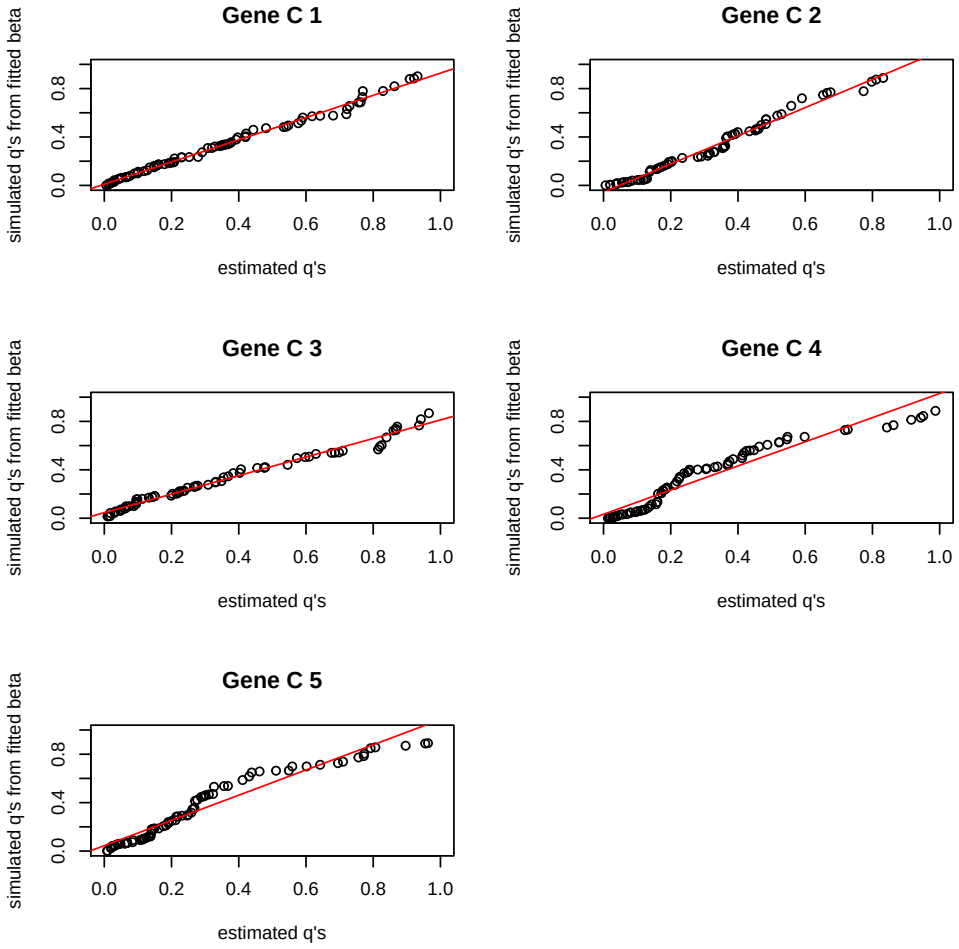

Figure 3: Q-Q plots for checking the assumption of a Beta prior within each condition (on gene level data). Note only a small set of genes are considered here for demonstration purposes. Much better fitting should be expected while using the full set of genes. For examples of reasonable diagnostics, see Figure 7.

Likewise, the `DenNHist` function may be used to check the density plot of empirical q's vs. simulated q's from the fitted Beta prior. Figure 4 shows an example on gene level data.

```
> par(mfrow=c(3,2))
> DenNHist(EBSeqHMMGeneOut, GeneLevel=TRUE)
```

