Package 'DRIMSeq'

October 12, 2016

Type Package

Title Differential splicing and sQTL analyses with Dirichlet-multinomial model in RNA-Seq

Version 1.0.2

Date 2016-03-29

Description The package provides two frameworks. One for the differential splicing analysis between different conditions and one for the sQTL analysis. Both are based on modeling the counts of genomic features (i.e., transcripts, exons or exonic bins) with Dirichlet-multinomial distribution. The package also makes available functions for visualization and exploration of the data and results.

biocViews SNP, AlternativeSplicing, DifferentialSplicing, Genetics, RNASeq, Sequencing, WorkflowStep, MultipleComparison, GeneExpression, DifferentialExpression

License GPL (>= 3)

Depends R (>= 3.3.0)

Imports GenomicRanges, IRanges, S4Vectors, BiocGenerics, methods, BiocParallel, edgeR, utils, stats, grDevices, ggplot2, reshape2

Suggests PasillaTranscriptExpr, GeuvadisTranscriptExpr, grid, BiocStyle, knitr, testthat

LazyData true

ByteCompile false

VignetteBuilder knitr

Collate 'DRIMSeq.R' 'class_show_utils.R' 'class_MatrixList.R' 'class_dmDSdata.R' 'class_dmDSdispersion.R' 'class_dmDSfit.R' 'class_dmDStest.R' 'class_dmSQTLdata.R' 'class_dmSQTLdispersion.R' 'class_dmSQTLfit.R' 'class_dmSQTLtest.R' 'dmDS_adjustmentCommon.R' 'dmDS_estimateCommonDispersion.R' 'dmDS_estimateTagwiseDispersion.R' 'dmDS_filter.R' 'dmDS_fitOneModel.R' 'dmDS_plotFit.R' 'dmDS_profileLikCommon.R' 'dmDS_test.R' 'dmSQTL_adjustmentCommon.R'

'dmSQTL_estimateCommonDispersion.R'
'dmSQTL_estimateTagwiseDispersion.R' 'dmSQTL_filter.R'
'dmSQTL_fitOneModel.R' 'dmSQTL_plotFit.R'
'dmSQTL_profileLikCommon.R' 'dmSQTL_test.R' 'dm_Hessian.R'
'dm_NewtonRaphson.R' 'dm_adjustmentOneGeneManyGroups.R'
'dm_adjustmentOneGeneOneGroup.R' 'dm_colorb.R' 'dm_deviance.R'
'dm_estimateMeanExpression.R' 'dm_fitOneGeneManyGroups.R'
'dm_fitOneGeneOneGroup.R' 'dm_lik.R' 'dm_plotData.R'
'dm_plotDispersion.R' 'dm_plotProportions.R' 'dm_plotPvalues.R'
'dm_profileLikTagwise.R' 'dm_rdirichlet.R' 'dm_score.R'
'dm_weirMoM.R'

RoxygenNote 5.0.1

NeedsCompilation no

Author Malgorzata Nowicka [aut, cre]

Maintainer Malgorzata Nowicka <gosia.nowicka@uzh.ch>

R topics documented:

data_dmDSdata *Sample data for differential splicing analysis*

Description

We use a subset of kallisto transcript counts from PasillaTranscriptExpr package.

Usage

```
data_dmDSdata
```

Format

data_dmDSdata is a dmDSdata object. See Examples.

Value

data_dmDSdata

Source

Brooks AN, Yang L, Duff MO, et al. Conservation of an RNA regulatory map between Drosophila and mammals. Genome Res. 2011;21(2):193-202

PasillaTranscriptExpr package

Examples

```
############################
### Create dmDSdata object
############################
### Get kallisto transcript counts from 'PasillaTranscriptExpr' package

library(PasillaTranscriptExpr)

data_dir  <- system.file("extdata", package = "PasillaTranscriptExpr")

metadata <- read.table(file.path(data_dir, "metadata.txt"),
 header = TRUE, as.is = TRUE)
metadata

counts <- read.table(file.path(data_dir, "counts.txt"),
 header = TRUE, as.is = TRUE)
head(counts)

# Create a dmDSdata object
d <- dmDSdata(counts = counts[, metadata$SampleName],
 gene_id = counts$gene_id, feature_id = counts$feature_id,
 sample_id = metadata$SampleName, group = metadata$condition)
```

```
plotData(d)

# Use a subset of genes, which is defined in the following file
gene_id_subset <- readLines(file.path(data_dir, "gene_id_subset.txt"))
d <- d[names(d) %in% gene_id_subset, ]

plotData(d)

data_dmDSdata <- d

###################################
### Differential splicing analysis
###################################
# If possible, use BPPARAM = BiocParallel::MulticoreParam() with more workers

d <- data_dmDSdata

head(counts(d))
samples(d)
head(names(d))
length(d)
d[1:20, ]
d[1:20, 1:3]

### Filtering
# Check what is the minimal number of replicates per condition
table(samples(d)$group)

d <- dmFilter(d, min_samps_gene_expr = 7, min_samps_feature_expr = 3,
 min_samps_feature_prop = 0)
plotData(d)

### Calculate dispersion
d <- dmDispersion(d, BPPARAM = BiocParallel::SerialParam())
plotDispersion(d)

head(mean_expression(d))
common_dispersion(d)
head(genewise_dispersion(d))

### Fit full model proportions
d <- dmFit(d, BPPARAM = BiocParallel::SerialParam())

head(proportions(d))
head(statistics(d))

### Fit null model proportions and test for DS
d <- dmTest(d, BPPARAM = BiocParallel::SerialParam())
plotTest(d)

head(proportions(d))
head(statistics(d))
```

```
head(results(d))

### Plot feature proportions for top DS gene
res <- results(d)
res <- res[order(res$pvalue, decreasing = FALSE), ]

gene_id <- res$gene_id[1]

plotFit(d, gene_id = gene_id)
plotFit(d, gene_id = gene_id, plot_type = "lineplot")
plotFit(d, gene_id = gene_id, plot_type = "ribbonplot")
```

`data_dmSQTLdata` *Sample data for sQTL analysis*

Description

A subset of data from GEUVADIS project where 462 RNA-Seq samples from lymphoblastoid cell lines were obtained. The genome sequencing data of the same individuals is provided by the 1000 Genomes Project. The samples in this project come from five populations: CEPH (CEU), Finns (FIN), British (GBR), Toscani (TSI) and Yoruba (YRI). Here, we use a subset of CEPH data from chromosome 19 available in `GeuvadisTranscriptExpr` package.

Usage

```
data_dmSQTLdata
```

Format

`data_dmSQTLdata` is a `dmSQTLdata` object. See Examples.

