Package ‘ROSeq’

April 6, 2024

Type Package
Title Modeling expression ranks for noise-tolerant differential expression analysis of scRNA-Seq data
Version 1.14.0
Description ROSeq - A rank bused approach to modeling gene expression with filtered and normalized read count matrix. ROSeq takes filtered and normalized read matrix and cell-annotation/condition as input and determines the differentially expressed genes between the contrasting groups of single cells. One of the input parameters is the number of cores to be used.

URL https://github.com/krishan57gupta/ROSeq
BugReports https://github.com/krishan57gupta/ROSeq/issues
License GPL-3
Encoding UTF-8
LazyData true
RoxygenNote 7.1.1
Depends R (>= 4.0)
biocViews GeneExpression, DifferentialExpression, SingleCell
Imports pbmcapply, edgeR, limma
Suggests knitr, markdown, testthat, RUnit, BiocGenerics
VignetteBuilder knitr
git_url https://git.bioconductor.org/packages/ROSeq
git_branch RELEASE_3_18
git_last_commit 356733a
git_last_commit_date 2023-10-24
Repository Bioconductor 3.18
Date/Publication 2024-04-05
Author  Krishan Gupta [aut, cre],
Manan Lalit [aut],
Aditya Biswas [aut],
Abhik Ghosh [aut],
Debarka Sengupta [aut]

Maintainer  Krishan Gupta <krishang@iiitd.ac.in>

R topics documented:

- computeDEG
- findParams
- getd
- getDataStatistics
- getdu1da
- getdu1db
- getdu2da
- getdu2db
- getdvda
- getdvdb
- getI
- getu1
- getu2
- getv
- initiateAnalysis
- L_Tung_single
- minimizeNLL
- ROSeq
- TMMnormalization

Index  14

computeDEG  Computes differential expression for the gene in question, by comparing the optimal parameters for sub-populations one and two

Description

Uses the (asymptotically) optimum two-sample Wald test based on the MLE of the parameters and its asymptotic variances given by the inverse of the Fisher information matrix

Usage

computeDEG(results_1, results_2)
**findParams**

**Arguments**

- **results_1**
  A vector corresponding to sub-population one and containing 5 values (a, b, A, number of bins, R2)

- **results_2**
  A vector corresponding to sub-population two and containing 5 values (a, b, A, number of bins, R2)

**Value**

- **T** The Wald test statistic for testing the null hypothesis

**See Also**

- `getI`, `findParams`

---

**findParams**

Finds the optimal values of parameters a and b that model the probability distribution of ranks, by Maximising the Log-Likelihood

**Description**

Takes in as input the read count data corresponding to one sub-population and the typical gene statistics. Then it splits the entire range into equally sized bins of size $k \times \sigma$, where $k$ is a scalar with a default value of 0.05, and $\sigma$ is the standard deviation of the pulled expression estimates across the cell-groups. Each of these bins corresponds to a rank. Therefore, for each group, cell frequency for each bin maps to a rank. These frequencies are normalized group-wise by dividing by the total cell count within a concerned group.

**Usage**

```r
findParams(ds, geneStats)
```

**Arguments**

- **ds**
  The (normalized and filtered) read count data corresponding to a sub-population

- **geneStats**
  A vector containing 7 values corresponding to the gene data (maximum, minimum, mean, standard deviation, upper multiple of the standard deviation, lower multiple of standard deviation and log_2(fold change))

**Value**

- **results** A vector containing 5 values (a, b, A, number of bins, R2)
getd

Finds the double derivative of A

Description
Finds the double derivative of A with respect to a, (a, b), b , (a, b) in respective templates from right to left. This first derivative is evaluated at the optimal (a_hat, b_hat). u1, v and u2 constitute the equations required for evaluating the first and second order derivatives of A with respect to parameters a and b.

Usage
getd(u1, v, du1da, dvda)

Arguments
- u1: First derivative of u1 with respect to parameter a
- v: First derivative of v with respect to parameter a
- du1da: First derivative of u1 with respect to parameter a
- dvda: First derivative of v with respect to parameter a

Value
d2logAda2

getDataStatistics

Evaluates statistics of the read counts corresponding to the gene

Description
Takes in the complete read count vector corresponding to the gene (sp) and also the data corresponding to the two sub-populations (sp1 and sp2).

