Package ‘satuRn’

May 7, 2024

Type Package
Title Scalable Analysis of Differential Transcript Usage for Bulk and Single-Cell RNA-sequencing Applications
Date 2022-08-05
Version 1.12.0
Description satuRn provides a highly performant and scalable framework for performing differential transcript usage analyses. The package consists of three main functions. The first function, fitDTU, fits quasi-binomial generalized linear models that model transcript usage in different groups of interest. The second function, testDTU, tests for differential usage of transcripts between groups of interest. Finally, plotDTU visualizes the usage profiles of transcripts in groups of interest.

Depends R (>= 4.1)
Imports locfdr, SummarizedExperiment, BiocParallel, limma, pbapply, ggplot2, boot, Matrix, stats, methods, graphics
Suggests knitr, rmarkdown, testthat, covr, BiocStyle, AnnotationHub, ensembldb, edgeR, DEXSeq, stageR, DelayedArray
VignetteBuilder knitr
Collate 'data.R' 'satuRn-framework.R' 'allGenerics.R' 'accessors.R' 'fitDTU.R' 'testDTU.R' 'plotDTU.R'
License Artistic-2.0
URL https://github.com/statOmics/satuRn
BugReports https://github.com/statOmics/satuRn/issues
Encoding UTF-8
LazyData false
RoxygenNote 7.2.0
biocViews Regression, ExperimentalDesign, DifferentialExpression, GeneExpression, RNASeq, Sequencing, Software, SingleCell, Transcriptomics, MultipleComparison, Visualization

1
Description

Parameter estimation of quasi-binomial models.

Usage

fitDTU(object, ...)

## S4 method for signature 'SummarizedExperiment'
fitDTU(
  object,
  formula,
  parallel = FALSE,
  BPPARAM = BiocParallel::bpparam(),
  verbose = TRUE
)
**getModel,StatModel-method**

**Arguments**

- **object**
  A ‘SummarizedExperiment’ instance generated with the `SummarizedExperiment` function of the `SummarizedExperiment` package. In the assay slot, provide the transcript-level expression counts as an ordinary ‘matrix’, ‘DataFrame’, a ‘sparseMatrix’ or a ‘DelayedMatrix’. The ‘rowData’ slot must be a ‘DataFrame’ object describing the rows, which must contain a column ‘isoform_id’ with the row names of the expression matrix and a column ‘gene_id’ with the corresponding gene identifiers of each transcript. ‘colData’ is a ‘DataFrame’ describing the samples or cells in the experiment. Finally, specify the experimental design as a formula in the metadata slot. This formula must be based on the `colData`. See the documentation examples and the vignette for more details.

- **formula**
  Model formula. The model is built based on the covariates in the data object.

- **parallel**
  Logical, defaults to `FALSE`. Set to `TRUE` if you want to parallelize the fitting procedure.

- **BPPARAM**
  Object of class `bpparamClass` that specifies the back-end to be used for computations. See `bpparam` in the `BiocParallel` package for details.

- **verbose**
  Logical, should progress be printed?

**Value**

An updated ‘SummarizedExperiment’ instance. The instance now includes a new list of models ("fitDTUModels") in its rowData slot, which can be accessed by `rowData(object)[["fitDTUModels"]].

**Author(s)**

Jeroen Gilis

**Examples**

```r
data(sumExp_example, package = "satuRn")
sumExp <- fitDTU(
  object = sumExp_example,
  formula = ~ 0 + group,
  parallel = FALSE,
  BPPARAM = BiocParallel::bpparam(),
  verbose = TRUE
)
```

**Accessor functions for StatModel class**
Description

Accessor functions for StatModel class

getModel(object) to get model
getDF(object) to get the residual degrees of freedom of the model
getDfPosterior(object) to get the degrees of freedom of the empirical Bayes variance estimator
getDispersion(object) to get the dispersion estimate of the model
getCoef(object) to get the parameter estimates of the mean model

