Package 'BBSeq'

September 15, 2011

Type Package

Title Beta-Binomial modeling of the overdispersion of the RNA-seq count data

Version 1.0

Date 2011-03-08

Author Yihui Zhou, Fred A. Wright

Maintainer Yihui Zhou <yzhou@bios.unc.edu>

Description We describe BBSeq, which incorporates two approaches: (i) a simple beta-binomial generalized linear model approach, which has not been extensively tested for RNA-Seq data, and (ii) an extension of an expression mean-variance modeling approach to RNA-Seq data, involving modeling of the overdispersion as a function of the mean.

License GPL-2

LazyLoad yes

R topics documented:

`BBSeq-package` *Beta-Binomial modeling of the overdispersion of the RNA-seq count data*

Description

This package is used to identify differential expression in high-throughput count data, such as RNA-seq count data which is derived from next-generation sequencing machines. Our modeling design is very flexible. It can not only solve the data with multiple comparisons, but also can find the affect from other covariates, such as age and other counfounding variables.

Details

Package:	anRpackage
Type:	Package
Version:	1.0
Date:	2011-02-18
License:	GPL-2
LazyLoad:	yes

See the vignette for more details

Author(s)

Yihui Zhou, Fred A. Wright

Maintainer: Yihui Zhou <yzhou@bios.unc.edu>

References

Yihui Zhou, Fred Wright, "A powerful and flexible method for RNA-Seq data analysis", 2011, In preparation for submission.

`Beta.free` *Beta estimation by FREE approach*

Description

This data was generated by free.estimate function based on the dataset data.Y. Gender is the parameter we are interested in. Male is the reference group.

Usage

```
data(Beta.free)
```

Format

A data frame with 5000 observations on the following 2 variables.

`V1` The estimation of the gene expression level for male

`V2` The estimation of the gender affection on the gene expression level.

`constrained` *Get the negative log-likelihood in Constrained Approach*

Description

We use the mean-overdisperson relation to get the gene specific negative log-likelihood.

Usage

```
constrained(para, X, Y.col, coeff, Y.c, p)
```

Arguments

`para`	A vector constains the gene specific initial value of beta estimation.
`X`	Design matrix.
`Y.col`	A vector constains the counts of a specified gene.
`coeff`	The coefficient of the fitted mean-overdispersion relation.
`Y.c`	The library size.
`p`	The number of parameters we are interested in.

Value

return the nagative log-likelihood of a specified gene.

Author(s)

Yihui Zhou, Fred A. Wright

Examples

```
###  data.Y is 5000 genes randomly selected from Montgomery data
data(data.Y)
###  Beta.free is 5000 x 2 matrix, it is the beta estimation by FREE approach
data(Beta.free)
X=cbind(1,c(rep(1,3),rep(0,3),rep(1,3),rep(0,3)))
coeff=c(-1.065,1.675,0.073,0.001)
### we use constrained to find the negative -loglikehood of the first gene
neglogl=constrained(as.numeric(Beta.free[1,]),X,as.numeric(data.Y[1,]),coeff,lib.size(dat
```

`constrained.estimate` *The main function of Constrained approach*

Description

Given the beta estimation from FREE approach, we use this function to estimate beta using mean-overdispersion modeling.

Usage

```
constrained.estimate(data.matrix, X, gn, Beta.free, psi.free)
```

Arguments

`data.matrix`	The matrix format of RNA-seq count data with size m by n. m corresponds to gene and n correponds to sample.
`X`	Design matrix.
`gn`	we use a polynomial with degree of freedom gn to fit the mean-overdispersion relation.
`Beta.free`	Beta etimation matrix from FREE approach. This is a p by m matrix, where p is the number of the parameters we are interested. The output from FREE approach was m by p matrix, and we need to transpose it first before inputing constrained.estimate function.
`psi.free`	psi is the log of dispersion parameter. psi.free is the estimation of psi from FREE approach.