Usage
getDataStatistics(sp, spOne, spTwo)

Arguments
- sp: The complete (normalized and filtered) read count data corresponding to the gene in question
- spOne: The (normalized and filtered) read count data corresponding to the first sub-population
- spTwo: The (normalized and filtered) read count data corresponding to the second sub-population
Value

geneStats A vector containing 6 values corresponding to the gene data (maximum, minimum, mean, standard deviation, upper multiple of standard deviation and lower multiple of standard deviation).

---

getdu1da

Finds the first derivative of u1 with respect to a. This first derivative is evaluated at the optimal (a_hat, b_hat).

---

Description

u1, v and u2 constitute the equations required for evaluating the first and second order derivatives of A with respect to parameters a and b.

Usage

getdu1da(coefficients, r)

Arguments

coefficients the optimal values of a and b
r the rank vector

Value

du1da

---

getdu1db

Finds the first derivative of u1 with respect to b. This first derivative is evaluated at the optimal (a_hat, b_hat).

---

Description

u1, v and u2 constitute the equations required for evaluating the first and second order derivatives of A with respect to parameters a and b.

Usage

getdu1db(coefficients, r)

Arguments

coefficients the optimal values of a and b
r the rank vector

Value

du1db
getdu2da

Finds the first derivative of u2 with respect to a. This first derivative is evaluated at the optimal (a_hat, b_hat).

Description

u1, v and u2 constitute the equations required for evaluating the first and second order derivatives of A with respect to parameters a and b

Usage

getu2da(coefficients, r)

Arguments

coefficients the optimal values of a and b
r the rank vector

Value

du2da

getdu2db

Finds the first derivative of u2 with respect to b. This first derivative is evaluated at the optimal (a_hat, b_hat).

Description

u1, v and u2 constitute the equations required for evaluating the first and second order derivatives of A with respect to parameters a and b

Usage

getu2db(coefficients, r)

Arguments

coefficients the optimal values of a and b
r the rank vector

Value

du2db
getdvda

Finds the first derivative of v with respect to a. This first derivative is evaluated at the optimal \((a_{\hat{a}}, b_{\hat{b}})\).

**Description**

\(u_1, v\) and \(u_2\) constitute the equations required for evaluating the first and second order derivatives of \(A\) with respect to parameters \(a\) and \(b\).

**Usage**

getdvda(coefficients, r)

**Arguments**

- coefficients: the optimal values of \(a\) and \(b\)
- r: the rank vector

**Value**

dvda

getdvdb

Finds the first derivative of v with respect to b. This first derivative is evaluated at the optimal \((a_{\hat{a}}, b_{\hat{b}})\).

**Description**

\(u_1, v\) and \(u_2\) constitute the equations required for evaluating the first and second order derivatives of \(A\) with respect to parameters \(a\) and \(b\).

**Usage**

getdvdb(coefficients, r)

**Arguments**

- coefficients: the optimal values of \(a\) and \(b\)
- r: the rank vector

**Value**

dvdb
getI Computes the Fisher Information Matrix

Description
The Fisher Information Matrix and its derivatives are essential to evaluate the minima of log likelihood.

Usage
getI(results)

Arguments
results A vector containing 5 values (a, b, A, number of bins, R2)

Value
I The Fisher Information Matrix

getu1 Computes u1

Description
u1, v and u2 constitute the equations required for evaluating the first and second order derivatives of A with respect to parameters a and b.

Usage
getu1(coefficients, r)

Arguments
coefficients the optimal values of a and b
r the rank vector

Value
u1
getu2

Computes $u_2$

Description

$u_1$, $v$ and $u_2$ constitute the equations required for evaluating the first and second order derivatives of $A$ with respect to parameters $a$ and $b$.

Usage

getu2(coefficients, $r$)

Arguments

coefficients the optimal values of $a$ and $b$

$r$ the rank vector

Value

$u_2$

---

getv

Computes $v$

Description

$u_1$, $v$ and $u_2$ constitute the equations required for evaluating the first and second order derivatives of $A$ with respect to parameters $a$ and $b$.

Usage

getv(coefficients, $r$)

Arguments

coefficients the optimal values of $a$ and $b$

$r$ the rank vector

Value

$v$
**initiateAnalysis**  
*Computes differential analysis for a given gene*

**Description**
Takes in the row index which corresponds to a gene and evaluates for differential expression across two cell types.

