

# Package ‘ROSeq’

May 2, 2024

**Type** Package

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

**Version** 1.16.0

**Description** ROSeq - A rank based 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, rmarkdown, testthat, RUnit, BiocGenerics

**VignetteBuilder** knitr

**git\_url** <https://git.bioconductor.org/packages/ROSeq>

**git\_branch** RELEASE\_3\_19

**git\_last\_commit** d804d7b

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.19

**Date/Publication** 2024-05-01

**Author** Krishan Gupta [aut, cre],  
 Manan Lalit [aut],  
 Aditya Biswas [aut],  
 Abhik Ghosh [aut],  
 Debarka Sengupta [aut]

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

## Contents

computeDEG . . . . .	2
findParams . . . . .	3
getd . . . . .	4
getDataStatistics . . . . .	4
getdu1da . . . . .	5
getdu1db . . . . .	5
getdu2da . . . . .	6
getdu2db . . . . .	6
getdvda . . . . .	7
getdvdb . . . . .	7
getI . . . . .	8
getu1 . . . . .	8
getu2 . . . . .	9
getv . . . . .	9
initiateAnalysis . . . . .	10
L_Tung_single . . . . .	10
minimizeNLL . . . . .	11
ROSeq . . . . .	12
TMMnormalization . . . . .	13
<b>Index</b>	<b>14</b>

---

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

---

## 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)
```

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

---

**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  $k * \sigma$ , 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**

```
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 <sub>2</sub> (fold change))

**Value**

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

---

getd	<i>Finds the double derivative of A</i>
------	---

---

### 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 ( $\hat{a}$ ,  $\hat{b}$ ). 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	u1
v	v
du1da	First derivative of u1 with respect to parameter a
dvda	First derivative of v with respect to parameter a

### Value

d2logAda2

---

getDataStatistics	<i>Evaluates statistics of the read counts corresponding to the gene</i>
-------------------	--

---

### 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	<i>Finds the first derivative of u1 with respect to a. This first derivative is evaluated at the optimal (a_hat, b_hat).</i>
----------	--

---

**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	<i>Finds the first derivative of u1 with respect to b. This first derivative is evaluated at the optimal (a_hat, b_hat).</i>
----------	--

---

**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	<i>Finds the first derivative of u2 with respect to a. This first derivative is evaluated at the optimal (a_hat, b_hat).</i>
----------	--

---

**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**

```
getdu2da(coefficients, r)
```

**Arguments**

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

**Value**

du2da

---

getdu2db	<i>Finds the first derivative of u2 with respect to b. This first derivative is evaluated at the optimal (a_hat, b_hat).</i>
----------	--

---

**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**

```
getdu2db(coefficients, r)
```

**Arguments**

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

**Value**

du2db

---

getdvda	<i>Finds the first derivative of v with respect to a. This first derivative is evaluated at the optimal (a_hat, b_hat).</i>
---------	---

---

**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**

```
getdvda(coefficients, r)
```

**Arguments**

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

**Value**

dvda

---

getdvdb	<i>Finds the first derivative of v with respect to b. This first derivative is evaluated at the optimal (a_hat, b_hat).</i>
---------	---

---

**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**

```
getdvdb(coefficients, r)
```

**Arguments**

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

**Value**

dvdb

---

getI	<i>Computes the Fisher Information Matrix</i>
------	---

---

**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	<i>Computes u1</i>
-------	--------------------

---

**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	<i>Computes u2</i>
-------	--------------------

---

**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**

```
getu2(coefficients, r)
```

**Arguments**

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

**Value**

u2

---

getv	<i>Computes v</i>
------	-------------------

---

**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**

```
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

```
initiateAnalysis(gene, sdata, scgroups, classOne, classTwo)
```

### Arguments

<code>gene</code>	The row index of the normalised and filtered, read count matrix
<code>sdata</code>	The normalised and filtered, read count matrix
<code>scgroups</code>	The location of the two sub-populations
<code>classOne</code>	The location of the first sub-population, for example, sample names as given as columns names
<code>classTwo</code>	The location of thesecond 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

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

### Format

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

**Details**

filtered and normalized data

**Source**

Tung, P.-Y. et al. Batch effects and the effective design of single-cell gene expression studies. *Scientific reports* 7, 39921 (2017).

**References**

Tung, P.-Y. et al. Batch effects and the effective design of single-cell gene expression studies. *Scientific reports* 7, 39921 (2017).

**Examples**

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

---

minimizeNLL	<i>Minimizes the Negative Log-Likelihood by iterating across values of parameters a and b</i>
-------------	---

---

**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	<i>Modeling expression ranks for noise-tolerant differential expression analysis of scRNA-Seq data</i>
-------	--

---

### 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<-countData$count[,colSums(countData$count> 0) > 2000]
g_keep <- apply(countData$count,1,function(x) sum(x>2)>=3)
countData$count<-countData$count[g_keep,]
countData$count<-limma::voom(ROSeq::TMMnormalization(countData$count))
output<-ROSeq(countData=countData$count$E, condition = countData$group)
output
```

---

TMMnormalization	<i>TMM Normalization.</i>
------------------	---------------------------

---

**Description**

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

**Usage**

```
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**

```
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](#)