Details

Constrained.estimate function is different from free.estimate since this one makes use of mean-overdispersion modeling to estimate psi inside of the function rather than using the optimation algorithm in FREE approach. This function provides beta estimation, the variance of beta estimation and the corresponding p-values.

Value

`betahat.model`	Estimation of beta, m by p matrix
`bvar.model`	Estimation of the variance of betahat,m by p
`p.model`	In the two groups comparison, p.model is the p value corresponding to the group estimation

Author(s)

Yihui Zhou, Fred A. Wright

Examples

```
###  data.Y is 5000 genes randomly selected from Montgomery data
data(data.Y)
###  Beta.free is 5000 x 2 matrix, it is the beta estimation by FREE approach
data(Beta.free)
###  psi.free is a vector with length 5000, it is the psi estimation
###  by FREE approach
data(psi.free)
set.seed(1)
index=sample(1:5000,20)
data.sample=data.Y[index,]
X= cbind(1,rep(rep(c(1,0),each=3),2))
out.model=constrained.estimate(data.sample,X,gn=3,Beta.free,psi.free)
```

---

data.anno *The annotation file for dataset data.Y*

Description

This file contains refseq Id, chromosome, gene information for the corresponding records in data.Y.

Usage

```
data(data.anno)
```

Format

A data frame with 5000 observations on the following 3 variables.

`ref.id` refseq ID for each transcript

`chr` Chromosome information for each transcript

`gene` Gene information for each transcript

Examples

```
data(data.anno)
data.anno[1:5,]   ## shows the annotation infromation for the first five genes
```

---

data.Y *the sample data of this BBSeq package*

Description

data.Y was generated by randomly selecting 5000 genes from the real data (Montgomery et al. 2010).

Usage

```
data(data.Y)
```

Format

A data frame with 5000 observations on the following 12 variables.

`female.1` a numeric vector

`female.2` a numeric vector

`female.3` a numeric vector

`male.1` a numeric vector

`male.2` a numeric vector

`male.3` a numeric vector

`female.4` a numeric vector

`female.5` a numeric vector

`female.6` a numeric vector

`male.4` a numeric vector

`male.5` a numeric vector

`male.6` a numeric vector

Details

We selected 12 samples(balanced gender) out of 60 from the original dataset, 6 females and 6 males totally. With the gender information, it is convienent for us to do the sample analysis on the two groups comparison.

Examples

```
data(data.Y)
data.Y[1:5,]          ## shows the counts of the first 5 genes
```

`free.estimate` *The main function for FREE approach*

Description

We use beta-binomial to handle the overdispersion of the count data. beta estimation from linear regression is used as initial value in the optimization procedure.

Usage

```
free.estimate(data.matrix, X)
```

Arguments

`data.matrix`	the count matrix (m by n) we are going to analysis
`X`	design matrix

Value

`betahat.free`	Estimation of beta, m by p matrix
`bvar.free`	Estimation of the variance of betahat,m by p
`p.free`	In the two groups comparison, p.free is the p value corresponding to the group estimation
`psi.free`	the estimation of the logit of the overdispersion parameter

Author(s)

Yihui Zhou, Fred A. Wright

Examples

```
###  data.Y is 5000 genes randomly selected from Montgomery data
data(data.Y)
set.seed(1)
index=sample(1:5000,20)
data.sample=data.Y[index,]
X=cbind(1,c(rep(1,3),rep(0,3),rep(1,3),rep(0,3)))
out.free=free.estimate(data.sample,X)
```

`free.mle` *Get negative loglikelihood values*

Description

This function gets all the negative loglikelihood values from myoptim.free function according to different design matrices we use.

Usage

```
free.mle(Y.count,Y.size,predictor)
```

Arguments

`Y.count`	the gene-specific count data.
`Y.size`	the library size of the samples
`predictor`	a matrix which gives all the detailed information for each parameter of interests. Each parameter of interests takes a column; for the discrete variable, the column is a vector indicating the level of each sample; for the continuous variable, it is a vector of 1 with length n (sample size).