**Usage**

```r
initiateAnalysis(gene, scdata, scgroups, classOne, classTwo)
```

**Arguments**

- **gene**  
The row index of the normalised and filtered, read count matrix
- **scdata**  
The normalised and filtered, read count matrix
- **scgroups**  
The location of the two sub-populations
- **classOne**  
The location of the first sub-population, for example, sample names as given as columns names
- **classTwo**  
The location of the second sub-population, for example, sample names as given as columns names

**Value**

`combinedResults` A vector containing 12 values (gr1: a, g1: b, gr1: A, gr1: number of bins, gr1: R2, gr2: a, gr2: b, gr2: A, gr2: number of bins, gr2: R2, T, p)

---

**L_Tung_single**  
*Single cell samples for DE genes analysis*

**Description**
Three replicates from three human induced pluripotent stem cell (iPSC) lines were considered. 96 single cells were considered in each of the three replicates corresponding to one of the three individuals (these shall be referred to by their labels NA19098, NA19101 and NA19239)

**Usage**

```r
data("L_Tung_single")
```

**Format**

The format is: list of cells corresponding NA19098 versus NA19101 and groups labels.
`minimizeNLL`  

Details  

filtered and normalized data  

Source  


References  


Examples  

```r  
data(L_Tung_single)  
## summary(ROSeq::L_Tung_single)  
```

---  

`minimizeNLL`  

`minimizeNLL`  

*Minimizes the Negative Log-Likelihood by iterating across values of parameters a and b*  

Description  

Takes in as input a vector of values (coefficients), the number of bins and the normalized probability distribution of ranks  

Usage  

`minimizeNLL(coefficients, r, readCount)`  

Arguments  

- `coefficients`: A vector containing two values for a and b  
- `r`: The number of bins  
- `readCount`: A vector of (normalized) frequency of read counts that fall within each bin  

Value  

NLL Negative-Log Likelihood for the input coefficients  

See Also  

`findParams`
ROSeq  

Modeling expression ranks for noise-tolerant differential expression analysis of scRNA-Seq data

Description

Takes in the complete filtered and normalized read count matrix, the location of the two sub-populations and the number of cores to be used

Usage

ROSeq(countData, condition, numCores = 1)

Arguments

countData The normalised and filtered, read count matrix, with row names as genes name/ID and column names as sample id/name

condition Labels for the two sub-populations

numCores The number of cores to be used

Value

pValues and FDR adjusted p significance values

Examples

countData<-list()
countData$count<-ROSeq::L_Tung_single$NA19098_NA19101_count
countData$group<-ROSeq::L_Tung_single$NA19098_NA19101_group
head(countData$count)
gene_names<rownames(countData$count)
countData$count<apply(countData$count,2,function(x) as.numeric(x))
rownames(countData$count)<gene_names
countData$count[count[,colSums(countData$count> 0) > 2000]
g_keep <- apply(countData$count,1,function(x) sum(x>=2)>=3)
countData$count[count_2[g_keep,]
countData$count<-limma::voom(ROSeq::TMMnormalization(countData$count))
output<-ROSeq(countData=countData$count$E, condition = countData$group)
output
**TMMnormalization**

**TMM Normalization.**

**Description**
Trimmed Means of M values (TMM) normalization (on the basis of edgeR package)

**Usage**

```r
TMMnormalization(countTable)
```

**Arguments**

- `countTable` The filtered, read count matrix, with row names as genes name/ID and column names as sample id/name

**Value**

`countTableTMM`

**Examples**

```r
countData <- list()
countData$count <- ROSeq::L_Tung_single$NA19098_NA19101_count
countData$group <- ROSeq::L_Tung_single$NA19098_NA19101_group
head(countData$count)
gene_names <- rownames(countData$count)
countData$count <- apply(countData$count, 2, function(x) as.numeric(x))
rownames(countData$count) <- gene_names
countData$count <- countData$count[, colSums(countData$count > 0) > 2000]
g_keep <- apply(countData$count, 1, function(x) sum(x > 2) >= 3)
countData$count <- countData$count[g_keep,]
countTableTMM <- ROSeq::TMMnormalization(countData$count)
countTableTMM
```
# Index

* **datasets**
  - L_Tung_single, 10

  computeDEG, 2

  findParams, 3, 3, 11

  getd, 4

  getDataStatistics, 4

  getdu1da, 5

  getdu1db, 5

  getdu2da, 6

  getdu2db, 6

  getdvda, 7

  getdvdb, 7

  getI, 3, 8

  getu1, 8

  getu2, 9

  getv, 9

  initiateAnalysis, 10

  L_Tung_single, 10

  minimizeNLL, 11

  ROSeq, 12

  TMMnormalization, 13