Usage

## S4 method for signature 'StatModel'
getModel(object)

## S4 method for signature 'StatModel'
getDF(object)

## S4 method for signature 'StatModel'
getDfPosterior(object)

## S4 method for signature 'StatModel'
getDispersion(object)

## S4 method for signature 'StatModel'
getCoef(object)

Arguments

object StatModel object

Value

The requested parameter of the StatModel object

Examples

## A fully specified dummy model
myModel <- StatModel(
  type = "glm",
  params = list(x = 3, y = 7, b = 4),
  varPosterior = c(0.1, 0.2, 0.3),
  dfPosterior = c(6, 7, 8)
)
getModel(myModel)
**plotDTU**

*Plot function to visualize differential transcript usage (DTU)*

**Description**

Plot function to visualize differential transcript usage (DTU)

**Usage**

```r
plotDTU(
  object, 
  contrast, 
  groups, 
  coefficients, 
  summaryStat = "model", 
  transcripts = NULL, 
  genes = NULL, 
  top.n = 6
)
```

**Arguments**

- **object** A 'SummarizedExperiment' containing the models and results of the DTU analysis as obtained by the 'fitDTU' and 'testDTU' function from this 'satuRn' package, respectively.
- **contrast** Specifies the specific contrast for which the visualization should be returned. This should be one of the column names of the contrast matrix that was provided to the 'testDTU' function.
- **groups** A 'list' containing two character vectors. Each character vector contains the names (sample names or cell names) of the respective groups in the target contrast.
- **coefficients** A 'list' containing two numeric vectors. Each numeric vector specifies the model coefficient of the corresponding groups in the selected contrast.
- **summaryStat** Which summary statistic for the relative usage of the transcript should be displayed. 'Character' or 'character vector', must be any of following summary statistics; model (default), mean or median.
- **transcripts** A 'character' or 'character vector' of transcript IDs, to specify which transcripts should be visualized. Can be used together with 'genes'. If not specified, 'plotDTU' will check if the 'genes' slot is specified.
- **genes** A single 'character' or 'character vector' of gene IDs, to specify the genes for which the individual transcripts should be visualized. Can be used together with 'transcripts'. If not specified, 'plotDTU' will check if the 'transcripts' slot is specified.
- **top.n** A 'numeric' value. If neither 'transcripts' nor 'genes' was was specified, this argument leads to the visualization of the 'n' most significant DTU transcripts in the contrast. Defaults to 6 transcripts.
**Value**

A ggplot object that can be directly displayed in the current R session or stored in a list.

**Author(s)**

Jeroen Gilis

**Examples**

```r
data(sumExp_example, package = "satuRn")
library(SummarizedExperiment)
sumExp <- fitDTU(
  object = sumExp_example,
  formula = ~ 0 + group,
  parallel = FALSE,
  BPPARAM = BiocParallel::bpparam(),
  verbose = TRUE
)
group <- as.factor(colData(sumExp)$group)
design <- model.matrix(~ 0 + group)
colnames(design) <- levels(group)
L <- matrix(0, ncol = 2, nrow = ncol(design))
rownames(L) <- colnames(design)
colnames(L) <- c("Contrast1", "Contrast2")
L[c("VISp.L5_IT_VISp_Hsd11b1_Endou", "ALM.L5_IT_ALM_Tnc"), 1] <- c(1, -1)
L[c("VISp.L5_IT_VISp_Hsd11b1_Endou", "ALM.L5_IT_ALM_Tnc63_Dmrtb1"), 2] <- c(1, -1)

sumExp <- testDTU(object = sumExp,
  contrasts = L,
  diagplot1 = FALSE,
  diagplot2 = FALSE,
  sort = FALSE)
group1 <- rownames(colData(sumExp))[colData(sumExp)$group == "VISp.L5_IT_VISp_Hsd11b1_Endou"]
group2 <- rownames(colData(sumExp))[colData(sumExp)$group == "ALM.L5_IT_ALM_Tnc63"]

plots <- plotDTU(
  object = sumExp,
  contrast = "Contrast1",
  groups = list(group1, group2),
  coefficients = list(c(0, 0, 1), c(0, 1, 0)),
  summaryStat = "model",
  transcripts = c("ENSMUST00000165123", "ENSMUST00000165721", "ENSMUST0000005067"),
  genes = NULL,
  top.n = 6)
```

plotDTU
StatModel

Description

Function for constructing a new 'StatModel' object.