Value

a vector of negative loglikelihood values according to all different design matrices we have from X.builder

Author(s)

Yihui Zhou, Fred A. Wright

Examples

```
data(data.Y)
data.Y=as.matrix(data.Y)
set.seed(5)                    ### We simulate score as a continuous covariate
score=rnorm(12,mean=0,sd=1)
categorical=c(1,0)
c1=as.matrix(c(rep(1,3),rep(0,3),rep(1,3),rep(0,3)))
predictor=cbind(c1,score)
neg.loglike=free.mle(Y.count=data.Y[1,],Y.size=lib.size(data.Y),predictor, categorical)
###  neg.loglike has four negative loglikelihood value according to four
###  different design matrices.
```

---

freefunction *Get the negative log-likelihood by FREE Approach*

Description

We use the beta-binomial modeling to handel the overdispersion of the count data and get the gene specific negative log-likelihood.

Usage

```
freefunction(para, X, Y.col, Y.c)
```

Arguments

para	A vector constains the starting values for beta estimation and psi estimation.
X	Designa matrix.
Y.col	A vector constains the counts of a specified gene.
Y.c	The library size.

Value

this function returns the negative log-likelihood of a specified gene.

Author(s)

Yihui Zhou, Fred A. Wright

Examples

```
###  data.Y is 5000 genes randomly selected from Montgomery data
data(data.Y)
X=cbind(1,c(rep(1,3),rep(0,3),rep(1,3),rep(0,3)))
betastart=c(-15.469,0.221)     ## starting value of beta for the first gene
psistart=-14.840                ## starting value of psi for the first gene
### we use constrained to find the negative -loglikehood of the first gene
neglogl=freefunction(c(betastart,psistart),X,as.numeric(data.Y[1,]),lib.size(data.Y))
```

`get.beta.start` *Find the initial gene-specific staring values for parameters of*

Description

Using current model with given count data of one gene, we approximate the initial values of all the parameters of interests by the estimations from the regular regression.

Usage

```
get.beta.start(X, Y.count, Y.size)
```

Arguments

`X`	the design matrix for the current model.
`Y.count`	the gene-specific count data.
`Y.size`	the library size of the samples

Details

In the simple linear regression, the response is the logit value of the probability of counts among the library size.

Value

a vector of initial values for the parameters we are interested in.

Author(s)

Yihui Zhou, Fred A. Wright

References

See suplementary matrial of Zhou and Wright's paper

Examples

```
data(data.Y)
data.Y=as.matrix(data.Y)
X=cbind(1,c(rep(1,3),rep(0,3),rep(1,3),rep(0,3)))
Y.count=data.Y[1,]               ### counts for the first gene of dataset data.Y
Y.size=lib.size(data.Y)          ### library size of data.Y
beta.estimate=get.beta.start(X,Y.count,Y.size)
```

`get.psi.start` *The initial value of the over-dispersion paramter of each gene.*

Description

Using Beta-Bionimial model to approximate the overdispersion of the count data, the initial value of the logit of the overdispersion is calculated by the given gene-specific count data and the library size.

Usage

```
get.psi.start(Y.count, Y.size)
```

Arguments

`Y.count`	the gene-specific count data.
`Y.size`	the library size of the samples.

Details

Beta-Binomial model is used for modeling the over-diserpsion of the count data. The overdispersion paramter is approximated by the average of the library size and the variance of the counts.

Value

The starting value of the logit of overdispersion parmater.

Note

Y.count has to be in class(integer).

Author(s)

Yihui Zhou, Fred A. Wright

References

Suplementary material of XX paper.