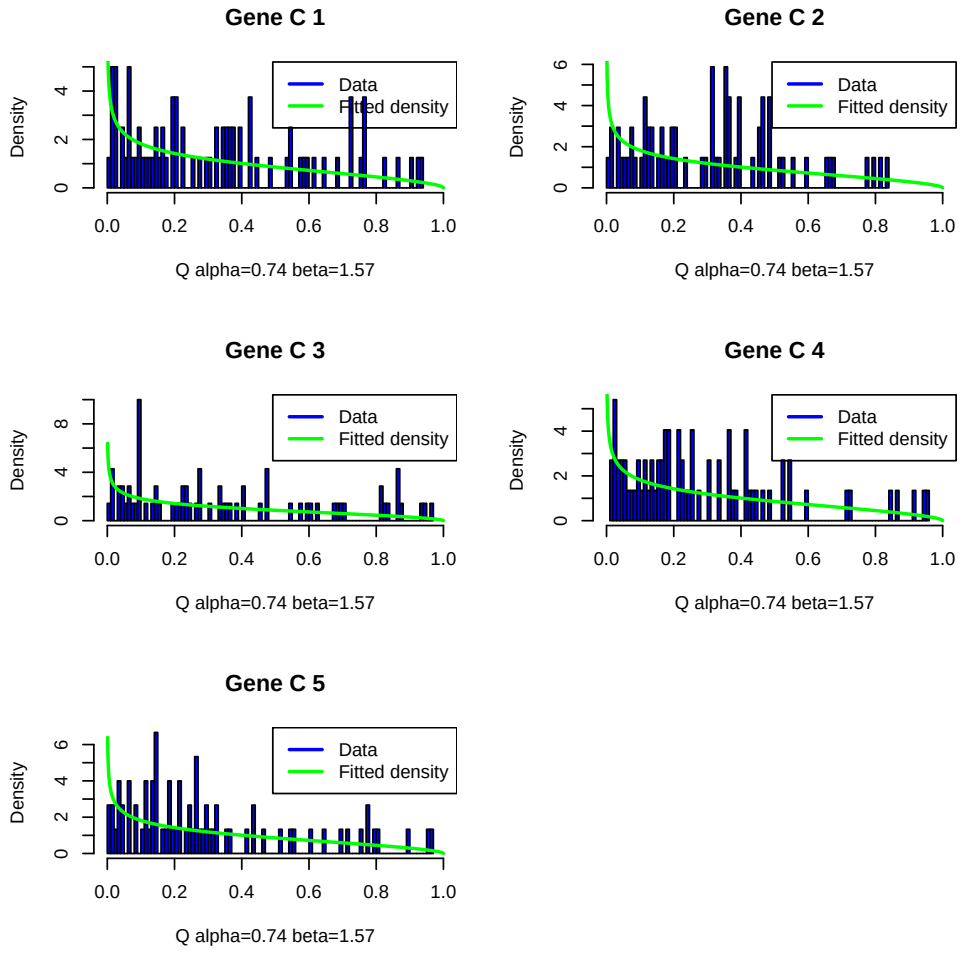

Figure 4: Density plots and histograms for checking the assumption of a Beta prior within each condition (on gene level data). Note only a small set of genes are considered here for demonstration purposes. Much better fitting should be expected while using the full set of genes. For examples of reasonable diagnostics, see Figure 7.

Figure 5 and 6 show similar plots on isoform level analysis.

```
> par(mfrow=c(4,4))
> QQP(EBSeqHMMIsoOut)
```

