Package 'beer'

November 12, 2024

Type Package

Title Bayesian Enrichment Estimation in R

Version 1.10.0

Description BEER implements a Bayesian model for analyzing phage-immunoprecipitation sequencing (PhIP-seq) data. Given a PhIPData object, BEER returns posterior probabilities of enriched antibody responses, point estimates for the relative fold-change in comparison to negative control samples, and more. Additionally, BEER provides a convenient implementation for using edgeR to identify enriched antibody responses.

License MIT + file LICENSE

Encoding UTF-8

LazyData false

Depends R (>= 4.2.0), PhIPData (>= 1.1.1), rjags

Imports cli, edgeR, BiocParallel, methods, progressr, stats, SummarizedExperiment, utils

Suggests testthat (>= 3.0.0), BiocStyle, covr, codetools, knitr, rmarkdown, dplyr, ggplot2, spelling

SystemRequirements JAGS (4.3.0)

biocViews Software, StatisticalMethod, Bayesian, Sequencing, Coverage

URL https://github.com/athchen/beer/

BugReports https://github.com/athchen/beer/issues

Config/testthat/edition 3

VignetteBuilder knitr

RoxygenNote 7.2.0

Language en-US

git_url https://git.bioconductor.org/packages/beer

git_branch RELEASE_3_20

git_last_commit a28e699

git_last_commit_date 2024-10-29

Repository Bioconductor 3.20

Date/Publication 2024-11-12

Author Athena Chen [aut, cre] (<https://orcid.org/0000-0001-6900-2264>),
Rob Scharpf [aut],
Ingo Ruczinski [aut]

Maintainer Athena Chen <achen70@jhu.edu>

Contents

`beer-package` *beer: an R package for quantifying antibody reactivities for PhIP-Seq data*

Description

The core functions of beer are `brew` and `runEdgeR`. To learn more about `beer`, see the vignette using `browseVignettes(package = "beer")`.

Author(s)

Maintainer: Athena Chen <achen70@jhu.edu> (ORCID)

Authors:

- Rob Scharpf <rscharpf@jhu.edu>
- Ingo Ruczinski <ingo@jhu.edu>

See Also

Useful links:

- https://github.com/athchen/beer/
- Report bugs at https://github.com/athchen/beer/issues

.beadsRRBeer *Function to run the beads-only round robin using BEER*

Description

Each sample is run in comparison to all other beads-only samples to approximate the false positive rate of detecting enrichments.

Usage

```
.beadsRRBeer(
  object,
  prior.params = list(method = "edgeR", a_pi = 2, b_pi = 300, a_phi = 1.25, b_phi =
    0.1, a_c = 80, b_c = 20, fc = 1),
  beads.args = list(lower = 1),
  jags.params = list(n.chains = 1, n.adapt = 1000, n.iter = 10000, thin = 1, na.rm =
   TRUE, burn.in = 0, post.thin = 1, seed = as.numeric(format(Sys.Date(), "%Y%m%d"))),
  sample.dir = NULL,
  assay.names = c(phi = NULL, phi_Z = "logfc", Z = "prob", c = "sampleInfo", pi =
    "sampleInfo"),
  summarize = TRUE,
  BPPARAM = bpparam()
)
```

Arguments

object	PhIPData object
prior.params	named list of prior parameters
beads.args	named list of parameters supplied to estimating beads-only prior parameters (a_0, b_0)
jags.params	named list of parameters for running MCMC using JAGS
sample.dir	path to temporarily store RDS files for each sample run, if NULL then [base::tempdir] is used to temporarily store MCMC output and cleaned afterwards.
assay.names	named vector indicating where MCMC results should be stored in the PhIPData object
summarize	logical indicating whether to return a PhIPData object.
BPPARAM	[BiocParallel::BiocParallelParam] passed to BiocParallel functions.

Value

vector of process IDs or a PhIPData object

`.beadsRREdgeR` *Function to run the beads-only round robin using edgeR*

Description

Each sample is run in comparison to all other beads-only samples to approximate the false positive rate of detecting enrichments.

