Package ‘NBAMSeq’

March 14, 2024

Type Package
Title Negative Binomial Additive Model for RNA-Seq Data
Version 1.18.0
Description High-throughput sequencing experiments followed by differential expression analysis is a widely used approach to detect genomic biomarkers. A fundamental step in differential expression analysis is to model the association between gene counts and covariates of interest. NBAMSeq a flexible statistical model based on the generalized additive model and allows for information sharing across genes in variance estimation.

License GPL-2
URL https://github.com/reese3928/NBAMSeq
BugReports https://github.com/reese3928/NBAMSeq/issues
Encoding UTF-8
Imports DESeq2, mgcv(>= 1.8-24), BiocParallel, genefilter, methods, stats,
Depends R (>= 3.6), SummarizedExperiment, S4Vectors
Suggests knitr, rmarkdown, testthat, ggplot2
RoxygenNote 6.1.0
VignetteBuilder knitr
biocViews RNASeq, DifferentialExpression, GeneExpression, Sequencing, Coverage

git_url https://git.bioconductor.org/packages/NBAMSeq
git_branch RELEASE_3_18
git_last_commit 56721c1
git_last_commit_date 2023-10-24
Repository Bioconductor 3.18
Date/Publication 2024-03-13
Author Xu Ren [aut, cre], Pei Fen Kuan [aut]
Maintainer Xu Ren <xuren2120@gmail.com>
**Description**

This function makes an example NBAMSeqDataSet.

**Usage**

```r
makeExample(n = 200, m = 30)
```

**Arguments**

- `n` number of genes
- `m` number of samples

**Value**

a NBAMSeqDataSet object

**References**


**Examples**

```r
gsd = makeExample()
```
**makeplot**  

*Making plots to visualize nonlinear associations*

**Description**

This function makes plots to visualize nonlinear associations.

**Usage**

```r
makeplot(object, phenoname, genename, ...)  
```

**Arguments**

- `object`: a NBAMSeqDataSet object
- `phenoname`: the name of nonlinear variable to be visualized
- `genename`: the name of gene to be visualized
- `...`: additional arguments provided to `plot.gam`

**Value**

the plot made by `plot.gam()` function

**Examples**

```r
gsd = makeExample(n = 3, m = 10)
gsd = NBAMSeq(gsd)
makeplot(gsd, "pheno", "gene3", main = "gene10")
```

---

**NBAMSeq**  

*Differential expression analysis based on negative binomial additive model*

**Description**

This function performs differential expression analysis based on negative binomial additive model.

**Usage**

```r
NBAMSeq(object, gamma = 2.5, parallel = FALSE, fitlin = FALSE,  
BPPARAM = bpparam(), ...)  
```
Arguments

object a NBAMSeqDataSet object
gamma a number greater or equal to 1. Increase gamma to create smoother models. Default gamma is 2.5. See \texttt{gam} for details.
parallel either TRUE or FALSE indicating whether parallel should be used. Default is FALSE
fitlin either TRUE or FALSE indicating whether linear model should be fitted. Default is FALSE
BPPARAM an argument provided to \texttt{bplapply}. See \texttt{register} for details.
... additional arguments provided to \texttt{gam}

Value

a NBAMSeqDataSet object

References


Examples

gsd = makeExample(n = 3, m = 10)
gsd = NBAMSeq(gsd)

Description

Accessor functions and replace methods for NBAMSeqDataSet object
For \texttt{getDesign()}: accessor to the design formula
For \texttt{getsf()}: accessor to the size factors
Replace methods for NBAMSeqDataSet object
For \texttt{setsf()}: replace size factors

Usage

gDesign\(\texttt{theObject}\)

## S4 method for signature 'NBAMSeqDataSet'
gDesign\(\texttt{theObject}\)

getsf\(\texttt{theObject}\)
## S4 method for signature 'NBAMSeqDataSet'
getsf(theObject)

setsf(theObject) <- value

## S4 replacement method for signature 'NBAMSeqDataSet,numeric'
setsf(theObject) <- value

### Arguments

theObject  a NBAMSeqDataSet object
value      the values to be included in the object