Value

`data_dmSQTLdata`

Source

Lappalainen T, Sammeth M, Friedlander MR, et al. Transcriptome and genome sequencing uncovers functional variation in humans. Nature. 2013;501(7468):506-11

`GeuvadisTranscriptExpr` package

Examples

```
############################
### Create dmSQTLdata object
############################
```

```
# Use subsets of data defined in GeuvadisTranscriptExpr package
library(GeuvadisTranscriptExpr)

counts <- GeuvadisTranscriptExpr::counts
genotypes <- GeuvadisTranscriptExpr::genotypes
gene_ranges <- GeuvadisTranscriptExpr::gene_ranges
snp_ranges <- GeuvadisTranscriptExpr::snp_ranges

# Make sure that samples in counts and genotypes are in the same order
sample_id <- colnames(counts[, -(1:2)])

d <- dmSQTLdataFromRanges(counts = counts[, sample_id],
   gene_id = counts$Gene_Symbol, feature_id = counts$TargetID,
   gene_ranges = gene_ranges, genotypes = genotypes[, sample_id],
   snp_id = genotypes$snpId, snp_ranges = snp_ranges, sample_id = sample_id,
   window = 5e3, BPPARAM = BiocParallel::SerialParam())

plotData(d)

data_dmSQTLdata <- d

##############################
### sQTL analysis
##############################
# If possible, use BPPARAM = BiocParallel::MulticoreParam() with more workers

d <- data_dmSQTLdata

head(names(d))
length(d)
d[1:10, ]
d[1:10, 1:10]

### Filtering
d <- dmFilter(d, min_samps_gene_expr = 70, min_samps_feature_expr = 5,
   min_samps_feature_prop = 0, minor_allele_freq = 5,
   BPPARAM = BiocParallel::SerialParam())
plotData(d)


### Calculate dispersion
d <- dmDispersion(d, BPPARAM = BiocParallel::SerialParam())
plotDispersion(d)

### Fit full model proportions
d <- dmFit(d, BPPARAM = BiocParallel::SerialParam())

### Fit null model proportions and test for sQTLs
d <- dmTest(d, BPPARAM = BiocParallel::SerialParam())
plotTest(d)

head(results(d))
```

```
### Plot feature proportions for top sQTL
res <- results(d)
res <- res[order(res$pvalue, decreasing = FALSE), ]

gene_id <- res$gene_id[1]
snp_id <- res$snp_id[1]

plotFit(d, gene_id, snp_id)
plotFit(d, gene_id, snp_id, plot_type = "boxplot2", order = FALSE)
plotFit(d, gene_id, snp_id, plot_type = "ribbonplot")
```

---

dmDispersion *Estimate dispersions in Dirichlet-multinomial model*

Description

Maximum likelihood estimates of dispersion parameters in the Dirichlet-multinomial model used in differential splicing or sQTL analysis.

Usage

```
dmDispersion(x, ...)

## S4 method for signature 'dmDSdata'
dmDispersion(x, mean_expression = TRUE,
  common_dispersion = TRUE, genewise_dispersion = TRUE,
  disp_adjust = TRUE, disp_mode = "grid", disp_interval = c(0, 1e+05),
  disp_tol = 1e-08, disp_init = 100, disp_init_weirMoM = TRUE,
  disp_grid_length = 21, disp_grid_range = c(-10, 10),
  disp_moderation = "common", disp_prior_df = 0.1, disp_span = 0.3,
  prop_mode = "constrOptimG", prop_tol = 1e-12, verbose = 0,
  BPPARAM = BiocParallel::MulticoreParam(workers = 1))

## S4 method for signature 'dmSQTLdata'
dmDispersion(x, mean_expression = TRUE,
  common_dispersion = TRUE, genewise_dispersion = TRUE,
  disp_adjust = TRUE, disp_mode = "grid", disp_interval = c(0, 10000),
  disp_tol = 1e-08, disp_init = 100, disp_init_weirMoM = TRUE,
  disp_grid_length = 21, disp_grid_range = c(-10, 10),
  disp_moderation = "none", disp_prior_df = 0.1, disp_span = 0.3,
  prop_mode = "constrOptimG", prop_tol = 1e-12, verbose = 0,
  speed = TRUE, BPPARAM = BiocParallel::MulticoreParam(workers = 1))
```

Arguments

x	dmDSdata or dmSQTLdata object.
...	Other parameters that can be defined by methods using this generic.

`mean_expression`
: Logical. Whether to estimate the mean expression of genes.

`common_dispersion`
: Logical. Whether to estimate the common dispersion.

`genewise_dispersion`
: Logical. Whether to estimate the gene-wise dispersion.

`disp_adjust`
: Logical. Whether to use the Cox-Reid adjusted or non-adjusted profile likelihood.

`disp_mode`
: Optimization method used to maximize the profile likelihood. Possible values are `"optimize"`, `"optim"`, `"constrOptim"`, `"grid"`. See Details.

`disp_interval`
: Numeric vector of length 2 defining the interval of possible values for the dispersion.

`disp_tol`
: The desired accuracy when estimating dispersion.

`disp_init`
: Initial dispersion. If `common_dispersion` is `TRUE`, then `disp_init` is overwritten by common dispersion estimate.

`disp_init_weirMoM`
: Logical. Whether to use the Weir moment estimator as an initial value for dispersion. If `TRUE`, then `disp_init` is replaced by Weir estimates.

`disp_grid_length`
: Length of the search grid.

`disp_grid_range`
: Vector giving the limits of grid interval.

`disp_moderation`
: Dispersion moderation method. One can choose to shrink the dispersion estimates toward the common dispersion (`"common"`) or toward the (dispersion versus mean expression) trend (`"trended"`)

`disp_prior_df`
: Degree of moderation (shrinkage).

`disp_span`
: Value from 0 to 1 defining the percentage of genes used in smoothing sliding window when calculating the dispersion versus mean expression trend.

`prop_mode`
: Optimization method used to estimate proportions. Possible values `"constrOptim"` and `"constrOptimG"`.

`prop_tol`
: The desired accuracy when estimating proportions.

`verbose`
: Numeric. Definie the level of progress messages displayed. 0 - no messages, 1 - main messages, 2 - message for every gene fitting.

`BPPARAM`
: Parallelization method used by `bplapply`.

`speed`
: Logical. If `FALSE`, dispersion is calculated per each gene-block. Such calculation may take a long time, since there can be hundreds of SNPs/blocks per gene. If `TRUE`, there will be only one dipsersion calculated per gene and it will be assigned to all the blocks matched with this gene.

Details

Parameters that are used in the dispersion estimation start with prefix `disp_`, and those that are used for the proportion estimation start with `prop_`.

There are 4 optimization methods implemented within dmDispersion ("optimize", "optim", "constrOptim" and "grid") that can be used to estimate the gene-wise dispersion. Common dispersion is estimated with "optimize".

Arguments that are used by all the methods are:

- disp_adjust
- prop_mode: Both "constrOptim" and "constrOptimG" use constrOptim function to maximize the likelihood of Dirichlet-multinomial proportions. The difference lays in the way the likelihood and score are computed. "constrOptim" uses the likelihood and score that are calculated based on the fact that x*Gamma(x) = Gamma(x+1). In "constrOptimG", we compute them using lgamma function. We recommend using the second approach, since it is much faster than the first one.
- prop_tol: The accuracy for proportions estimation defined as reltol in constrOptim.

