Package ‘MPRAnalyze’
May 30, 2024

Type Package
Title Statistical Analysis of MPRA data
Version 1.22.0
Author Tal Ashuach [aut, cre], David S Fischer [aut], Anat Kriemer [ctb], Fabian J Theis [ctb], Nir Yosef [ctb],
Maintainer Tal Ashuach <tal_ashuach@berkeley.edu>
Description MPRAnalyze provides statistical framework for the analysis of data generated by Massively Parallel Reporter Assays (MPRAs), used to directly measure enhancer activity. MPRAnalyze can be used for quantification of enhancer activity, classification of active enhancers and comparative analyses of enhancer activity between conditions. MPRAnalyze construct a nested pair of generalized linear models (GLMs) to relate the DNA and RNA observations, easily adjustable to various experimental designs and conditions, and provides a set of rigorous statistical testing schemes.
License GPL-3
Encoding UTF-8
Imports BiocParallel, methods, progress, stats, SummarizedExperiment
biocViews ImmunoOncology, Software, StatisticalMethod, Sequencing, GeneExpression, CellBiology, CellBasedAssays, DifferentialExpression, ExperimentalDesign, Classification
Suggests knitr
ByteCompile true
BugReports https://github.com/YosefLab/MPRAnalyze
URL https://github.com/YosefLab/MPRAnalyze
RoxygenNote 7.1.1
VignetteBuilder knitr
git_url https://git.bioconductor.org/packages/MPRAnalyze
git_branch RELEASE_3_19
git_last_commit 12abab2
git_last_commit_date 2024-04-30
Repository Bioconductor 3.19
Date/Publication 2024-05-29
analyzeComparative

Run a comparative analysis between conditions

Usage

```r
analyzeComparative(
  obj,
  rnaDesign,
  dnaDesign = NULL,
  fit.se = FALSE,
  reducedDesign = NULL,
  correctControls = TRUE,
  verbose = TRUE,
  mode = "classic",
  BPPARAM = NULL
)
```

Arguments

- `obj` the MpraObject
- `rnaDesign` the design for the RNA model.
- `dnaDesign` the design for the DNA model. Only terms that are matched with the RNA design should be included.
analyzeQuantification

Perform quantitative analysis on the MPRA data. This analysis aims to determine which sequences have a regulatory function, when no condition is being tested.

Description

- empirical: the model is fitted as specified, enabling future empirical testing (either empirical p-value if negative controls are provided, or a global devience analysis, see details in 'test.empirical')
- lrt: only available if negative controls are provided. A likelihood ratio test is used, with the null hypothesis a joint model of the controls and a given candidate sequence, and the alternative model being a separate model for controls and candidates.

Examples

data <- simulateMPRA(tr = rep(2,5), da=c(rep(0,2), rep(1,3)),
nbatch=2, nbc=15)
obj <- MpraObject(dnaCounts = data$obs.dna,
rnaCounts = data$obs.rna,
colAnnot = data$annot)
obj <- estimateDepthFactors(obj, lib.factor = "batch", which.lib = "both")
## run an LRT-based analysis, as recommended:
obj <- analyzeComparative(obj, dnaDesign = ~ batch + barcode + condition,
rnaDesign = ~ condition, reducedDesign = ~ 1)

## alternatively, run a coefficient-based analysis:
obj <- analyzeComparative(obj, dnaDesign = ~ batch + barcode + condition,
rnaDesign = ~ condition, fit.se = TRUE)
Usage

analyzeQuantification(obj, dnaDesign = ~1, rnaDesign = ~1, BPPARAM = NULL)

Arguments

obj the MpraObject
dnaDesign the design of the DNA counts
rnaDesign the design of the RNA counts
BPPARAM a parallelization object created by BiocParallel. This overwrites the BPPARAM object set in the object creation.