Usage

```
.beadsRREdgeR(
  object,
  threshold.cpm = 0,
  threshold.prevalence = 0,
  assay.names = c(logfc = "logfc", prob = "prob"),
  de.method = "exactTest",
  BPPARAM = BiocParallel::bpparam()
)
```

Arguments

`object`	A PhIPData object of only beads-only samples.
`threshold.cpm`	CPM threshold to be considered present in a sample
`threshold.prevalence`	proportion of beads-only samples that surpass `threshold.cpm`.
`assay.names`	named vector specifying the assay names for the log2(fold-change) and exact test p-values. If the vector is not names, the first and second entries are used as defaults.
`de.method`	character describing which edgeR test for differential expression should be used. Must be one of 'exactTest' or 'glmQLFTest'.
`BPPARAM`	`[BiocParallel::BiocParallelParam]` passed to BiocParallel functions.

Value

vector of process IDs

`.brewSamples` *Run BEER for all samples*

Description

Encapsulated function to run each sample against all beads-only samples. The code is wrapped in this smaller function to (1) modularize the code and (2) make sure the cli output colors don't change.

Usage

```
.brewSamples(
  object,
  sample.id,
  beads.id,
  se.matrix,
  prior.params,
  beads.prior,
  beads.args,
  jags.params,
  tmp.dir,
  BPPARAM
)
```

Arguments

`object`	PhIPData object
`sample.id`	vector of sample IDs to iterate over
`beads.id`	vector of IDs corresponding to beads-only samples
`se.matrix`	matrix indicating which peptides are clearly enriched
`prior.params`	list of prior parameters
`beads.prior`	data frame of beads-only prior parameters
`beads.args`	named list of parameters supplied to estimating beads-only prior parameters (a_0, b_0)
`jags.params`	list of JAGS parameters
`tmp.dir`	directory to store JAGS samples
`BPPARAM`	`[BiocParallel::BiocParallelParam]` passed to BiocParallel functions.

Value

vector of process id's for internal checking of whether functions were parallelized correctly.

.checkCounts *Function to check that the counts matrix only contains integers*

Description

Function to check that the counts matrix only contains integers

Usage

```
.checkCounts(object)
```

Arguments

object	PhIPData object

Value

nothing if all counts are integers, and error otherwise

.checkOverwrite *Function to check whether an assay will be overwritten*

Description

If the an assay is not specified (e.g. with NA), then .checkOverwrite() will return FALSE (rather than NA).

Usage

```
.checkOverwrite(object, assay.names)
```

Arguments

object	PhIPData object
assay.names	character vector of assay names

Value

logical vector indicating whether data in an assay will be overwritten

.edgeRBeads — *Estimate edgeR dispersion parameters from the beads-only data using qCML*

Description

Wrapper function to estimate edgeR dispersion parameters from beads-only samples. Peptides can be pre-filtered based on a minimum read count per million (cpm) and the proportion of beads-only samples that surpass the cpm threshold.

Usage

```
.edgeRBeads(object, threshold.cpm = 0, threshold.prevalence = 0)
```

Arguments

`object`	`PhIPData` object (can have actual serum samples)
`threshold.cpm`	CPM threshold to be considered present in a sample
`threshold.prevalence`	proportion of beads-only samples that surpass `threshold.cpm`.

Value

a DGEList object with common, trended, and tagwise dispersion estimates

.edgeRBeadsQLF — *Estimate edgeR dispersion parameters from the beads-only samples using Cox-Reid profile adjusted likelihood method for estimating dispersions.*

Description

Wrapper function to estimate edgeR dispersion parameters from beads-only samples. Peptides can be pre-filtered based on a minimum read count per million (cpm) and the proportion of beads-only samples that surpass the cpm threshold.

Usage

```
.edgeRBeadsQLF(object, threshold.cpm = 0, threshold.prevalence = 0)
```

Arguments

`object`	`PhIPData` object (can have actual serum samples)
`threshold.cpm`	CPM threshold to be considered present in a sample
`threshold.prevalence`	proportion of beads-only samples that surpass `threshold.cpm`.