Usage

StatModel(
  type = "fitError",
  params = list(),
  varPosterior = NA_real_,
  dfPosterior = NA_real_
)

Arguments

type                default set to fiterror, can be a glm
params              A list containing the parameters of the fitted glm
varPosterior        Numeric, posterior variance of the glm, default is NA
dfPosterior         Numeric, posterior degrees of freedom of the glm, default is NA

Value

A StatModel object

Examples

## A fully specified dummy model
myModel <- StatModel(
  type = "glm",
  params = list(x = 3, y = 7, b = 4),
  varPosterior = c(0.1, 0.2, 0.3),
  dfPosterior = c(6, 7, 8)
)
myModel
The `StatModel` class contains a statistical model generated by the DTU analysis. Models are created by the dedicated user-level function `fitDTU()` or manually, using the `StatModel()` constructor. In the former case, each quantitative feature is assigned its statistical model and the models are stored as a variable in a 'DataFrame' object, that in turn will be stored in the 'RowData' slot of a 'SummarizedExperiment object'.

**Slots**
- `type` 'character(1)' defining type of the used model. Default is "fitError", i.e. an error model. If not an error, class type will be "glm".
- `params` A list() containing the parameters of the fitted model.
- `varPosterior` numeric() of posterior variance.
- `dfPosterior` numeric() of posterior degrees of freedom.

**Author(s)**
Jeroen Gilis

**Examples**
```r
## A fully specified dummy model
myModel <- StatModel(
  type = "glm",
  params = list(x = 3, y = 7, b = 4),
  varPosterior = c(0.1, 0.2, 0.3),
  dfPosterior = c(6, 7, 8)
)
myModel
```

A 'SummarizedExperiment' derived from our case study which builds on the dataset of Tasic et al. It contains the same cells as the data object used in the vignette (see '?Tasic_counts_vignette' for more information). In this SummarizedExperiment, we performed a filtering with 'filterByExpr' of edgeR with more stringent than default parameter settings (min.count = 100, min.total.count = 200, large.n = 50, min.prop = 0.9) to reduced the number of retained transcripts. We used this object to create an executable example in the help files of satuRn.
**Tasic_counts_vignette**

**Description**

A ‘SummarizedExperiment’ derived from our case study which builds on the dataset of Tasic et al. It contains the same cells as the data object used in the vignette (see ‘?Tasic_counts_vignette’ for more information). In this SummarizedExperiment, we performed a filtering with ‘filterByExpr’ of edgeR with more stringent than default parameter settings (min.count = 100, min.total.count = 200, large.n = 50, min.prop = 0.9) to reduced the number of retained transcripts. We used this object to create an executable example in the help files of satuRn.

**Usage**

```r
data(sumExp_example)
```

**Format**

An object of class SummarizedExperiment with 1286 rows and 60 columns.

---

**Tasic_counts_vignette**

A ‘Matrix’ with transcript-level counts derived from our case study which builds on the dataset of Tasic et al. We used Salmon (V1.1.0) to quantify all L5IT cells (both for ALM and VISp tissue) from mice with a normal eye condition. From these cells, we randomly sampled 20 cells of each of the following cell types to use for this vignette; L5_IT_VISp_Hsd11b1_Endou, L5_IT_ALM_Tmem163_Dmrtb1 and L5_IT_ALM_Tnc. The data has already been leniently filtered with the ‘filterByExpr’ function of edgeR.

**Usage**

```r
data(Tasic_counts_vignette)
```

**Format**

An object of class matrix (inherits from array) with 22273 rows and 60 columns.
testDTU

Tasic_metadata_vignette

Metadata associated with the expression matrix 'Tasic_counts_vignette'. See '?Tasic_counts_vignette' for more information on the dataset.

Description

Metadata associated with the expression matrix 'Tasic_counts_vignette'. See '?Tasic_counts_vignette' for more information on the dataset.