Value
the MpraObject, with populated models

Examples

data <- simulateMPRA(tr = rep(2,10), nbatch=2, nbc=15)
obj <- MpraObject(dnaCounts = data$obs.dna,
rnaCounts = data$obs.rna,
colAnnot = data$annot)
obj <- estimateDepthFactors(obj, lib.factor = "batch", which.lib = "both")
obj <- analyzeQuantification(obj, dnaDesign = ~ batch + barcode,
rnaDesign = ~1)

---

ChrEpi  Sample MPRA data

Description

A subset of MPRA data from Inoue et al., comparing enhancer activity of episomal constructs vs. chromosomally integrated constructs (integration was performed with a lentivirus). Data included negative control enhancers, multiple batches and barcodes, a subsample of which are included in this sample data for runtime purposes.

Usage

data(ChrEpi)
ce.colAnnot
ce.dnaCounts
ce.rnaCounts
ce.control
estimateDepthFactors

Format

- **ce.colAnnot**: Column annotations for each column (sample) in the data matrices
  - **batch**: batch identifier, factor
  - **condition**: condition identifier, factor. WT corresponds to chromosomal and MT corresponds to episomal
  - **barcode**: barcode identifier, factor
- **ce.dnaCounts**: DNA observations
- **ce.rnaCounts**: DNA observations
- **ce.control**: indices of control enhancers
  - An object of class `data.frame` with 40 rows and 3 columns.
  - An object of class `matrix` with 110 rows and 40 columns.
  - An object of class `matrix` with 110 rows and 40 columns.
  - An object of class `logical` of length 110.

Source

[https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5204343/](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5204343/)

**estimateDepthFactors**  
*estimate library size correction factors*

Description

estimate library size correction factors

Usage

```r
estimateDepthFactors(
  obj,
  lib.factor = NULL,
  which.lib = "both",
  depth.estimator = "uq"
)
```

Arguments

- **obj**: the MpraObject
- **lib.factor**: the factor associating each sample to a library. Can be a factor or the name of a column in the object’s colAnnot. If not provided, the data is assumed to have been generated from a single library, and constant library depth is set.
- **which.lib**: which library to compute the depth factors for. Options are "both" (default), "dna" or "rna". If the DNA and RNA counts have different library factors, this function should be called twice: once with "dna" and once with "rna"
getAlpha

a character indicating which depth estimation to use, or a function to perform
the estimation. Currently supported values are "uq" for upper quantile of non-
zero values (default), "rle" for RLE (uses geometric mean, and is therefore not
recommended if libraries have 0 counts), or "totsum" for total sum. For a func-
tion input: function should take a numeric vector and return a single numeric,
and preferably handle NA values. See examples.

Value
the MpraObject with estimated values for sequencing depth factors

Note
since in most MPRA experiments multiple barcodes exist within a single library, each column in the
matrix is usually not a separate library. For this reason, it is recommended to supply this function
with the appropriate partitioning of the data matrix columns into libraries, see lib.factor

Examples

data <- simulateMPRA(tr = rep(2,10), da=NULL, nbatch=2, nbc=20)
obj <- MpraObject(dnaCounts = data$obs.dna,
rnaCounts = data$obs.rna,
colAnnot = data$annot)
obj <- estimateDepthFactors(ojb, lib.factor = "batch", which.lib = "both")
## Upper quantile, using a higher quantile than 0.75:
obj <- estimateDepthFactors(ojb, lib.factor = "batch", which.lib = "both",
depth.estimator = function(x) quantile(x, .95,
na.rm=TRUE))

getAlpha

return the fitted value for the transcription rate.

Description
return the fitted value for the transcription rate.

Usage
getAlpha(obj, by.factor = NULL, full = TRUE)

Arguments
obj
the MpraObject to extract from, must be after model fitting
by.factor
return a matrix of values, corresponding to the estimated rates of transcription
under different values of a factor included in the design. Value must be of these
options: NULL: (default) return only the intercept term, a single baseline rate
for each enhancer "all": will return the corresponding transcription rates for all
values included in the model factor name: must be a factor included in the RNA annotations and the rna design. Will return the corresponding rates for all values of the given factor

full if true, return rate of the full model (default), otherwise of the reduced model (only applies if an LRT-based analysis was used)

