Package ‘FNBSeq’

April 6, 2016

Version 0.1.0

Title Analysis of RNA-Seq data Using a Family of Negative Binomial Models

Author Lili Zhao <zhaolili@umich.edu> and Dai Feng <dai_feng@merck.com>

Maintainer Lili Zhao <zhaolili@umich.edu>

Imports robCompositions (>= 1.9.1), IRanges (>= 1.22.10)

Depends DEXSeq (>= 1.10.8), parallel (>= 3.2.2), doParallel (>= 1.0.10), foreach (>= 1.4.3)

Suggests pasilla (>= 0.2.22)

Description This package performs differenital analysis in RNA-Seq data using a family of negative binomial models. The differenital analysis includes testing for the expression of genes, exons and transcripts and the relative usage of exons and transcripts. It calculates posterior probabilities of the differential expression via Gibbs sampling with fully tractable closed-forms.

License GPL (>= 3)

NeedsCompilation yes

R topics documented:

FNBSeq *Exon/transcript analysis for RNA-Seq data*

Description

This function tests differential exon/transcript expression and relative usage between two conditions. It models transcript/exon counts within a particular gene using a family of negative binomial models (FNB).

Usage

```
FNBSeq(dd, condition, model=c("transcript", "exon"),
       nsim=10000, burn=2000, thin=1, seed=100,
       a0 = 1, b0 = 1, f0 = 0.1, e0 =0.1, e0.i=0.1,
       ncore=detectCores(), ncluster=NULL)
```

Arguments

`dd`	Input data (the data format should be the same as the output file from the ProcessDEXSeq function. This input data can be obtained from the ProcessDEXSeq function.
`condition`	A factor of experimental conditions (or treatments). The length of the factor has to be equal to the number of samples, assigning a condition to each sample.
`model`	"transcript" or "exon" analysis.
`nsim`	The number of MCMC iterations. The default value is 10000.
`burn`	The number of burn-in. The default is 2000.
`thin`	It specifies the intervals at which the Markov chain is stored. The detault is thin =1 meaning we stored every iteration.
`seed`	Random seed (default is 100).
`a0,b0`	The beta prior Beta(a0, b0) for the probability parameter, p. Defaut is Beta(1,1).
`e0,f0`	The gamam prior Gamma(e0,f0) for the gene-level dispersion parameter. Default is Gamma(0.1,0.1).
`e0.i`	The dirichlet prior Dirichlet(e0.i,...,e0.i,...e0.i) for the relative usage. Default is e0.i=0.1.
`ncore`	The number of cores used for paralell computation. The default is using the maximum number of cores available in a processor.
`ncluster`	The number clusters used for paralell computation. The default is `NULL`.

Details

Given the nomalized counts on exons (transcripts), a family of negative binomial model (FNB) is used to estimate the expression of exons (transcripts) and their relative uage (Zhao et al., 2016).

In the FNB model, read counts in exons (transcripts) are modelled by a negative binomial (NB) distribution. The NB distribution is parametrized using a probability parameter (0<=p<=1) and a dispersion (r>0) parameter. It has a mean mu=r*p/(1-p) and a variance mu+r^-1 mu^2. The NB distributions for exon (transcript) counts share the same parameter p but each has its own dispersion parameter. Under this formulation, the total read count in a gene also has a NB with the same p and a dispersion parameter, r, which is the sum of the dispersion parameters over the exons(transcripts). The r is assumed to be the same between conditions.

Parameters in the FNB model are estimated using Gibbs sampling with conjugate forms. Posterior probabilities of the hypothsis are calcuated and converted to Bayesian FDRs (Lewin et al.,2007 and Luis et al., 2013). In the transcript analysis, transcript counts are imputed using the gamma-poisson mixture formulation of a NB distribution and are embeded in the MCMC sampling. Estimates (expression and relative usage) are adjusted by the effective length of transcripts. The gene expression is the sum of the transcript expressions.

Value

In the transcript analysis, there are two output files, one is on the gene levle (named as “Results.Gene”) and the other is on the transcript level (named as “Results.Transcript”). Both files are a data frame. The output file on the gene level has the following columns:

`Gene`	Gene names.
`Exp`	Mean expression values for two conditions.

`mug.Prob`	Posterior probability of the expression in the second condition is larger than the first condtion.
`Adist.Median`	Posterior median of the Aitchison distance (Aitchison,2000), a larger value indicates a larger difference in the overall relative usage between two conditons.
`Qg_fdr2_1`	FDRs for testing if the gene experession in the second condition is larger than the first condition.
`Qg_fdr1_2`	FDRs for testing if the gene experession in the first condition is larger than the second condition.