Value

a DGEList object with common, trended, and tagwise dispersion estimates

.getABEdgeR *Derive beta shape parameters using edgeR dispersion estimates*

Description

Given a PhIPData object, beads-only shape parameters are estimated by first deriving the peptide-specific edgeR dispersion estimate ϕ^{edgeR}. ϕ^{edgeR} corresponds to the squared coefficient of variation for the proportion of reads pulled for a given peptide. Using ϕ^{edgeR} to derive an estimate of the variance for the proportion of reads pulled by a single peptide, the mean and variance are converted to shape parameters of a beta distribution.

Usage

```
.getABEdgeR(
  object,
  threshold.cpm = 0,
  threshold.prevalence = 0,
  lower = 1,
  upper = Inf
)
```

Arguments

object	a PhIPData object.
threshold.cpm	CPM threshold to be considered present in a sample.
threshold.prevalence	proportion of beads-only samples that surpass threshold.cpm.
lower	minimum value of the beta shape parameters.
upper	maximum value of the beta shape parameters.

Value

dataframe with rows corresponding to peptides and columns corresponding to estimated shape parameters of the beta distribution.

See Also

[.edgeRBeads()] for estimating ϕ^{edgeR}

.getABMLE *Wrapper function to derive MLE estimates of a, b from beads-only samples*

Description

Wrapper function to derive MLE estimates of a, b from beads-only samples

Usage

```
.getABMLE(
  object,
  prop.offset = 1e-08,
  optim.method = "default",
  lower = 1,
  upper = Inf
)
```

Arguments

object	a PhIPData object
prop.offset	offset to use when the proportion of reads is 0.
optim.method	optimization method passed to [stats::optim].
lower	lower bound for the shape parameters.
upper	upper bound for the shape parameters.

Value

a data frame of MLE estimates of a, b

See Also

[stats::optim] for available optimization methods

.getABMLEProp *Helper function to derive MLE estimates of a, b from a vector of proportions*

Description

Helper function to derive MLE estimates of a, b from a vector of proportions

Usage

```
.getABMLEProp(
  prop,
  prop.offset = 1e-08,
  optim.method = "default",
  lower = 1,
  upper = Inf
)
```

Arguments

prop	vector of proportions.
prop.offset	offset to use when the proportion of reads is 0.
optim.method	optimization method passed to [stats::optim].
lower	lower bound for the shape parameters.
upper	upper bound for the shape parameters.

Value

a data frame of MLE estimates of a, b

See Also

[stats::optim] for available optimization methods

`.getABMOM` *Wrapper function to derive MOM estimates of a, b from beads-only samples*

Description

Wrapper function to derive MOM estimates of a, b from beads-only samples

Usage

```
.getABMOM(
  object,
  offsets = c(mean = 1e-08, var = 1e-08),
  lower = 1,
  upper = Inf,
  ...
)
```

Arguments

object	a PhIPData object.
offsets	vector defining the offset to use when the mean and/or variance are zero.
lower	lower bound for the shape parameters.
upper	upper bound for the shape parameters.
...	parameters passed to [base::mean] and [stats::var].

Value

a data frame with MOM estimates of a, b

.getABMOMProp *Helper function to derive MOM estimates of a, b from a vector of proportions*

Description

Helper function to derive MOM estimates of a, b from a vector of proportions

Usage

```
.getABMOMProp(
  prop,
  offsets = c(mean = 1e-08, var = 1e-08),
  lower = 1,
  upper = Inf,
  ...
)
```

Arguments

prop	vector of proportions.
offsets	vector defining the offset to use when the mean and/or variance are zero.
lower	lower bound for the shape parameters.
upper	upper bound for the shape parameters.
...	parameters passed to [base::mean] and [stats::var].

Value

a data frame with MOM estimates of a, b

.guessEnrichedEdgeR *Guess super-enriched peptides based on edgeR fold-change estimates*

Description

Guess super-enriched peptides based on edgeR fold-change estimates

Usage

```
.guessEnrichedEdgeR(object, threshold = 15, fc.name = "logfc")
```

Arguments

object	PhIPData object.
threshold	minimum estimated fc for a peptide to be considered super-enriched.
fc.name	assay name corresponding to the assay that stores the edgeR estimated log2 fold-changes.