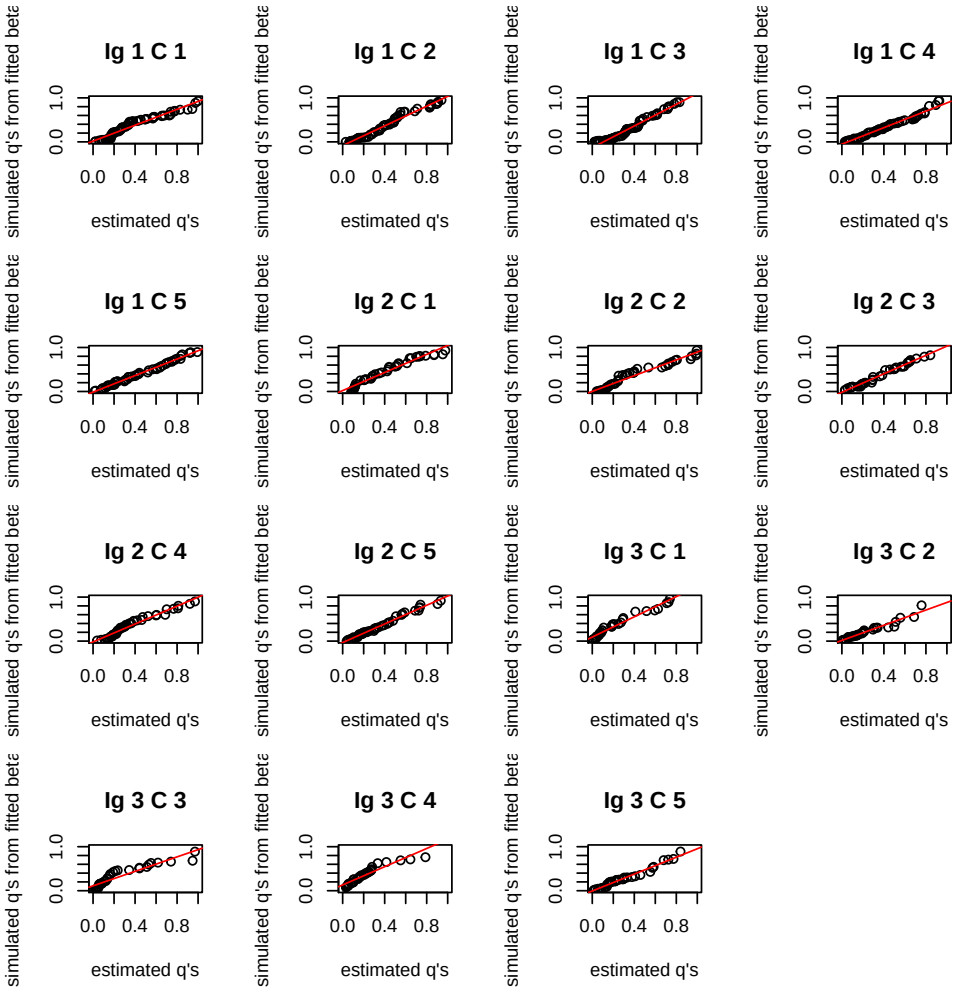

Figure 5: Q-Q plots for checking the assumption of a Beta prior within each condition (on isoform level data). Note only a small set of isoforms are considered here for demonstration purposes. Much better fitting should be expected while using the full set of isoforms. For examples of reasonable diagnostics, see Figure 7

```
> par(mfrow=c(4,4))
> DenNHist(EBSeqHMMIsoOut)
```