The output for transcripts has the following columns:

`Gene`	Gene names.
`Transcript`	Transcript names.
`mut1.Mean`	Mean expression values for transcripts in the first condtion.
`mut2.Mean`	Mean expression values for transcripts in the second condtion.
`mut.Prob`	Posterior probability of the transcript expression in the second condition is larger than the first condtion.
`Qt1.Mean`	Mean relative usage for transcripts in the first condtion.
`Qt2.Mean`	Mean relative usage for transcripts in the second condtion.
`Qt.Prob`	Posterior probability of the transcript relative usage in the second condition is larger than the first condtion.
`mut_fdr2_1`	FDRs for testing if the transcript expression in the second condition is larger than the first condition.
`mut_fdr1_2`	FDRs for testing if the transcript expression in the first condition is larger than the second condition.
`Qt_fdr2_1`	FDRs for testing if the transcript relative usage in the second condition is larger than the first condition.
`Qt_fdr1_2`	FDRs for testing if the transcript relative usage in the first condition is larger than the second condition.

In the exon analysis, there are two output files, one is on the gene levle and one is on the exon level. Both files are a data frame. The output for genes has the following columns:

`Gene`	Gene names.
`Exp`	Mean expression values for two conditions.
`mug.Prob`	Posterior probability of the expression in the second condition is larger than the first condtion.
`Adist.Median`	Posterior median of the Aitchison distance (Aitchison:2000), a larger value indicates a larger difference in the relative usage between two conditons.
`Qg_fdr2_1`	FDRs for testing if the gene experession in the second condition is larger than the first condition.
`Qg_fdr1_2`	FDRs for testing if the gene experession in the first condition is larger than the second condition.

The output for exons has the following columns:

`Gene`	Gene names.
`Exon`	Exon names.

`mue1.Mean`	Mean expression values for exons in the first condtion.
`mue2.Mean`	Mean expression values for exons in the second condtion.
`mue.Prob`	Posterior probability of the exon expression in the second condition is larger than the first condtion.
`Qe1.Mean`	Mean relative usage for exonss in the first condtion.
`Qe2.Mean`	Mean relative usage for exonss in the second condtion.
`Qe.Prob`	Posterior probability of the exon relative usage in the second condition is larger than the first condtion.
`mue_fdr2_1`	FDRs for testing if the exon expression in the second condition is larger than the first condition.
`mue_fdr1_2`	FDRs for testing if the exon expression in the first condition is larger than the second condition.
`Qe_fdr2_1`	FDRs for testing if the exon relative usage in the second condition is larger than the first condition.
`Qe_fdr1_2`	FDRs for testing if the exon relative usage in the first condition is larger than the second condition.

Author(s)

Lili Zhao `<zhaolili@umich.edu>`, Dai Feng `<dai_feng@merck.com>`

References

Anders, S., Reyes, A., and Huber, W. (2012). Detecting differential usage of exons from RNA-seq data. Genome Research, 22, 2008-2017.

Aitchison, J., Barcelo-Vidal, C., Martin-Fernandez, J. A., and Pawlowsky-Glahn, V.(2000). Logratio analysis and compositional distance. Mathematical Geology, 32,271-275.

Lewin, A., Bochkina, N., and Richardson, S. (2007). Fully bayesian mixture model for differential gene expression: simulations and model checks. Statistical Applications in Genetics and Molecular Biologys, 6, Article36.

Len-Novelo, L. G., Mller, P., Arap,W., Kolonin, M., Sun, J., Pasqualini, R., and Do, K. A. (2013). Semi-parametric bayesian inference for phage display data. Biometrics, 69, 174-183.

Zhao, L., Wu, W., Feng, D., Jiang, H. and Nguyen X.(2016). Analysis of RNA-Seq Data Using a Family of Negative Binomial Models. Submitted to Biometrics