Only some of the rest of dispersion parameters in dmDispersion have an influence on the output for a given disp_mode. Here is a list of such active parameters for different modes:

"optimize", which uses optimize to maximize the profile likelihood.

- disp_interval: Passed as interval.
- disp_tol: The accuracy defined as tol.

"optim", which uses optim to maximize the profile likelihood.

- disp_init and disp_init_weirMoM: The initial value par.
- disp_tol: The accuracy defined as factr.

"constrOptim", which uses constrOptim to maximize the profile likelihood.

- disp_init and disp_init_weirMoM: The initial value theta..
- disp_tol: The accuracy defined as reltol.

"grid", which uses the grid approach from edgeR.

- disp_init, disp_grid_length, disp_grid_range: Parameters used to construct the search grid disp_init * 2^seq(from = disp_grid_range[1], to = disp_grid_range[2], length = disp_grid_length).
- disp_moderation: Dipsersion shrinkage is available only with "grid" method.
- disp_prior_df: Used only when dispersion shrinkage is activated. Moderated likelihood is equal to loglik + disp_prior_df * moderation. Higher disp_prior_df, more shrinkage toward common or trended dispersion is applied.
- disp_span: Used only when dispersion moderation toward trend is activated.

Value

Returns a dmDSdispersion or dmSQTLdispersion object.

Author(s)

Malgorzata Nowicka

See Also

data_dmDSdata, data_dmSQTLdata, plotDispersion, dmFit, dmTest

Examples

```
###################################
### Differential splicing analysis
###################################
# If possible, use BPPARAM = BiocParallel::MulticoreParam() with more workers

d <- data_dmDSdata

### Filtering
# Check what is the minimal number of replicates per condition
table(samples(d)$group)
d <- dmFilter(d, min_samps_gene_expr = 7, min_samps_feature_expr = 3,
 min_samps_feature_prop = 0)

### Calculate dispersion
d <- dmDispersion(d, BPPARAM = BiocParallel::SerialParam())
plotDispersion(d)

head(mean_expression(d))
common_dispersion(d)
head(genewise_dispersion(d))
```

---

dmDSdata *Create dmDSdata object Constructor function for a dmDSdata object.*

Description

Create dmDSdata object

Constructor function for a dmDSdata object.

Usage

```
dmDSdata(counts, gene_id, feature_id, sample_id, group)
```

Arguments

counts	Numeric matrix or data frame of counts. Rows represent features, for example, exons, exonic bins or transcripts. Columns represent samples.
gene_id	Vector of gene IDs corresponding to counts.
feature_id	Vector of feature IDs corresponding to counts.
sample_id	Vector of unique sample IDs corresponding to the columns in counts.
group	Vector that defines the grouping of samples.

Value

Returns a dmDSdata object.

Author(s)

Malgorzata Nowicka

See Also

plotData, dmFilter, dmDispersion, dmFit, dmTest

Examples

```
############################
### Create dmDSdata object
############################
### Get kallisto transcript counts from 'PasillaTranscriptExpr' package

library(PasillaTranscriptExpr)

data_dir  <- system.file("extdata", package = "PasillaTranscriptExpr")

metadata <- read.table(file.path(data_dir, "metadata.txt"), header = TRUE,
 as.is = TRUE)
metadata

counts <- read.table(file.path(data_dir, "counts.txt"), header = TRUE,
 as.is = TRUE)
head(counts)

# Create a dmDSdata object
d <- dmDSdata(counts = counts[, metadata$SampleName],
 gene_id = counts$gene_id, feature_id = counts$feature_id,
 sample_id = metadata$SampleName, group = metadata$condition)

plotData(d)

# Use a subset of genes, which is defined in the following file
gene_id_subset <- readLines(file.path(data_dir, "gene_id_subset.txt"))
d <- d[names(d) %in% gene_id_subset, ]
```

```
plotData(d)
```

`dmDSdata-class` *dmDSdata object*

Description

dmDSdata contains expression, in counts, of genomic features such as exons or transcripts and sample information needed for the differential splicing (DS) analysis. It can be created with function `dmDSdata`.

Usage

```
dm_counts(x, ...)

## S4 method for signature 'dmDSdata'
dm_counts(x)

dm_samples(x, ...)

## S4 method for signature 'dmDSdata'
dm_samples(x)

## S4 method for signature 'dmDSdata'
counts(object)

samples(x, ...)

## S4 method for signature 'dmDSdata'
samples(x)

## S4 method for signature 'dmDSdata'
names(x)

## S4 method for signature 'dmDSdata'
length(x)

## S4 method for signature 'dmDSdata,ANY'
x[i, j]
```

Arguments

`...`	Other parameters that can be defined by methods using this generic.
`object, x`	dmDSdata object.
`i, j`	Parameters used for subsetting.

Value

- `counts(object)`: Get a data frame with counts.
- `samples(x)`: Get a data frame with the sample information.
- `names(x)`: Get the gene names.
- `length(x)`: Get the number of genes.
- `x[i, j]`: Get a subset of dmDSdata object that consists of counts for genes i and samples j.
- `dm_counts(object)`: Get the `counts` slot.
- `dm_samples(object)`: Get the `samples` slot.

Slots

`counts` `MatrixList` of expression, in counts, of genomic features. Rows correspond to genomic features, such as exons or transcripts. Columns correspond to samples. MatrixList is partitioned in a way that each of the matrices in a list contains counts for a single gene.

`samples` Data frame with information about samples. It must contain variables: `sample_id` of unique sample names and `group` which groups samples into conditions.

Author(s)

Malgorzata Nowicka

See Also

`data_dmDSdata`, `dmDSdispersion`, `dmDSfit`, `dmDStest`

Examples

```
###################################
### Differential splicing analysis
###################################

d <- data_dmDSdata

head(counts(d))
samples(d)
head(names(d))
length(d)
d[1:20, ]
d[1:20, 1:3]
```

dmDSdispersion-class *dmDSdispersion object*

Description

dmDSdispersion extends the dmDSdata by adding the dispersion estimates of Dirichlet-multinomial distribution used to model the feature (e.g., transcript, exon, exonic bin) coiunts for each gene in the differential splicing analysis. Result of dmDispersion.

Usage

```
mean_expression(x, ...)

## S4 method for signature 'dmDSdispersion'
mean_expression(x)

common_dispersion(x, ...)

## S4 method for signature 'dmDSdispersion'
common_dispersion(x)

common_dispersion(x) <- value

## S4 replacement method for signature 'dmDSdispersion'
common_dispersion(x) <- value

genewise_dispersion(x, ...)

## S4 method for signature 'dmDSdispersion'
genewise_dispersion(x)

genewise_dispersion(x) <- value

## S4 replacement method for signature 'dmDSdispersion'
genewise_dispersion(x) <- value
```

Arguments

x	dmDSdispersion object.
...	Other parameters that can be defined by methods using this generic.
value	Values that replace current attributes.

Value

- `mean_expression(x)`: Get a data frame with mean gene expression.
- `common_dispersion(x)`, `common_dispersion(x) <- value`: Get or set common dispersion. `value` must be numeric of length 1.