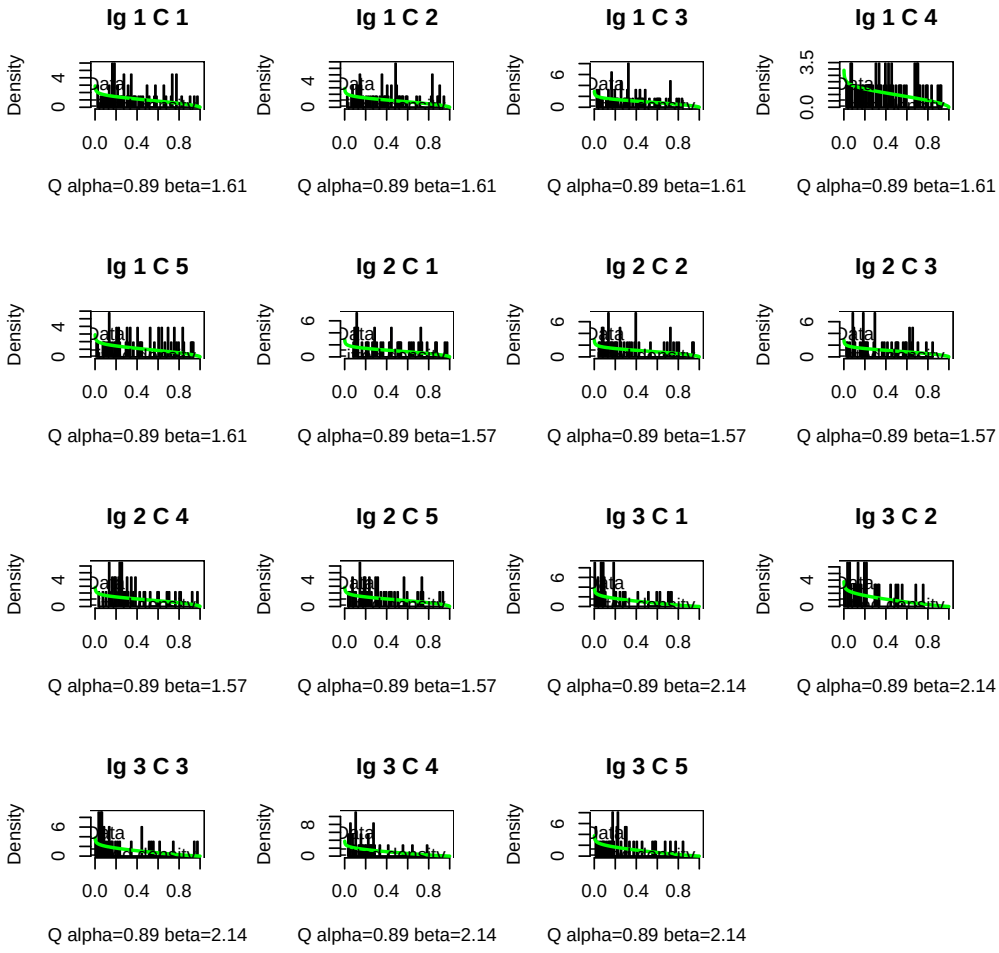

Figure 6: Density plots and histograms for checking the assumption of a Beta prior within each condition (on isoform level data). Note only a small set of isoforms are considered here for demonstration purposes. Much better fitting should be expected while using the full set of isoforms. For examples of reasonable diagnostics, see Figure 7