Value

the estimate for transcription rate as fitted by the model

Examples

data <- simulateMPRA(tr = rep(2,10), da=c(rep(0,5), rep(1,5)),
nbatch=2, nbc=15)
obj <- MpraObject(dnaCounts = data$obs.dna,
rnaCounts = data$obs.rna,
colAnnot = data$annot)
obj <- estimateDepthFactors(obj, lib.factor = "batch", which.lib = "both")
obj <- analyzeComparative(obj, dnaDesign = ~ batch + barcode + condition,
rnaDesign = ~ condition, reducedDesign = ~ 1)
## get alpha estimate for the two conditions
alpha <- getAlpha(obj, by.factor="condition")

getDistrParam_DNA

Get model distribution parameters from an MpraObject of a given candidate enhancer

Description

Get model distribution parameters from an MpraObject of a given candidate enhancer

Usage

getDistrParam_DNA(obj, enhancer, full = TRUE)
getDistrParam_RNA(obj, enhancer = NULL, full = TRUE)

Arguments

obj MpraObject to extract from
enhancer enhancer to extract
full whether to extract from full model

Value

fit parameters (numeric, samples x parameters)
Examples

```r
data <- simulateMPRA(tr = rep(2,5), nbatch=2, nbc=15)
obj <- MpraObject(dnaCounts = data$obs.dna,
                  rnaCounts = data$obs.rna,
                  colAnnot = data$annot)
obj <- estimateDepthFactors(obj, lib.factor = "batch", which.lib = "both")
obj <- analyzeQuantification(obj, dnaDesign = ~ batch + barcode,
                             rnaDesign = ~1)
## get distributional parameters of the first enhancer:
dist.params.dna <- getDistrParam_DNA(obj, 1)
dist.params.rna <- getDistrParam_RNA(obj, 1)
```

getFits_DNA

Get DNA model-based estimates from an MpraObject (the expected values based on the model). These can be compared with the observed counts to assess goodness of fit.

Description

Get DNA model-based estimates from an MpraObject (the expected values based on the model). These can be compared with the observed counts to assess goodness of fit.

Usage

```r
getFits_DNA(
  obj,
  enhancers = NULL,
  depth = TRUE,
  full = TRUE,
  transition = FALSE
)
```

Arguments

- `obj`: MpraObject to extract from
- `enhancers`: which enhancers to get the fits for. Can be character vectors with enhancer names, logical or numeric enhancer indices, or `NULL` if all enhancers are to be extracted (default)
- `depth`: include depth correction in the model fitting (default `TRUE`)
- `full`: if LRT modeling was used, `TRUE` (default) would return the fits of the full model, `FALSE` would return the reduced model fits.
- `transition`: use the DNA->RNA transition matrix (default: `FALSE`). This is useful if the DNA observations need to be distributed to match the RNA observations.
getFits_RNA

Description

Get RNA model-based estimates from an MpraObject (the expected values based on the model). These can be compared with the observed counts to assess goodness of fit.

Usage

getFits_RNA(obj, enhancers = NULL, depth = TRUE, full = TRUE, rnascale = TRUE)

Arguments

- `obj` MpraObject to extract from
- `enhancers` which enhancers to get the fits for. Can be character vectors with enhancer names, logical or numeric enhancer indices, or NULL if all enhancers are to be extracted (default)
- `depth` include depth correction in the model fitting (default TRUE)
- `full` if LRT modeling was used, TRUE (default) would return the fits of the full model, FALSE would return the reduced model fits.
- `rnascale` if controls were used to correct the fitting (in comparative analyses), use these factors to re-adjust the estimates back.