### Value

For `getDesign()`: design formula
For `getsf()`: size factor
For `setsf()`: NBAMSeq object

### References


### Examples

```r
## For getDesign() ##
gsd = makeExample()
design_gsd = getDesign(gsd)

## For getsf() ##
gsd = makeExample()
sf = getsf(gsd)

## For setsf() ##
n = 100
m = 50
gsd = makeExample(n = n, m = m)
sf = sample(1:5, m, replace = TRUE)
setsf(gsd) = sf
```

---

**NBAMSeqDataSet**  

**NBAMSeqDataSet constructor**

---

**Description**

NBAMSeqDataSet constructor
Usage

NBAMSeqDataSet(countData, colData, design, ...)

Arguments

- countData: a matrix or data frame contains gene count
- colData: a DataFrame or data.frame
- design: a mgcv type design. e.g. ~ s(pheno) or ~ s(pheno) + var1 + var2
- ... optional arguments passed to SummarizedExperiment

Value

a NBAMSeqDataSet object

Examples

n = 100  ## n stands for number of genes
m = 20   ## m stands for sample size
countData = matrix(rnbinom(n*m, mu=100, size=1/3), ncol = m)
mode(countData) = "integer"
colnames(countData) = paste0("sample", 1:m)
rownames(countData) = paste0("gene", 1:n)
pheno = runif(m, 20, 80)
colData = data.frame(pheno = pheno)
colnames(colData) = paste0("sample", 1:m)
gsd = NBAMSeqDataSet(countData = countData, 
colData = colData, design = ~s(pheno))

Description

NBAMSeqDataSet is a class inherited from SummarizedExperiment. It is used to store the count matrix, colData, and design formula in differential expression analysis.

Slots

design a mgcv-type design formula

References

**Description**

This function pulls out result from NBAMSeqDataSet object returned by `NBAMSeq`.

**Usage**

```r
results(object, name, contrast, indepfilter = TRUE, alpha = 0.1, 
pAdjustMethod = "BH", parallel = FALSE, BPPARAM = bpparam(), ...)
```

**Arguments**

- `object`: a NBAMSeqDataSet object returned by `NBAMSeq`.
- `name`: the name of nonlinear variable or continuous linear variable.
- `contrast`: a character of length 3. 1st element: name of factor variable; 2nd element: name of numerator level; 3rd element: name of denominator level. `contrast = c("group", "treatment", "control")` means comparing treatment vs control for group variable.
- `indepfilter`: either TRUE or FALSE indicating whether independent filtering should be performed. Default is TRUE.
- `alpha`: significant threshold for declaring genes as differentially expressed. Default is 0.1.
- `parallel`: either TRUE or FALSE indicating whether parallel should be used. Default is FALSE.
- `BPPARAM`: an argument provided to `bplapply`. See `register` for details.
- `...`: additional arguments provided to `pvalueAdjustment` function in DESeq2. See `results` for details.

**Value**

A DataFrame which contains the result.

**References**


**Examples**

```r
gsd = makeExample(n = 3, m = 10)
gsd = NBAMSeq(gsd)
res = results(gsd, name = "pheno")
```
Index

bplapply, 4, 7

gam, 4
getDesign (NBAMSeq-methods), 4
getDesign, NBAM_seqDataSet-method
(NBAMSeq-methods), 4
getsf (NBAMSeq-methods), 4
getsf, NBAM_seqDataSet-method
(NBAMSeq-methods), 4

makeExample, 2
makeplot, 3

NBAMSeq, 3, 7
NBAMSeq-methods, 4
NBAMSeqDataSet, 5
NBAMSeqDataSet-class, 6

p.adjust, 7
plot.gam, 3

register, 4, 7
results, 7, 7

setsf<- (NBAMSeq-methods), 4
setsf<-, NBAMSeqDataSet, numeric-method
(NBAMSeq-methods), 4
SummarizedExperiment, 6