- `genewise_dispersion(x)`, `genewise_dispersion(x) <- value`: Get a data frame with gene-wise dispersion or set new gene-wise dispersion. `value` must be a data frame with "gene_id" and "genewise_dispersion" columns.

Slots

`mean_expression` Numeric vector of mean gene expression.

`common_dispersion` Numeric value of estimated common dispersion.

`genewise_dispersion` Numeric vector of estimated gene-wise dispersions.

Author(s)

Malgorzata Nowicka

See Also

`data_dmDSdata`, `dmDSdata`, `dmDSfit`, `dmDStest`

Examples

```
###################################
### Differential splicing analysis
###################################
# If possible, use BPPARAM = BiocParallel::MulticoreParam() with more workers

d <- data_dmDSdata

### Filtering
# Check what is the minimal number of replicates per condition
table(samples(d)$group)
d <- dmFilter(d, min_samps_gene_expr = 7, min_samps_feature_expr = 3,
 min_samps_feature_prop = 0)

### Calculate dispersion
d <- dmDispersion(d, BPPARAM = BiocParallel::SerialParam())
plotDispersion(d)

head(mean_expression(d))
common_dispersion(d)
head(genewise_dispersion(d))
```

dmDSfit-class *dmDSfit object*

Description

dmDSfit extends the dmDSdispersion class by adding the full model Dirichlet-multinomial feature proportion estimates needed for the differential splicing analysis. Feature ratios are estimated for each gene and each condition. Result of dmFit.

Usage

```
proportions(x, ...)

## S4 method for signature 'dmDSfit'
proportions(x)

statistics(x, ...)

## S4 method for signature 'dmDSfit'
statistics(x)
```

Arguments

x	dmDSdispersion object.
...	Other parameters that can be defined by methods using this generic.

Value

- proportions(x): Get a data frame with estimated feature ratios for each condition.
- statistics(x): Get a data frame with maximum log-likelihoods for each condition.

Slots

dispersion Character specifying which type of dispersion was used for fitting: "common_dispersion" or "genewise_dispersion".

fit_full MatrixList containing the per gene feature ratios. Columns correspond to different conditions. Additionally, the full model likelihoods are stored in metadata slot.

Author(s)

Malgorzata Nowicka

See Also

data_dmDSdata, dmDSdata, dmDSdispersion, dmDStest

Examples

```
###################################
### Differential splicing analysis
###################################
# If possible, use BPPARAM = BiocParallel::MulticoreParam() with more workers

d <- data_dmDSdata

### Filtering
# Check what is the minimal number of replicates per condition
table(samples(d)$group)
d <- dmFilter(d, min_samps_gene_expr = 7, min_samps_feature_expr = 3,
 min_samps_feature_prop = 0)

### Calculate dispersion
d <- dmDispersion(d, BPPARAM = BiocParallel::SerialParam())

### Fit full model proportions
d <- dmFit(d, BPPARAM = BiocParallel::SrialParam())

head(proportions(d))
head(statistics(d))
```

---

dmDStest-class *dmDStest object*

Description

dmDStest extends the dmDSfit class by adding the null model Dirichlet-multinomial feature proportion estimates and the results of testing for differential splicing. Proportions are calculated for each gene from pooled (no grouping into conditions) counts. Result of dmTest.

Usage

```
## S4 method for signature 'dmDStest'
proportions(x)

## S4 method for signature 'dmDStest'
statistics(x)

results(x, ...)

## S4 method for signature 'dmDStest'
results(x)
```

Arguments

x	dmDStest object.
...	Other parameters that can be defined by methods using this generic.

Value

- `proportions(x)`: Get a data frame with estimated feature ratios for full model and null models specified in `dmTest` with `compared_groups` parameter.
- `statistics(x)`: Get a data frame with full and null log-likelihoods and degrees of freedom.
- `results(x)`: Get a data frame with results. See Slots.

Slots

`compared_groups` Character vector specifying which groups/conditions should be compared. By default, the comparison is done among all the groups specified by `group` column in `samples(x)`.

`fit_null` `MatrixList`. Contains null proportions, likelihoods and degrees of freedom for a comparison specified with `compared_groups`.

`results` Data frame with `gene_id` - gene IDs, `lr` - likelihood ratio statistics, `df` - degrees of freedom, `pvalue` - p-values and `adj_pvalue` - Benjamini & Hochberg adjusted p-values for comparison specified in `compared_groups`.

Author(s)

Malgorzata Nowicka

See Also

`data_dmDSdata`, `dmDSdata`, `dmDSdispersion`, `dmDSfit`

Examples

```
###################################
### Differential splicing analysis
###################################
# If possible, use BPPARAM = BiocParallel::MulticoreParam() with more workers

d <- data_dmDSdata

### Filtering
# Check what is the minimal number of replicates per condition
table(samples(d)$group)
d <- dmFilter(d, min_samps_gene_expr = 7, min_samps_feature_expr = 3,
 min_samps_feature_prop = 0)

### Calculate dispersion
d <- dmDispersion(d, BPPARAM = BiocParallel::SerialParam())
### Fit full model proportions
d <- dmFit(d, BPPARAM = BiocParallel::SerialParam())
```

```
### Fit null model proportions and test for DS
d <- dmTest(d, BPPARAM = BiocParallel::SerialParam())
plotTest(d)

head(proportions(d))
head(statistics(d))
head(results(d))
```

---

dmFilter *Filtering*

Description

Filtering of genes and features with low expression. Additionally, for the dmSQTLdata object, filtering of genotypes with low frequency.

Usage

```
dmFilter(x, ...)

## S4 method for signature 'dmDSdata'
dmFilter(x, min_samps_gene_expr, min_samps_feature_expr,
  min_samps_feature_prop, min_gene_expr = 10, min_feature_expr = 10,
  min_feature_prop = 0, max_features = Inf)

## S4 method for signature 'dmSQTLdata'
dmFilter(x, min_samps_gene_expr, min_samps_feature_expr,
  min_samps_feature_prop, minor_allele_freq, min_gene_expr = 10,
  min_feature_expr = 10, min_feature_prop = 0, max_features = Inf,
  BPPARAM = BiocParallel::MulticoreParam(workers = 1))
```

Arguments

x	dmDSdata or dmSQTLdata object.
...	Other parameters that can be defined by methods using this generic.
min_samps_gene_expr	Minimal number of samples where genes should be expressed. See Details.
min_samps_feature_expr	Minimal number of samples where features should be expressed. See Details.
min_samps_feature_prop	Minimal number of samples where features should be expressed. See details.
min_gene_expr	Minimal gene expression.

`min_feature_expr`
: Minimal feature expression.

`min_feature_prop`
: Minimal proportion for feature expression. This value should be between 0 and 1.

`max_features`
: Maximum number of features, which pass the filtering criteria, that should be kept for each gene. If equal to `Inf`, all features that pass the filtering criteria are kept.

`minor_allele_freq`
: Minimal number of samples where each of the genotypes has to be present.