Value

RNA fits (numeric, enhancers x samples)
**Examples**

```r
data <- simulateMPRA(tr = rep(2,5), nbatch=2, nbc=15)
obj <- MpraObject(dnaCounts = data$obs.dna,
                   rnaCounts = data$obs.rna,
                   colAnnot = data$annot)
obj <- estimateDepthFactors(obj, lib.factor = "batch", which.lib = "both")
obj <- analyzeQuantification(obj, dnaDesign = ~ batch + barcode,
                             rnaDesign = ~1)
RNA.fits <- getFits_RNA(obj)
model.params.dna <- getModelParameters_DNA(obj)
```

**Description**

extract the DNA model parameters

**Usage**

```r
getModelParameters_DNA(obj, features = NULL, full = TRUE)
getModelParameters_RNA(obj, features = NULL, full = TRUE)
```

**Arguments**

- `obj` the MpraObject to extract the parameters from
- `features` the features to extract the parameters from (by default, parameters will be returned for all features)
- `full` if TRUE (default), return the parameters of the full model. Otherwise, return the parameters of the reduced model (only relevant for LRT-based analyses)

**Value**

a data.frame of features (rows) by parameters (cols). By convention, the first parameter is related to the second moment, and the interpretation of it depends on the distributional model used (‘alpha’ for ‘gamma.pois’, variance for ‘ln.nb’ and ‘ln.ln’)

**Examples**

```r
data <- simulateMPRA(tr = rep(2,5), nbatch=2, nbc=15)
obj <- MpraObject(dnaCounts = data$obs.dna,
                   rnaCounts = data$obs.rna,
                   colAnnot = data$annot)
obj <- estimateDepthFactors(obj, lib.factor = "batch", which.lib = "both")
obj <- analyzeQuantification(obj, dnaDesign = ~ batch + barcode,
                             rnaDesign = ~1)
model.params.dna <- getModelParameters_DNA(obj)
```
model.params.rna <- getModelParameters_RNA(obj)

### Description

The main object MPRAnalyze works with, contains the input data, associated annotations, model parameters and analysis results.

### Usage

```r
MpraObject(
  dnaCounts,
  rnaCounts,
  dnaAnnot = NULL,
  rnaAnnot = NULL,
  colAnnot = NULL,
  controls = NA_integer_,
  rowAnnot = NULL,
  BPPARAM = NULL
)
```

```r
## S4 method for signature 'matrix'
MpraObject(
  dnaCounts,
  rnaCounts,
  dnaAnnot = NULL,
  rnaAnnot = NULL,
  colAnnot = NULL,
  controls = NA_integer_,
  rowAnnot = NULL,
  BPPARAM = NULL
)
```

```r
## S4 method for signature 'SummarizedExperiment'
MpraObject(
  dnaCounts,
  rnaCounts,
  dnaAnnot = NULL,
  rnaAnnot = NULL,
  colAnnot = NULL,
  controls = NA,
  rowAnnot = NULL,
  BPPARAM = NULL
)
```
dnaCounts(obj)

## S4 method for signature 'MpraObject'
dnaCounts(obj)

rnaCounts(obj)

## S4 method for signature 'MpraObject'
rnaCounts(obj)

dnaAnnot(obj)

## S4 method for signature 'MpraObject'
dnaAnnot(obj)

rnaAnnot(obj)

## S4 method for signature 'MpraObject'
rnaAnnot(obj)

rowAnnot(obj)

## S4 method for signature 'MpraObject'
rowAnnot(obj)

controls(obj)

## S4 method for signature 'MpraObject'
controls(obj)

dnaDepth(obj)

## S4 method for signature 'MpraObject'
dnaDepth(obj)

rnaDepth(obj)

## S4 method for signature 'MpraObject'
rnaDepth(obj)

model(obj)

## S4 method for signature 'MpraObject'
model(obj)
MpraObject

Arguments

dnaCounts the DNA count matrix, or a SummarizedExperiment object containing the DNA Counts and column annotations for the DNA data. If the input is a SummarizedExperiment object, the dnaAnnot (or colAnnot) arguments will be ignored

rnaCounts the RNA count matrix, or a SummarizedExperiment object containing the RNA Counts and column annotations for the RNA data. If the input is a SummarizedExperiment object, the rnaAnnot (or colAnnot) arguments will be ignored

dnaAnnot data.frame with the DNA column (sample) annotations

rnaAnnot data.frame with the RNA column (sample) annotations

colAnnot if annotations for DNA and RNA are identical, they can be set at the same time using colAnnot instead of using both rnaAnnot and dnaAnnot

controls IDs of the rows in the matrices that correspond to negative control enhancers. These are used to establish the null for quantification purposes, and to correct systemic bias in comparative analyses. Can be a character vectors (corresponding to rownames in the data matrices), logical or numeric indices.

rowAnnot a data.frame with the row (candidate enhancer) annotations. The names must match the row names in the DNA and RNA count matrices.