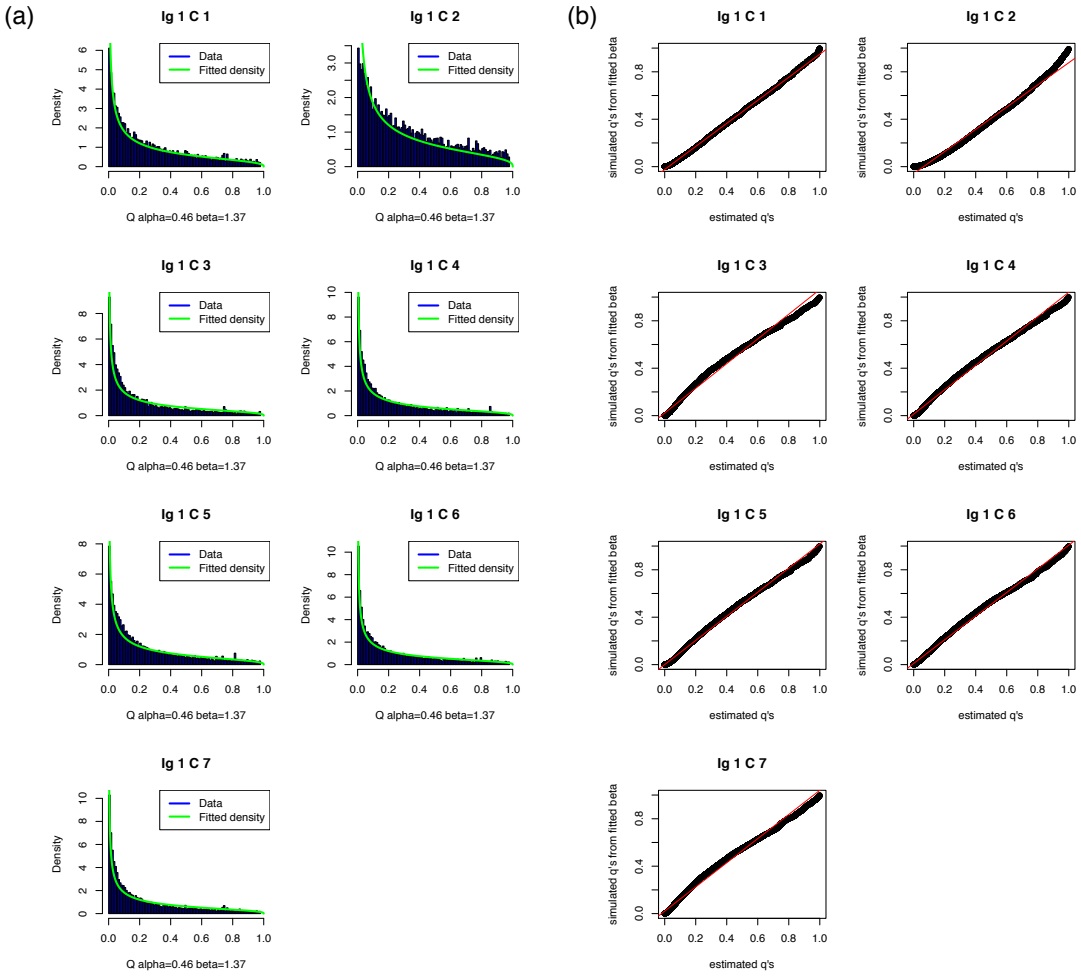

Figure 7: Shown are model diagnostic plots on a case study data set (a) Shown are the histograms of empirical q's estimated within each condition and the fitted Beta densities using that same data. (b) Shown are estimated q's and the same number of points simulated from the Beta prior.

4.4 Using mappability ambiguity clusters instead of the I_g vector when the gene-isoform relationship is unknown

When working with a de-novo assembled transcriptome, in which case the gene-isoform relationship is unknown, a user can use read mapping ambiguity cluster information instead of Ig, as provided by RSEM (Li and Dewey, 2011) in the output file `output_name.ngvec`. The file contains a vector with the same length as the total number of transcripts. Each transcript has been assigned to one of 3 levels (1, 2, or 3) to indicate the mapping uncertainty level of that transcript. The mapping ambiguity clusters are partitioned via a k-means algorithm on the unmappability scores that are provided by RSEM. A user can read in the mapping ambiguity cluster information using:

```
> IsoNgTrun <- scan(file="output_name.ngvec", what=0, sep="\n")
```

More details on using the RSEM-EBSeq pipeline on de novo assembled transcriptomes can be found at `http://deweylab.biostat.wisc.edu/rsem/README.html#de`.

Other unmappability scores and other cluster methods (e.g. Gaussian Mixture Model) could also be used to form the uncertainty clusters. Also, the number of uncertainty groups is not limited to 3. A user may consider grouping isoforms into more than 3 groups if needed.

More details can be found in the EBSeq vignette:
`http://www.bioconductor.org/packages/devel/bioc/vignettes/EBSeq/inst/doc/EBSeq_Vignette.pdf`

References

Anders, S. and Huber, W. (2010). Differential expression analysis for sequence count data. *Genome Biology*, **11**, R106.

Bullard, J. H., Purdom, E. A., Hansen, K. D., and Dudoit, S. (2010). Evaluation of statistical methods for normalization and differential expression in mrna-seq experiments. *BMC Bioinformatics*, **11**, 94.

Leng, N., Li, Y., Mcintosh, B. E., Nguyen, B. K., Duffin, B., Tian, S., Thomson, J. A., Colin, D., Stewart, R. M., and Kendziorski, C. (2014). Ebseq-hmm: A bayesian approach for identifying gene-expression changes in ordered rna-seq experiments.

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

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