Examples

```
data(data.Y)
data.Y=as.matrix(data.Y)
get.psi.start(Y.count=data.Y[1,],Y.size=lib.size(data.Y))
```

like *get the maximum likelihood matrix*

Description

A likelihood matrix builder. It produces the maximum likelihood matrix according to all different design matrices.

Usage

```
like.matrix(data.matrix,Y.size,quant,predictor)
```

Arguments

data.matrix	the count matrix (m by n) we are going to analysis
Y.size	library size of the dataset
predictor	a matrix which gives all the detailed information for each parameter of interests. Each parameter of interests takes a column; for the discrete variable, the column is a vector indicating the level of each sample; for the continuous variable, it is a vector of 1 with length n (sample size).
quant	a vector which points out if our parameters of interests are continuous or discrete.

Value

This function returns a m by length(X.builder(...)) matrix, each column corresponds to the maximum likelihood value under its current design matrix.

Author(s)

Yihui Zhou, Fred A. Wright

Examples

```
data(data.Y)

set.seed(5)                    ### We simulate score as a continuous covariate
score=rnorm(12,mean=0,sd=1)
categorical=c(1,0)
c1=as.matrix(c(rep(1,3),rep(0,3),rep(1,3),rep(0,3)))
predictor=cbind(c1,score)
X.builder(predictor,categorical)
#### find the maximum likelihood matrix of the first 5 genes
like=like.matrix(data.matrix=data.Y[1:5,],Y.size=lib.size(data.Y),categorical,predictor)
```

`myoptim.free` *Optimization function for estimating the gene-specific parameter of*

Description

This function is to get the optimal estimation of the paratmeters we are interested in using Beta-Binomial modeling of the overdispersion in the RNA-seq count data.

Usage

```
myoptim.free(X, Y.count, Y.size)
```

Arguments

`X`	X is the design matrix for the current model.
`Y.count`	Y.count the gene-specific count data.
`Y.size`	Y.size the library size of the samples.

Value

`like`	negative likelihood of the given gene
`beta`	the optimized beta estimation for one given gene
`psi.free`	the optimized estimation of the logit(overdispersion) for one given gene
`psivar`	the optimized estimation of the variance of logit(overdispersion)
`bvar`	the optimized estimation of the variance of beta for one given gene

Author(s)

Yihui Zhou, Fred A. Wright

Examples

```
data(data.Y)
data.Y=as.matrix(data.Y)
Y.count=data.Y[1,]
Y.size=lib.size(data.Y)
X=cbind(1,c(rep(1,3),rep(0,3),rep(1,3),rep(0,3)))
output=myoptim.free(X,Y.count,Y.size)
beta.free=output$beta                    ### gene-specific beta estimation
psi.free=output$psi.free                 ### gene-specific psi estimation
```

myregression *Simple linear regression function.*

Description

Approximate the parameter we are interested in by simple linear regression function.

Usage

```
myregression(y, x)
```

Arguments

y	a vector of response for the linear regression.
x	degian matrix.

Value

betahat	the parameter estimates from the simple linear regression.
loglike	log-likelihood of y.

Author(s)

Yihui Zhou, Fred A. Wright

outlier *outlier detection function*

Description

This function can find outliers in a group of numbers.

Usage

```
outlier(x)
```

Arguments

x	A vector which needs to be tested.

Value

this function returns a vector. If the number is outlier, then returns TRUE; otherwise returns FALSE.

Author(s)

Yihui Zhou, Fred A. Wright

References

Davies, P.L. and Gather, U. (1993). "The identification of multiple outliers" J. Amer. Statist. Assoc., 88, 782-801.

Examples

```
set.seed(100)
toy=rnorm(100)
toy=c(toy,60,30,10)
result.outlier=outlier(toy)  ## the last three numbers are detected as outliers.
```

`outlier.flag` *flag the suspicious gene*

Description

We "flag" the gene as suspicious if its maximum library-scaled value per gene is more than 5 times as great as its second-largest value.