`BPPARAM`
: Parallelization method used by `bplapply`.

Details

Filtering parameters should be adjusted according to the sample size of the experiment data and the number of replicates per condition.

`min_samps_gene_expr` defines the minimal number of samples where genes are required to be expressed at the minimal level of `min_gene_expr` in order to be included in the downstream analysis. Ideally, we would like that genes were expressed at some minimal level in all samples because this would lead to good estimates of feature ratios.

Similarly, `min_samps_feature_expr` and `min_samps_feature_prop` defines the minimal number of samples where features are required to be expressed at the minimal levels of counts `min_feature_expr` or proportions `min_feature_prop`. In differential splicing analysis, we suggest using `min_samps_feature_expr` and `min_samps_feature_prop` equal to the minimal number of replicates in any of the conditions. For example, in an assay with 3 versus 5 replicates, we would set these parameters to 3, which allows a situation where a feature is expressed in one condition but may not be expressed at all in another one, which is an example of differential splicing.

By default, we do not use filtering based on feature proportions. Therefore, `min_samps_feature_prop` and `min_feature_prop` equals 0.

In sQTL analysis, usually, we deal with data that has many more replicates than data from a standard differential splicing assay. Our example data set consists of 91 samples. Requiring that genes are expressed in all samples may be too stringent, especially since there may be missing values in the data and for some genes you may not observe counts in all 91 samples. Slightly lower threshold ensures that we do not eliminate such genes. For example, if `min_samps_gene_expr = 70` and `min_gene_expr = 10`, only genes with expression of at least 10 in at least 70 samples are kept. Samples with expression lower than 10 have `NA`s assigned and are skipped in the analysis of this gene. `minor_allele_freq` indicates the minimal number of samples for the minor allele presence. Usually, it is equal to 5% of total samples.

Value

Returns filtered `dmDSdata` or `dmSQTLdata` object.

Author(s)

Malgorzata Nowicka

See Also

data_dmDSdata, data_dmSQTLdata, plotData, dmDispersion, dmFit, dmTest

Examples

```
###################################
### Differential splicing analysis
###################################

d <- data_dmDSdata

### Filtering
# Check what is the minimal number of replicates per condition
table(samples(d)$group)
d <- dmFilter(d, min_samps_gene_expr = 7, min_samps_feature_expr = 3,
 min_samps_feature_prop = 0)
plotData(d)

#############################
### sQTL analysis
#############################
# If possible, use BPPARAM = BiocParallel::MulticoreParam() with more workers

d <- data_dmSQTLdata

### Filtering
d <- dmFilter(d, min_samps_gene_expr = 70, min_samps_feature_expr = 5,
 min_samps_feature_prop = 0, minor_allele_freq = 5,
 BPPARAM = BiocParallel::SerilaParam())
plotData(d)
```

---

dmFit *Estimate proportions in Dirichlet-multinomial model*

Description

Maximum likelihood estimates of genomic feature (for instance, transcript, exon, exonic bin) proportions in full Dirichlet-multinomial model used in differential splicing or sQTL analysis. Full model estimation means that proportions are estimated for every group/condition separately.

Usage

```
dmFit(x, ...)

## S4 method for signature 'dmDSdispersion'
dmFit(x, dispersion = "genewise_dispersion",
  prop_mode = "constrOptimG", prop_tol = 1e-12, verbose = 0,
  BPPARAM = BiocParallel::MulticoreParam(workers = 1))
```

```
## S4 method for signature 'dmSQTLdispersion'
dmFit(x, dispersion = "genewise_dispersion",
  prop_mode = "constrOptimG", prop_tol = 1e-12, verbose = 0,
  BPPARAM = BiocParallel::MulticoreParam(workers = 1))
```

Arguments

x	dmDSdispersion or dmSQTLdispersion object.
...	Other parameters that can be defined by methods using this generic.
dispersion	Character defining which dispersion should be used for fitting. Possible values "genewise_dispersion" or "common_dispersion".
prop_mode	Optimization method used to estimate proportions. Possible values "constrOptim" and "constrOptimG".
prop_tol	The desired accuracy when estimating proportions.
verbose	Numeric. Definie the level of progress messages displayed. 0 - no messages, 1 - main messages, 2 - message for every gene fitting.
BPPARAM	Parallelization method used by bplapply.

Value

Returns a dmDSfit or dmSQTLfit object.

Author(s)

Malgorzata Nowicka

See Also

data_dmDSdata, data_dmSQTLdata, plotFit, dmDispersion, dmTest

Examples

```
###################################
### Differential splicing analysis
###################################
# If possible, use BPPARAM = BiocParallel::MulticoreParam() with more workers

d <- data_dmDSdata

### Filtering
# Check what is the minimal number of replicates per condition
table(samples(d)$group)
d <- dmFilter(d, min_samps_gene_expr = 7, min_samps_feature_expr = 3,
 min_samps_feature_prop = 0)

### Calculate dispersion
d <- dmDispersion(d, BPPARAM = BiocParallel::SerialParam())
```

```
### Fit full model proportions
d <- dmFit(d, BPPARAM = BiocParallel::SerialParam())

head(proportions(d))
head(statistics(d))
```

---

dmSQTLdata *Create dmSQTLdata object*

Description

Constructor functions for a dmSQTLdata object. dmSQTLdata requires that SNPs are already matched to corresponding genes. dmSQTLdataFromRanges does the matching by assigning to a gene all the SNPs that are located in a given surrounding (window) of this gene.

Usage

```
dmSQTLdata(counts, gene_id, feature_id, genotypes, gene_id_genotypes, snp_id,
  sample_id, BPPARAM = BiocParallel::MulticoreParam(workers = 1))

dmSQTLdataFromRanges(counts, gene_id, feature_id, gene_ranges, genotypes,
  snp_id, snp_ranges, sample_id, window = 5000,
  BPPARAM = BiocParallel::MulticoreParam(workers = 1))
```

Arguments

counts	Numeric matrix or data frame of counts. Rows represent features, for example, exons, exonic bins or transcripts. Columns represent samples.
gene_id	Vector of gene IDs corresponding to counts.
feature_id	Vector of feature IDs corresponding to counts.
genotypes	Numeric matrix with genotypes. Rows represent SNPs, columns represent samples. The genotype of each sample is coded in the following way: 0 for ref/ref, 1 for ref/not ref, 2 for not ref/not ref, -1 or NA for missing value.
gene_id_genotypes	Vector of gene IDs corresponding to genotypes.
snp_id	Vector of SNP IDs corresponding to genotypes.
sample_id	Vector of unique sample IDs corresponding to the columns in counts.
BPPARAM	Parallelization method used by bplapply.
gene_ranges	GRanges object with gene location. It must contain gene names when calling names().
snp_ranges	GRanges object with SNP location. It must contain SNP names when calling names().

window — Size of a down and up stream window, which is defining the surrounding for a gene. Only SNPs that are located within a gene or its surrounding are considered in the sQTL analysis.