Usage

data(Tasic_metadata_vignette)

Format

An object of class data.frame with 60 rows and 3 columns.

testDTU

Test function to obtain a top list of transcripts that are differentially used in the contrast of interest

Description

Function to test for differential transcript usage (DTU)

Usage

testDTU(object, contrasts, diagplot1 = TRUE, diagplot2 = TRUE, sort = FALSE)

Arguments

object
  A ‘SummarizedExperiment’ instance containing a list of objects of the ‘StatModel’ class as obtained by the ‘fitDTU’ function of the ‘satuRn’ package.

contrasts
  ‘numeric’ matrix specifying one or more contrasts of the linear model coefficients to be tested. The rownames of the matrix should be equal to the names of parameters of the model that are involved in the contrast. The column names of the matrix will be used to construct names to store the results in the rowData of the SummarizedExperiment.

diagplot1
  ‘boolean(1)’ Logical, defaults to TRUE. If set to TRUE, a plot of the histogram of the z-scores (computed from p-values) is displayed using the locfdr function of the ‘locfdr’ package. The blue dashed curve is fitted to the mid 50 to originate from null transcripts, thus representing the estimated empirical null component
densities. The maximum likelihood estimates (MLE) and central matching estimates (CME) of this estimated empirical null distribution are given below the plot. If the values for delta and sigma deviate from 0 and 1 respectively, the downstream inference will be influenced by the empirical adjustment implemented in satuRn.

diagplot2 'boolean(1)' Logical, defaults to TRUE. If set to TRUE, a plot of the histogram of the "empirically adjusted" test statistics and the standard normal distribution will be displayed. Ideally, the majority (mid portion) of the adjusted test statistics should follow the standard normal.

sort 'boolean(1)' Logical, defaults to FALSE. If set to TRUE, the output of the topTable test function is sorted according to the empirical p-values.

Value

An updated ‘SummarizedExperiment’ that contains the ‘Dataframes’ that display the significance of DTU for each transcript in each contrast of interest.

Author(s)

Jeroen Gilis

Examples

data(sumExp_example, package = "satuRn")
library(SummarizedExperiment)
sumExp <- fitDTU(
    object = sumExp_example,
    formula = ~ 0 + group,
    parallel = FALSE,
    BPPARAM = BiocParallel::bpparam(),
    verbose = TRUE
)
group <- as.factor(colData(sumExp)$group)
design <- model.matrix(~ 0 + group)
colnames(design) <- levels(group)
L <- matrix(0, ncol = 2, nrow = ncol(design))
rownames(L) <- colnames(design)
colnames(L) <- c("Contrast1", "Contrast2")
L[c("VISp.L5.IT_VISp_Hsd11b1_Endou",
    "ALM.L5.IT_ALM_Tnc"), 1] <- c(1, -1)
L[c("VISp.L5.IT_VISp_Hsd11b1_Endou",
    "ALM.L5.IT_ALM_Tmem163_Dmrtb1"), 2] <- c(1, -1)
sumExp <- testDTU(object = sumExp,
    contrasts = L,
    diagplot1 = FALSE,
    diagplot2 = FALSE,
    sort = FALSE)
Index

* datasets
  sumExp_example, 8
  Tasic_counts_vignette, 9
  Tasic_metadata_vignette, 10
  .StatModel (StatModel-class), 8

fitDTU, 2
fitDTU, SummarizedExperiment-method
  (fitDTU), 2

getCoef (getModel, StatModel-method), 3
getCoef, StatModel-method
  (getModel, StatModel-method), 3
getDF (getModel, StatModel-method), 3
getDF, StatModel-method
  (getModel, StatModel-method), 3
getDfPosterior
  (getModel, StatModel-method), 3
getDfPosterior, StatModel-method
  (getModel, StatModel-method), 3
getDispersion
  (getModel, StatModel-method), 3
getDispersion, StatModel-method
  (getModel, StatModel-method), 3
getModel (getModel, StatModel-method), 3
getModel, StatModel-method, 3

plotDTU, 5

StatModel, 7
StatModel-class, 8
statModelAccessors
  (getModel, StatModel-method), 3
sumExp_example, 8

Tasic_counts_vignette, 9
Tasic_metadata_vignette, 10
testDTU, 10