Usage

```
outlier.flag(data.matrix)
```

Arguments

`data.matrix`	the count matrix (m by n) we are going to analysis

Value

It returns a numerical vector, 1 for suspicious gene and 0 for un-suspicious.

Author(s)

Yihui Zhou, Fred A. Wright

Examples

```
data(data.Y)
flag=outlier.flag(data.Y)
```

`outlier.total` *flag the suspicious gene*

Description

This is a comprehensive function for detecting outliers. We "flag" the gene as suspicious if its maximum library-scaled value per gene is more than 5 times as great as its second-largest value, given the second-largest value is not 0. We also "flag" the genes if they contain more than 95 percentage 0 counts.

Usage

```
outlier.total(data.matrix)
```

Arguments

`data.matrix`	the count matrix (m by n) we are going to analysis

Value

`flag1`	It returns a numerical vector, 1 for the genes with the "ratio" (see description part) greater than 5.
`flag2`	It returns a numerical vector, 1 for those genes with more 95 percentages 0 counts.
`flag.total`	It is the union of genes based on flag1 and flag2.

Author(s)

Yihui Zhou, Fred A. Wright

References

Supplementary Methods of paper "A Powerful and Flexible Approach to the Analysis of RNA Sequence Count Data."

See Also

outlier.flag

Examples

```
data(data.Y)
data.Y=as.matrix(data.Y)
flag1=outlier.total(data.Y)$flag1
flag2=outlier.total(data.Y)$flag2
flag.total=outlier.total(data.Y)$flag.total
```

psi.free *psi estimation from FREE approach*

Description

the estimation of logit of overdispersion parameter from free.estimate.

Usage

```
data(psi.free)
```

Format

The format is: num [1:5000] -13.5 -11.2 -12.7 -15.1 -12 ...

regression *the linear regression function*

Description

We use this simple linear regression function to find the initial value of beta estimation.

Usage

```
regression(y, x)
```

Arguments

y	the response vector
x	explanatory variable

Value

betahat	beta estimation from simple linear regression.

Author(s)

Yihui Zhou, Fred A. Wright

Examples

```
data(data.Y)
s= lib.size(data.Y)
phat=as.vector(as.numeric(data.Y[1,])/s)
phat[phat==0]=1/(2*s[phat==0])
X=cbind(1,c(rep(1,3),rep(0,3),rep(1,3),rep(0,3)))
temp.reg=regression(log(phat/(1-phat)),X)
betahat=temp.reg$betahat    ## betahat is the beta esitmation for the first gene
```

X.builder *Construct different design matrix with respect to the paramters of*

Description

A desgian matrix builder. It produces all different design matrices with respect to the factors in the predictor matrix.

Usage

```
X.builder(predictor, categorical)
```

Arguments

predictor	a matrix which gives all the detailed information for each parameter of interests. Each parameter of interests takes a column; for the discrete variable, the column is a vector indicating the level of each sample; for the continuous variable, it is a vector of 1 with length n (sample size).
categorical	a vector which points out if we treat the parameters of interests are continuous or discrete.

Details

E.g., if there are 2 columns(two factors) in the predictor matrix, it will produce 4 design matrices: 1.a column of ones; 2. a column of ones, plus factor 1; 3. a column of ones, plus factor 2; 4.a column of ones, plus factor 1 and factor 2.

Value

D1,D2,...	the design matrix for the discrete factors in the predictor matrix
Q1,Q2,...	the design matrix for the quantative factors in the predictor matrix
Big	the design matrix containing all the factors
Int	the design matrix only including the intercept

Author(s)

Yihui Zhou, Fred A. Wright

Examples

```
set.seed(5)                    ### We simulate score as a continuous covariate
score=rnorm(12,mean=0,sd=1)
categorical=c(1,0)
c1=as.matrix(c(rep(1,3),rep(0,3),rep(1,3),rep(0,3)))
predictor=cbind(c1,score)
X.builder(predictor,categorical)
```

Index