Details

It is quite common that sample grouping defined by some of the SNPs is identical. Compare `dim(genotypes)` and `dim(unique(genotypes))`. In our sQTL analysis, we do not repeat tests for the SNPs that define the same grouping of samples. Each grouping is tested only once. SNPs that define such unique groupings are aggregated into blocks. P-values and adjusted p-values are estimated at the block level, but the returned results are extended to a SNP level by repeating the block statistics for each SNP that belongs to a given block.

Value

Returns a `dmSQTLdata` object.

Author(s)

Malgorzata Nowicka

See Also

`data_dmSQTLdata`, `dmFilter`, `dmDispersion`, `dmFit`, `dmTest`

Examples

```
############################
### Create dmSQTLdata object
############################

# Use subsets of data defined in GeuvadisTranscriptExpr package
library(GeuvadisTranscriptExpr)

counts <- GeuvadisTranscriptExpr::counts
genotypes <- GeuvadisTranscriptExpr::genotypes
gene_ranges <- GeuvadisTranscriptExpr::gene_ranges
snp_ranges <- GeuvadisTranscriptExpr::snp_ranges

# Make sure that samples in counts and genotypes are in the same order
sample_id <- colnames(counts[, -(1:2)])

d <- dmSQTLdataFromRanges(counts = counts[, sample_id],
   gene_id = counts$Gene_Symbol, feature_id = counts$TargetID,
   gene_ranges = gene_ranges, genotypes = genotypes[, sample_id],
   snp_id = genotypes$snpId, snp_ranges = snp_ranges, sample_id = sample_id,
   window = 5e3, BPPARAM = BiocParallel::SerialParam())

plotData(d)
```

dmSQTLdata-class *dmSQTLdata object*

Description

dmSQTLdata contains genomic feature expression (counts), genotypes and sample information needed for the sQTL analysis. It can be created with function dmSQTLdataFromRanges or dmSQTLdata.

Usage

```
## S4 method for signature 'dmSQTLdata'
names(x)

## S4 method for signature 'dmSQTLdata'
length(x)

## S4 method for signature 'dmSQTLdata,ANY'
x[i, j]
```

Arguments

x	dmSQTLdata object.
i, j	Parameters used for subsetting.

Value

- names(x): Get the gene names.
- length(x): Get the number of genes.
- x[i, j]: Get a subset of dmDSdata object that consists of counts, genotypes and blocks corresponding to genes i and samples j.

Slots

counts MatrixList of expression, in counts, of genomic features. Rows correspond to genomic features, such as exons or transcripts. Columns correspond to samples. MatrixList is partitioned in a way that each of the matrices in a list contains counts for a single gene.

genotypes MatrixList of unique genotypes. Rows correspond to blocks, columns to samples. Each matrix in this list is a collection of unique genotypes that are matched with a given gene.

blocks MatrixList with two columns block_id and snp_id. For each gene, it identifies SNPs with identical genotypes across the samples and assigns them to blocks.

samples Data frame with information about samples. It contains unique sample names sample_id.

Author(s)

Malgorzata Nowicka

See Also

data_dmSQTLdata, dmSQTLdispersion, dmSQTLfit, dmSQTLtest

Examples

```
############################
### sQTL analysis
############################

d <- data_dmSQTLdata

head(names(d))
length(d)
d[1:10, ]
d[1:10, 1:10]
```

---

dmSQTLdispersion-class

dmSQTLdispersion object

Description

dmSQTLdispersion extends the dmSQTLdata by adding the dispersion estimates of Dirichlet-multinomial distribution used to model the feature (e.g., transcript, exon, exonic bin) counts for each gene-SNP pair in the sQTL analysis. Result of dmDispersion.

Slots

`mean_expression` Numeric vector of mean gene expression.

`common_dispersion` Numeric value of estimated common dispersion.

`genewise_dispersion` List of estimated gene-wise dispersions. Each element of this list is a vector of dispersions estimated for all the genotype blocks assigned to a given gene.

Author(s)

Malgorzata Nowicka

See Also

data_dmSQTLdata, dmSQTLdata, dmSQTLfit, dmSQTLtest

dmSQTLfit-class *dmSQTLfit object*

Description

dmSQTLfit extends the dmDSdispersion class by adding the full model Dirichlet-multinomial feature proportion estimates needed for the sQTL analysis. Feature ratios are estimated for each gene and each group that is defined by different SNPs/blocks. Result of dmFit.

Slots

`dispersion` Character specifying which type of dispersion was used for fitting: "common_dispersion" or "genewise_dispersion".

`fit_full` List of MatrixList objects. Each element of this list contains the full model proportion estimates for all the blocks associated with a given gene. Columns of MatrixLists correspond to 3 genotypes (0,1,2). The full model likelihoods are stored in `metadata` slot.

Author(s)

Malgorzata Nowicka

See Also

data_dmSQTLdata, dmSQTLdata, dmSQTLdispersion, dmSQTLtest

dmSQTLtest-class *dmSQTLtest object*

Description

dmSQTLtest extends the dmSQTLfit class by adding the null model Dirichlet-multinomial feature proportion estimates and the results of testing for sQTLs. Proportions are calculated for each gene-block pair from pooled (no grouping into conditions) counts. Result of dmTest.

Usage

```
## S4 method for signature 'dmSQTLtest'
results(x)
```

Arguments

x	dmSQTLtest object.
...	Other parameters that can be defined by methods using this generic.

Value

- `results(x)`: Get a data frame with results. See Slots.

Slots

`fit_null` List of `MatrixList`. Each of them contains null proportions, likelihoods and degrees of freedom for all the blocks (unique SNPs) assigned to a given gene.

`results` Data frame with `gene_id` - gene IDs, `block_id` - block IDs, `snp_id` - SNP IDs, `lr` - likelihood ratio statistics, `df` - degrees of freedom, `pvalue` - p-values and `adj_pvalue` - Benjamini & Hochberg adjusted p-values.

Author(s)

Malgorzata Nowicka

See Also

`data_dmSQTLdata`, `dmSQTLdata`, `dmSQTLdispersion`, `dmSQTLfit`

Examples

```
############################
### sQTL analysis
############################
# If possible, use BPPARAM = BiocParallel::MulticoreParam() with more workers

d <- data_dmSQTLdata

### Filtering
d <- dmFilter(d, min_samps_gene_expr = 70, min_samps_feature_expr = 5,
   min_samps_feature_prop = 0, minor_allele_freq = 5,
   BPPARAM = BiocParallel::SerialParam())

### Calculate dispersion
d <- dmDispersion(d, BPPARAM = BiocParallel::SerialParam())

### Fit full model proportions
d <- dmFit(d, BPPARAM = BiocParallel::SerialParam())

### Fit null model proportions and test for sQTLs
d <- dmTest(d, BPPARAM = BiocParallel::SerialParam())
plotTest(d)

head(results(d))
```

dmTest *Likelihood ratio test*

Description

First, estimate the null Dirichlet-multinomial model proportions, i.e., feature ratios are estimated based on pooled (no grouping into conditions) counts. Use the likelihood ratio statistic to test for the difference between feature proportions in different groups to identify the differentially spliced genes (differential splicing analysis) or the sQTLs (sQTL analysis).