Examples

```
## Not run:

   library(doParallel)
   library(foreach)
   library(FNBSeq)


   #--------------------------------------------------------------------------------
   #obtain a small subset of the pasilla data (named as dxd in DEXSeq)
   #--------------------------------------------------------------------------------
    data(pasillaDEXSeqDataSet, package="pasilla")

    #Obtain a normalized count matrix
```

```
dxd <- estimateSizeFactors(dxd)
count.matrix.norm.all <- round(featureCounts(dxd, normalized=TRUE))

condition <- c("A", "A", "A", "B", "B","B","B")
read.size <- 37

#data processing for the exon analysis
dd=ProcessDEXSeq(count.matrix.norm.all, dxd, model="exon", trimReads=5)

# exon analysis
results <- FNBSeq(dd, condition, model="exon",
                  nsim=10000, burn=2000, thin=1, seed=100,
                  a0 = 1, b0 = 1, f0=0.1, e0 =0.1, e0.i=0.1, ncore=4)

# data processing for the transcript analysis
dd=ProcessDEXSeq(count.matrix.norm.all, dxd, model="transcript",
                 trimReads=0, read.size=37)

# transcript analysis
results <- FNBSeq(dd, condition, model="transcript",
                  nsim=10000, burn=2000, thin=1, seed=100,
                  a0 = 1, b0 = 1, f0=0.1, e0 =0.1, e0.i=0.1, ncore=4)


#-----------------------------------------------------------------------------
#obtain all the asilla data in DEXSeq, as described in Zhao et al (2016)
#this can take a long time
#-----------------------------------------------------------------------------
inDir = system.file("extdata", package="pasilla", mustWork=TRUE)
dir(inDir)

# As in DEXSeq
annotationfile = file.path(inDir, "Dmel.BDGP5.25.62.DEXSeq.chr.gff")

sampleTable = data.frame(row.names = c( "treated1fb", "treated2fb",
                                        "treated3fb", "untreated1fb",
                                        "untreated2fb", "untreated3fb",
                                        "untreated4fb" ),
                         condition = c("A", "A", "A", "B", "B","B","B"),
                         type = c( "single-end", "paired-end", "paired-end",
                                    "single-end","single-end", "paired-end",
                                    "paired-end"))

dxd = DEXSeqDataSetFromHTSeq(
        countfiles = file.path(inDir, paste(rownames(sampleTable), "txt", sep=".")),
        sampleData=sampleTable, design= ~ sample + exon + condition:exon,
        flattenedfile = annotationfile)

# obtained a normalized count matrix
dxd <- estimateSizeFactors(dxd)
count.matrix.norm.all <- round(featureCounts(dxd, normalized=TRUE))

condition <- c("A", "A", "A", "B", "B", "B", "B")

# data processing for the exon analysis
dd=ProcessDEXSeq(count.matrix.norm.all, dxd, model="exon", trimReads=5)
```

```
    # exon analysis
    results <- FNBSeq(dd, condition, model="exon",
                      nsim=10000, burn=2000, thin=1, seed=100,
                      a0 = 1, b0 = 1, f0=0.1, e0 =0.1, e0.i=0.1, ncore=4)

    # data processing for the transcript analysis
    dd=ProcessDEXSeq(count.matrix.norm.all, dxd, model="transcript",
                     trimReads=0, read.size=37)

    # transcript analysis
    results <- FNBSeq(dd, condition, model="transcript",
                      nsim=10000, burn=2000, thin=1, seed=100,
                      a0 = 1, b0 = 1, f0=0.1, e0 =0.1, e0.i=0.1, ncore=4)


## End(Not run)
```

`ProcessDEXSeq` *Process the data to be used in FNBSeq function. This function relies on the DEXSeq package (Anders et al., 2012)*

Description

This function takes the data object from DEXSeq and processes the data for the FNBSeq function

Usage

```
ProcessDEXSeq(count.matrix.norm.all, dxd,
                model=c("transcript", "exon"),
                trimReads=0, read.size=0)
```

Arguments

`count.matrix.norm.all`	The normalized count matrix. The rows represent exons and the columns represent samples.
`dxd`	An object from DEXSeq.
`model`	Process the data for the "transcript" or "exon" analysis.
`trimReads`	Ananalysis only includes exons with read counts > trimReads. The default is trimReads=0.
`read.size`	The read length in the RNA-seq sequencing. It is used in the transcript analysis to obtain transcript lengths (defualt is 0).

Details

Data produced from this function is the input file for the FNBSeq function

Value

The output file is a list (the length of the list is equal to the number of genes). Each gene is also a list, when model="exon", each gene list contains

`gene.name`	A gene name.
`Exon.name`	exon names.
`X`	A count matrix with rows corresponding to exons (or exon counting bins) and columns corresponding to samples.
`n.Exon`	Number of exons per gene.
`ELen`	A vector of exon lengths.

when model="transcript", each gene list contains

`gene.name`	A gene name.
`transcript.name`	Transcript names.
`X`	A count matrix with rows corresponding to exons (or exon counting bins) and columns corresponding to samples.
`n.T`	Number of transcripts per gene.
`n.E`	Number of exons per gene.
`ELen`	A vector of exon lengths.
`TLen`	A vector of transcript lengths.
`M`	The n.E * n.T matrix with 0/1 indicating if the exon belongs to the transcript.

Author(s)

Lili Zhao <zhaolili@umich.edu>, Dai Feng <dai_feng@merck.com>

References

Anders, S., Reyes, A., and Huber, W. (2012). Detecting differential usage of exons from RNA-seq data. Genome Research, 22, 2008-2017

Examples

```
#obtain a small subset of the pasilla data (named as dxd in DEXSeq)
data(pasillaDEXSeqDataSet, package="pasilla")

# obtained a normalized count matrix
dxd <- estimateSizeFactors(dxd)
count.matrix.norm.all <- round(featureCounts(dxd, normalized=TRUE))

condition <- c("A", "A", "A", "B", "B","B","B")

# data processing for the exon analysis
dd=ProcessDEXSeq(count.matrix.norm.all, dxd, model="exon", trimReads=5)

# data processing for the transcript analysis
dd=ProcessDEXSeq(count.matrix.norm.all, dxd, model="transcript",
                 trimReads=0, read.size=37)
```

Index