Value

logical matrix of the with the same dimensions as `object` indicating which peptides are considered super-enriched.

.guessEnrichedMLE *Guess enriched peptides based on MLE estimates of the true fold-change*

Description

Guess enriched peptides based on MLE estimates of the true fold-change

Usage

```
.guessEnrichedMLE(object, beads.prior, threshold = 15)
```

Arguments

`object`	PhIPData object.
`beads.prior`	data.frame of prior parameters for beads-only samples.
`threshold`	minimum estimated fc for a peptide to be considered super-enriched.

Value

logical matrix of the with the same dimensions as `object` indicating which peptides are considered super-enriched.

.tidyAssayNames *Clean-up specified assay names*

Description

Tidy inputs related to ‘assay.names‘. Supplies default values for missing parameters and ensures that all required parameters are present.

Usage

```
.tidyAssayNames(assay.names)
```

Arguments

`assay.names`	named list specifying where to store each assay.

Value

tidied list of assay.names

.tidyInputsJAGS *Clean inputs for JAGS parameters*

Description

Tidy inputs related to ‘jags.params‘. Supplies default values for missing parameters and ensures that all required parameters are present.

Usage

```
.tidyInputsJAGS(jags.params)
```

Arguments

jags.params	named list of JAGS parameters

Value

tidied list of JAGS parameters.

.tidyInputsPrior *Clean up inputs for prior estimation*

Description

Tidy inputs related to ‘prior.parameters‘. Supplies default values for missing parameters and ensures that all required parameters are present.

Usage

```
.tidyInputsPrior(prior.params, object, beads.args)
```

Arguments

prior.params	named list of prior parameters
object	PhIPData object
beads.args	parameters used to estimate a_0, b_0

Value

tidied list of prior parameters.

.tidyInputsSE *Clean up inputs for identifying super-enriched peptides*

Description

Tidy inputs related to 'se.params'. Supplies default values for missing parameters and ensures that all required parameters are present.

Usage

```
.tidyInputsSE(se.params, beads.prior)
```

Arguments

se.params	named list of parameters for super-enriched estimation
beads.prior	data.frame with beads-only parameters

Value

tidied list of parameters for identifying super-enriched peptides.

beadsRR *Beads-only round robin*

Description

To approximate the false positive rate of each approach, each beads-only sample is run individually against all other samples. For BEER, this means that the sample to be compared is encoded as an actual sample, and prior parameters for beads-only samples are re-estimated. Thus, the beads-only round robin also serves to assess how similar the beads-only samples are to one another.

Usage

```
beadsRR(object, method, BPPARAM = BiocParallel::bpparam(), ...)
```

Arguments

object	PhIPData object
method	one of 'beer' or 'edgeR' specifying which method to use.
BPPARAM	[BiocParallel::BiocParallelParam] passed to BiocParallel functions.
...	parameters passed to the method specific functions. See the *Details* section below for additional information.

Details

If `method == 'beer'`, then valid parameters include `prior.params`, `beads.args`, `jags.params`, `sample.dir`, `assay.names`, and `summarize`. A description of the first four parameters can be found in `brew`. `summarize` is a logical value indicating whether a PhIPData object with the summarized results should be returned. When running `beadsRR`, `summarize` typically does not need to be changed.

When `method == 'edgeR'`, `threshold.cpm`, `threshold.prevalence`, and `assay.names` are valid additional parameters that can be supplied to `beadsRR`. See `edgeR` for additional details on each of these parameters.

Value

a PhIPData object

See Also

`brew` for BEER parameters, `edgeR` for edgeR parameters, and `[BiocParallel::BiocParallelParam]` for parallelization.

Examples

```
sim_data <- readRDS(system.file("extdata", "sim_data.rds", package = "beer"))

beadsRR(sim_data, method = "beer")
beadsRR(sim_data, method = "edgeR")
beadsRR(sim_data, method = "edgeR", de.method = "glmQLFTest")
```

---

brew *Bayesian Enrichment Estimation in R (BEER)*

Description

Run BEER to estimate posterior probabilities of enrichment, sample-specific attenuation constants, relative fold-changes in comparison to beads-only samples, and proportion of peptides enriched per sample as described in Chen et. al. See *Details* for more information on input parameters.