Usage

```
dmTest(x, ...)

## S4 method for signature 'dmDSfit'
dmTest(x, compared_groups = levels(samples(x)$group),
  prop_mode = "constrOptimG", prop_tol = 1e-12, verbose = 0,
  BPPARAM = BiocParallel::MulticoreParam(workers = 1))

## S4 method for signature 'dmSQTLfit'
dmTest(x, prop_mode = "constrOptimG",
  prop_tol = 1e-12, verbose = 0,
  BPPARAM = BiocParallel::MulticoreParam(workers = 1))
```

Arguments

x	dmDSfit or dmSQTLfit object.
...	Other parameters that can be defined by methods using this generic.
compared_groups	Vector that defines which experimental conditions should be tested for differential splicing. By default, we test for a difference between any of the groups specified in samples(x)$group. Values in this vector should indicate levels or numbers of levels in samples(x)$group.
prop_mode	Optimization method used to estimate proportions. Possible values "constrOptim" and "constrOptimG".
prop_tol	The desired accuracy when estimating proportions.
verbose	Numeric. Definie the level of progress messages displayed. 0 - no messages, 1 - main messages, 2 - message for every gene fitting.
BPPARAM	Parallelization method used by bplapply.

Value

Returns a dmDStest or dmSQTLtest object.

Author(s)

Malgorzata Nowicka

See Also

data_dmDSdata, data_dmSQTLdata, plotTest, dmDispersion, dmFit

Examples

```
###################################
### Differential splicing analysis
###################################
# If possible, use BPPARAM = BiocParallel::MulticoreParam() with more workers

d <- data_dmDSdata

### Filtering
# Check what is the minimal number of replicates per condition
table(samples(d)$group)
d <- dmFilter(d, min_samps_gene_expr = 7, min_samps_feature_expr = 3,
 min_samps_feature_prop = 0)

### Calculate dispersion
d <- dmDispersion(d, BPPARAM = BiocParallel::SerialParam())

### Fit full model proportions
d <- dmFit(d, BPPARAM = BiocParallel::SerialParam())

### Fit null model proportions and test for DS
d <- dmTest(d, BPPARAM = BiocParallel::SerialParam())
plotTest(d)

head(proportions(d))
head(statistics(d))
head(results(d))
```

---

dm_plotDataDSInfo *Plot the frequency of present features*

Description

Plot the frequency of present features

Usage

```
dm_plotDataDSInfo(info, ds_info)
```

Arguments

info	Data frame with gene_id and feature_id of ALL features
ds_info	Data frame with gene_id and feature_id of ONLY DS features

Value

ggplot object

---

dm_plotProportions *Plot feature proportions*

Description

Plot observed and/or estimated feature proportions.

Usage

```
dm_plotProportions(counts, group, pi_full = NULL, pi_null = NULL,
  main = NULL, plot_type = "boxplot1", order = TRUE)
```

Arguments

counts	Matrix with rows corresponding to features and columns corresponding to samples. Row names are used as labels on the plot.
group	Factor that groups samples into conditions.
pi_full	Matrix of estimated proportions with rows corresponding to features and columns corresponding to conditions defined by factor group. If NULL, nothing is plotted.
pi_null	Matrix of estimated proportions with rows corresponding to features and one column. If NULL, nothing is plotted.
main	Character vector with main title for the plot. If NULL, nothing is plotted.
plot_type	Character defining the type of the plot produced. Possible values "barplot", "boxplot1", "boxplot2", "lineplot", "ribbonplot".
order	Logical. Whether to plot the features ordered by their expression.

Value

ggplot object with the observed and/or estimated with Dirichlet-multinomial model feature ratios. Estimated proportions are marked with diamond shapes.

`MatrixList-class` *MatrixList object*

Description

A MatrixList object is a container for a list of matrices which have the same number of columns but can have varying number of rows. Additionally, one can store an extra information corresponding to each of the matrices in `metadata` matrix.

Usage

```
## S4 method for signature 'MatrixList'
names(x)

## S4 replacement method for signature 'MatrixList'
names(x) <- value

## S4 method for signature 'MatrixList'
rownames(x)

## S4 replacement method for signature 'MatrixList'
rownames(x) <- value

## S4 method for signature 'MatrixList'
colnames(x)

## S4 replacement method for signature 'MatrixList'
colnames(x) <- value

## S4 method for signature 'MatrixList'
length(x)

## S4 method for signature 'MatrixList'
elementNROWS(x)

## S4 method for signature 'MatrixList'
dim(x)

## S4 method for signature 'MatrixList'
nrow(x)

## S4 method for signature 'MatrixList'
ncol(x)

## S4 method for signature 'MatrixList'
x[[i, j]]
```

```
## S4 method for signature 'MatrixList'
x$name

## S4 method for signature 'MatrixList,ANY'
x[i, j]
```

Arguments

`x`	MatrixList object.
`value, i, j, name`	Parameters used for subsetting and assigning new attributes to x.

Value

- `names(x)`, `names(x) <- value`: Get or set names of matrices.
- `rownames(x)`, `rownames(x) <- value`, `colnames(x)`, `colnames(x) <- value`: Get or set row names or column names of unlistData slot.
- `length(x)`: Get the number of matrices in a list.
- `elementNROWS(x)`: Get the number of rows of each of the matrices.
- `dim(x)`, `nrow(x)`, `ncol(x)`: Get the dimensions, number of rows or number of columns of unlistData slot.
- `x[[i]]`, `x[[i, j]]`: Get the matrix i, and optionally, get only columns j of this matrix.
- `x$name`: Shortcut for `x[["name"]]`.
- `x[i, j]`: Get a subset of MatrixList that consists of matrices i with columns j.

Slots

`unlistData`	Matrix which is a row binding of all the matrices in a list.
`partitioning`	List of indexes which defines the row partitioning of unlistData matrix into the original matrices.
`metadata`	Matrix of additional information where each row corresponds to one of the matrices in a list.

Author(s)

Malgorzata Nowicka

plotData *Plot data summary*

Description

Plot data summary

Usage

```
plotData(x, ...)

## S4 method for signature 'dmDSdata'
plotData(x, out_dir = NULL)

## S4 method for signature 'dmSQTLdata'
plotData(x, out_dir = NULL)
```

Arguments

x	dmDSdata or dmSQTLdata object.
...	Other parameters that can be defined by methods using this generic.
out_dir	Character string that is used to save the plot in paste0(out_dir, plot_name, ".pdf") file. plot_name depends on type of a plot produced, for example, plot_name = "hist_features" for histogram with number of features per gene. If NULL, the plot is returned as ggplot object and can be further modified, for example, using theme().

Value

Plot a histogram of the number of features per gene. Additionally, for dmSQTLdata object, plot a histogram of the number of SNPs per gene and a histogram of the number of unique SNPs (blocks) per gene.