BPPARAM a parallelization backend using the BiocParallel package, see more details [here](http://bioconductor.org/packages/release/bioc/html/BiocParallel.html)

obj The MpraObject to extract properties from

Value

an initialized MpraObject

Accessors

MpraObject properties can be accessed using accessor functions

dnaCounts the DNA count matrix

rnaCounts the RNA count matrix

daAnnot data.frame with the DNA column (sample) annotations

rnaAnnot data.frame with the RNA column (sample) annotations

rowAnnot data.frame with the row (candidate enhancer) annotations

model the distributional model used. The Gamma-Poisson convolutional model is used by default. see setModel

daDepth The library size correction factors computed for the DNA libraries. These are computed by the estimateDepthFactors function and can be set manually using the setDepthFactors function

rnaDepth The library size correction factors computed for the RNA libraries. These are computed by the estimateDepthFactors function and can be set manually using the setDepthFactors function
setDepthFactors

Description

Manually set library depth correction factors

Usage

setDepthFactors(obj, dnaDepth, rnaDepth)

Arguments

obj  the MpraObject
dnaDepth  library size factors for the DNA data, a numeric vector of length of the number of columns in the DNA data matrix
rnaDepth  library size factors for the RNA data, a numeric vector of length of the number of columns in the RNA data matrix
setValue

the MpraObject with library depth factors

Examples

data <- simulateMPRA(tr = rep(2,10), da=NULL, nbatch=2, nbc=20)
obj <- MpraObject(dnaCounts = data$obs.dna, 
rnaCounts = data$obs.rna, 
colAnnot = data$annot)
## set constant depth factors (no depth correction)
obj <- setDepthFactors(obj, dnaDepth = rep(1, NCOL(data$obs.dna)), 
                       rnaDepth = rep(1, NCOL(data$obs.rna)))

setValue

Set the distributional model used. Default is gamma.pois, and is recommended. Other supported models are ln.nb in which the DNA follows a log-normal distribution and the RNA follows a negative binomial, and ln.ln in which both follow log-normal distributions. To use alternative distributional models, use this function before fitting the model.

Description

Set the distributional model used. Default is gamma.pois, and is recommended. Other supported models are ln.nb in which the DNA follows a log-normal distribution and the RNA follows a negative binomial, and ln.ln in which both follow log-normal distributions. To use alternative distributional models, use this function before fitting the model.

Usage

setValue(obj, model)

Arguments

obj the MPRAnalyze object
model the character identifier of the model to be used. Currently supported models: "ln.nb", "gamma.pois", "ln.ln"

Value

the MPRAnalyze with the model set for the given value
Examples

```r
data <- simulateMPRA(tr = rep(2, 10), da=NULL, nbatch=2, nbc=20)
obj <- MpraObject(dnaCounts = data$obs.dna,
rnaCounts = data$obs.rna,
colAnnot = data$annot)
obj <- estimateDepthFactors(obj, lib.factor = "batch", which.lib = "both")
obj <- setModel(obj, "ln.ln")
obj <- analyzeQuantification(obj, dnaDesign = ~ batch + barcode,
rnaDesign = ~1)
```

**simulateMPRA**  
*Simulate an MPRA dataset*

**Description**

Simulate an MPRA dataset

**Usage**

```r
simulateMPRA(
  tr = rep(2, 100),
  da = NULL,
  dna.noise.sd = 0.2,
  rna.noise.sd = 0.3,
  dna.inter = 5,
  dna.inter.sd = 0.5,
  nbc = 100,
  coef.bc.sd = 0.5,
  nbatch = 3,
  coef.batch.sd = 0.5
)
```