Author(s)

Malgorzata Nowicka

See Also

data_dmDSdata, data_dmSQTLdata, plotDispersion, plotFit, plotTest

Examples

```
###################################
### Differential splicing analysis
###################################

d <- data_dmDSdata
plotData(d)
```

```
############################
### sQTL analysis
############################

d <- data_dmSQTLdata
plotData(d)
```

`plotDispersion` *Dispersion versus mean expression plot*

Description

Dispersion versus mean expression plot

Usage

```
plotDispersion(x, ...)

## S4 method for signature 'dmDSdispersion'
plotDispersion(x, out_dir = NULL)

## S4 method for signature 'dmSQTLdispersion'
plotDispersion(x, out_dir = NULL)
```

Arguments

`x`	`dmDSdispersion` or `dmSQTLdispersion` object.
`...`	Other parameters that can be defined by methods using this generic.
`out_dir`	Character string that is used to save the plot in `paste0(out_dir, plot_name, ".pdf")` file. `plot_name` depends on type of a plot produced, for example, `plot_name = "hist_features"` for histogram with number of features per gene. If `NULL`, the plot is returned as `ggplot` object and can be further modified, for example, using `theme()`.

Value

Scatterplot of Dirichlet-multinomial gene-wise dispersion versus mean gene expression. Both variables are scaled with log10. One dot in the plot corresponds to a gene.

Author(s)

Malgorzata Nowicka

See Also

`data_dmDSdata`, `data_dmSQTLdata`, `plotData`, `plotFit`, `plotTest`

Examples

```
###################################
### Differential splicing analysis
###################################
# If possible, use BPPARAM = BiocParallel::MulticoreParam() with more workers

d <- data_dmDSdata

### Filtering
# Check what is the minimal number of replicates per condition
table(samples(d)$group)
d <- dmFilter(d, min_samps_gene_expr = 7, min_samps_feature_expr = 3,
 min_samps_feature_prop = 0)

### Calculate dispersion
d <- dmDispersion(d, BPPARAM = BiocParallel::SerialParam())

plotDispersion(d)
```

---

plotFit *Plot feature proportions*

Description

Plot feature proportions

Usage

```
plotFit(x, ...)

## S4 method for signature 'dmDSfit'
plotFit(x, gene_id, plot_type = "barplot", order = TRUE,
  plot_full = TRUE, plot_main = TRUE, out_dir = NULL)

## S4 method for signature 'dmDStest'
plotFit(x, gene_id, plot_type = "barplot",
  order = TRUE, plot_full = TRUE, plot_null = TRUE, plot_main = TRUE,
  out_dir = NULL)

## S4 method for signature 'dmSQTLfit'
plotFit(x, gene_id, snp_id, plot_type = "boxplot1",
  order = TRUE, plot_full = TRUE, plot_main = TRUE, out_dir = NULL)

## S4 method for signature 'dmSQTLtest'
plotFit(x, gene_id, snp_id, plot_type = "boxplot1",
```

```
  order = TRUE, plot_full = TRUE, plot_null = TRUE, plot_main = TRUE,
  out_dir = NULL)
```

Arguments

x	dmDSfit, dmDStest or dmSQTLfit, dmSQTLtest object.
...	Other parameters that can be defined by methods using this generic.
gene_id	Character indicating a gene ID to be plotted.
plot_type	Character defining the type of the plot produced. Possible values "barplot", "boxplot1", "boxplot2", "lineplot", "ribbonplot".
order	Logical. Whether to plot the features ordered by their expression.
plot_full	Logical. Whether to plot the proportions estimated by the full model.
plot_main	Logical. Whether to plot a title with the information about the Dirichlet-multinomial estimates.
out_dir	Character string that is used to save the plot in paste0(out_dir, plot_name, ".pdf") file. plot_name depends on type of a plot produced, for example, plot_name = "hist_features" for histogram with number of features per gene. If NULL, the plot is returned as ggplot object and can be further modified, for example, using theme().
plot_null	Logical. Whether to plot the proportions estimated by the null model.
snp_id	Character indicating a SNP ID to be plotted. snp_id must match gene_id.

Value

Plot, per gene, the observed and estimated with Dirichlet-multinomial model feature ratios. Estimated proportions are marked with diamond shapes.

Author(s)

Malgorzata Nowicka

See Also

data_dmDSdata, data_dmSQTLdata, plotData, plotDispersion, plotTest

Examples

```
###################################
### Differential splicing analysis
###################################
# If possible, use BPPARAM = BiocParallel::MulticoreParam() with more workers

d <- data_dmDSdata

### Filtering
# Check what is the minimal number of replicates per condition
table(samples(d)$group)
d <- dmFilter(d, min_samps_gene_expr = 7, min_samps_feature_expr = 3,
```

```
 min_samps_feature_prop = 0)

### Calculate dispersion
d <- dmDispersion(d, BPPARAM = BiocParallel::SerialParam())

### Fit full model proportions
d <- dmFit(d, BPPARAM = BiocParallel::SerialParam())

### Fit null model proportions and test for DS
d <- dmTest(d, BPPARAM = BiocParallel::SerialParam())

### Plot feature proportions for top DS gene
res <- results(d)
res <- res[order(res$pvalue, decreasing = FALSE), ]

gene_id <- res$gene_id[1]

plotFit(d, gene_id = gene_id)
plotFit(d, gene_id = gene_id, plot_type = "lineplot")
plotFit(d, gene_id = gene_id, plot_type = "ribbonplot")
```

`plotTest` *Plot p-values distribution*

Description

Plot p-values distribution

Usage

```
plotTest(x, ...)

## S4 method for signature 'dmDStest'
plotTest(x, out_dir = NULL)

## S4 method for signature 'dmSQTLtest'
plotTest(x, out_dir = NULL)
```

Arguments

`x`	`dmDStest` or `dmSQTLtest` object.
`...`	Other parameters that can be defined by methods using this generic.
`out_dir`	Character string that is used to save the plot in `paste0(out_dir, plot_name, ".pdf")` file. `plot_name` depends on type of a plot produced, for example, `plot_name = "hist_features"` for histogram with number of features per gene. If `NULL`, the plot is returned as `ggplot` object and can be further modified, for example, using `theme()`.

Value

Plot a histogram of p-values.

Author(s)

Malgorzata Nowicka

See Also

data_dmDSdata, data_dmSQTLdata, plotData, plotDispersion, plotFit

Examples

```
###################################
### Differential splicing analysis
###################################
# If possible, use BPPARAM = BiocParallel::MulticoreParam() with more workers

d <- data_dmDSdata

### Filtering
# Check what is the minimal number of replicates per condition
table(samples(d)$group)
d <- dmFilter(d, min_samps_gene_expr = 7, min_samps_feature_expr = 3,
 min_samps_feature_prop = 0)

### Calculate dispersion
d <- dmDispersion(d, BPPARAM = BiocParallel::SerialParam())

### Fit full model proportions
d <- dmFit(d, BPPARAM = BiocParallel::SerialParam())

### Fit null model proportions and test for DS
d <- dmTest(d, BPPARAM = BiocParallel::SerialParam())

plotTest(d)
```

Index