**Arguments**

- `tr`: a vector of the true transcription rates, in log scale. The length of the vector determines the number of enhancers included in the dataset. Default is 100 enhancers of identical transcription rate of 2.
- `da`: a vector determinig differential activity. Values are assumed to be in log scale, and will be used in the model as log Fold-Change values. If NULL (default) a single condition is simulated.
- `dna.noise.sd`: level of noise to add to the DNA library
- `rna.noise.sd`: level of noise to add to the RNA library
- `dna.inter`: the baseline DNA levels (intercept term), controlling the true mean abundance of plasmids
- `dna.inter.sd`: the true variation of the plasmid levels
- `nbc`: number of unique barcode to include per enhancer
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>coef.bc.sd</td>
<td>true variation between barcodes</td>
</tr>
<tr>
<td>nbatch</td>
<td>number of batches to simulate</td>
</tr>
<tr>
<td>coef.batch.sd</td>
<td>the level of true variation that distinguishes batches (the size of the batch effects)</td>
</tr>
</tbody>
</table>

### Details

The data is generated by using the same nested-GLM construct that MPRAnalyze uses, with non-standard log-normal noise models (whereas default MPRAnalyze uses a Gamma-Poisson model). The data generated can have multiple batches, and either 1 or 2 conditions, and the simulated data is always paired (DNA and RNA extracted from the same library). User can control both true and observed variation levels (noise), the number of expected plasmids per barcode, the true transcription ratio, the size of the batch and barcode effects.

### Value

A list:

- true.dna: The true DNA abundances
- obs.dna: The observed DNA counts
- true.rna: The true RNA abundances
- obs.rna: The observed RNA counts
- annot: The annotations data.frame for each sample

### Examples

```r
# single condition
data <- simulateMPRA()
# two conditions
data <- simulateMPRA(da=c(rep(-0.5, 50), rep(0.5, 50)))
# more observed noise
data <- simulateMPRA(dna.noise.sd = 0.75, rna.noise.sd = 0.75)
# gradually increasing dataset
data <- simulateMPRA(tr = seq(2,3,0.01), da=NULL)
```

---

**testCoefficient**

*Calculate the significance of a factor in the regression model*

### Description

Calculate the significance of a factor in the regression model

### Usage

```r
testCoefficient(obj, factor, contrast)
```
testEmpirical

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>obj</code></td>
<td>the MpraObject</td>
</tr>
<tr>
<td><code>factor</code></td>
<td>the name of the factor to make the comparison on</td>
</tr>
<tr>
<td><code>contrast</code></td>
<td>the character value of the factor to use as a contrast. See details.</td>
</tr>
</tbody>
</table>

Value

a data.frame of the results this include the test statistic, logFC, p-value and BH-corrected FDR.

Examples

data <- simulateMPRA(tr = rep(2,5), da=c(rep(0,2), rep(1,3)),
nbatch=2, nbc=15)
obj <- MpraObject(dnaCounts = data$obs.dna,
rnaCounts = data$obs.rna,
colAnnot = data$annot)
obj <- estimateDepthFactors(obj, lib.factor = "batch", which.lib = "both")

## fit.se must be TRUE for coefficient based testing to work
obj <- analyzeComparative(obj, dnaDesign = ~ batch + barcode + condition,
rnaDesign = ~ condition, fit.se = TRUE)
results <- testCoefficient(obj, "condition", "contrast")

testEmpirical test for significant activity (quantitative analysis) using various empirical tests (see details)

Description

test for significant activity (quantitative analysis) using various empirical tests (see details)

Usage

testEmpirical(
    obj,
    statistic = NULL,
    useControls = TRUE,
    twoSided = FALSE,
    subset = NULL
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>obj</code></td>
<td>the MpraObject, after running an analysis function</td>
</tr>
<tr>
<td><code>statistic</code></td>
<td>if null [default], the intercept term is used as the score. An alternate score can be provided by setting 'statistic'. Must be a numeric vector.</td>
</tr>
</tbody>
</table>
useControls is TRUE and controls are available, use the controls to establish the background model and compare against. This allows for more accurate zscores as well as empirical p-values.

twoSided should the p-value be from a two-sided test (default: FALSE, right-side test)

subset only test a subset of the enhancers in the object (logical, indices or names). Default is NULL, then all the enhancers are included.

Value

a data.frame of empirical summary statistics based on the model’s estimate of slope, or the given statistic. These are:

- statistic: the statistic (either the provided, or extracted from the models)
- zscore: Z-score of the statistic (number of standard deviations from the mean). If controls are available, the score is based on their distribution: so it’s the number of control-sd from the control-mean
- mad.score: a median-based equivalent of the Z-score, with less sensitivity to outlier values. If controls are provided, it’s based on their distribution.
- pval.zscore: a p-value based on the normal approximation of the Z-scores
- pval.empirical: only available if negative controls are provided. empirical P-value, using the control distribution as the null

Examples

data <- simulateMPRA(tr = rep(2,10), da=NULL, nbatch=2, nbc=15)
obj <- MpraObject(dnaCounts = data$obs.dna,
rnaCounts = data$obs.rna,
colAnnot = data$annot)
obj <- estimateDepthFactors(obj, lib.factor = "batch", which.lib = "both")
obj <- analyzeQuantification(obj, dnaDesign = ~ batch + barcode,
rnaDesign = ~1)
results <- testEmpirical(obj)

## or test with a different statistic:
aggregated.ratio <- rowSums(data$obs.rna) / rowSums(data$obs.dna)
results <- testEmpirical(obj, aggregated.ratio)

testLrt

Calculate likelihood ratio test for the specific nested model

description

Calculate likelihood ratio test for the specific nested model

usage

testLrt(obj)
Arguments

obj the MpraObject containing the full and reduced Value

results data frame

Note

Must be run after running an LRT-based analysis

Examples

data <- simulateMPRA(tr = rep(2,5), da=c(rep(0,2), rep(1,3)),
nbatch=2, nbc=15)
obj <- MpraObject(dnaCounts = data$obs.dna,
rnaCounts = data$obs.rna,
colAnnot = data$annot)
obj <- estimateDepthFactors(obj, lib.factor = "batch", which.lib = "both")
obj <- analyzeComparative(obj, dnaDesign = ~ batch + barcode + condition,
rnaDesign = ~ condition, reducedDesign = ~ 1)
results <- testLrt(obj)
Index

* datasets
  ChrEpi, 4
  analyzeComparative, 2
  analyzeQuantification, 3
  ce.colAnnot (ChrEpi), 4
  ce.control (ChrEpi), 4
  ce.rnaCounts (ChrEpi), 4
 ChrEpi, 4
controls (MpraObject), 11
controls, MpraObject-method (MpraObject), 11

dnaAnnot (MpraObject), 11
dnaAnnot, MpraObject-method (MpraObject), 11
dnaCounts (MpraObject), 11
dnaCounts, MpraObject-method (MpraObject), 11
dnaDepth (MpraObject), 11
dnaDepth, MpraObject-method (MpraObject), 11

estimateDepthFactors, 5, 13
extractModelParameters_DNA (getModelParameters_DNA), 10
extractModelParameters_RNA (getModelParameters_DNA), 10

getAlpha, 6
getDistrParam_DNA, 7
getDistrParam_RNA (getDistrParam_DNA), 7
getFits_DNA, 8
getFits_RNA, 9
getModelParameters_DNA, 10
getModelParameters_RNA (getModelParameters_DNA), 10

model (MpraObject), 11
model, MpraObject-method (MpraObject), 11
MpraObject, 11
MpraObject, matrix-method (MpraObject), 11
MpraObject, SummarizedExperiment-method (MpraObject), 11

rnaAnnot (MpraObject), 11
rnaAnnot, MpraObject-method (MpraObject), 11
rnaCounts (MpraObject), 11
rnaCounts, MpraObject-method (MpraObject), 11
rnaDepth (MpraObject), 11
rnaDepth, MpraObject-method (MpraObject), 11
rowAnnot (MpraObject), 11
rowAnnot, MpraObject-method (MpraObject), 11

setDepthFactors, 13, 14
setModel, 13, 15
simulateMPRA, 16

testCoefficient, 17
testEmpirical, 18
testLrt, 19