Usage

```
brew(
  object,
  prior.params = list(method = "edgeR", a_pi = 2, b_pi = 300, a_phi = 1.25, b_phi =
    0.1, a_c = 80, b_c = 20, fc = 1),
  beads.args = list(lower = 1),
  se.params = list(method = "mle"),
  jags.params = list(n.chains = 1, n.adapt = 1000, n.iter = 10000, thin = 1, na.rm =
   TRUE, burn.in = 0, post.thin = 1, seed = as.numeric(format(Sys.Date(), "%Y%m%d"))),
  sample.dir = NULL,
  assay.names = c(phi = NULL, phi_Z = "logfc", Z = "prob", c = "sampleInfo", pi =
    "sampleInfo"),
  beadsRR = FALSE,
  BPPARAM = bpparam()
)
```

Arguments

object	PhIPData object
prior.params	named list of prior parameters
beads.args	named list of parameters supplied to estimating beads-only prior parameters (a_0, b_0)
se.params	named list of parameters specific to identifying clearly enriched peptides
jags.params	named list of parameters for running MCMC using JAGS
sample.dir	path to temporarily store RDS files for each sample run, if NULL then [base::tempdir] is used to temporarily store MCMC output and cleaned afterwards.
assay.names	named vector indicating where MCMC results should be stored in the PhIPData object
beadsRR	logical value specifying whether each beads-only sample should be compared to all other beads-only samples.
BPPARAM	[BiocParallel::BiocParallelParam] passed to BiocParallel functions.

Details

prior.params. List of prior parameters. Parameters include,

- method: method used to estimate beads-only prior parameters a_0, b_0. Valid methods include 'custom' or any of the methods specified in getAB. If method = 'custom' is specified, a_0 and b_0 must be included in the list of prior parameters. 'edgeR' is used as the default method for estimating a_0, b_0.
- a_pi, b_pi: prior shape parameters for the proportion of peptides enriched in a sample. Defaults to 2 and 300, respectively.
- a_phi, b_phi: prior shape parameters of the gamma distribution that describe the valid range of enriched-fold changes. The shift is specified by fc. The default values of a_phi and b_phi are 1.25 and 0.1, respectively.
- a_c, b_c: prior shape parameters for the attenuation constant. Default values for a_c and b_c are 80 and 20.
- fc: minimum fold change for an enriched-peptide. fc describes the shift in the gamma distribution.

beads.args. Named list of parameters supplied to getAB. The estimation method used is specified in prior.params, but other valid parameters include lower and upper bounds for elicited parameters. As JAGS recommends that $a, b > 1$ for the beta distribution, beads.args defaults to list(lower = 1).

se.params. Named list of parameters supplied to guessEnriched. By default list(method = 'mle') is used to identify clearly enriched peptides.

jags.params. Named list of parameters for MCMC sampling. By default, BEER only runs one chain with 1,000 adaptation iteration and 10,000 sampling iterations. If unspecified, BEER uses the current date as the seed.

sample.dir. Path specifying where to store the intermediate results. If NULL, then results are stored in the default temporary directory. Otherwise, the MCMC samples for running BEER on each sample is stored as a single RDS file in the specified directory.

assay.names. Named list specifying where to store the point estimates. If NULL, estimates are not added to the PhIPData object. Valid exported estimates include,

- phi: fold-change estimate after marginalizing over the posterior probability of enrichment. By default point estimates are not exported.
- phi_Z: fold-change estimate presuming the peptide is enriched. By default phi_Z estimates are stored in 'logfc' assay.
- Z: posterior probability of enrichment. Estimates are stored in the 'prob' assay by default.
- c: attenuation constant estimates. Stored in 'sampleInfo' by default.
- pi: point estimates for the proportion of peptides enriched in a sample. Stored in 'sampleInfo' by default.

Value

A PhIPData object with BEER results stored in the locations specified by assay.names.

See Also

[BiocParallel::BiocParallelParam] for subclasses, beadsRR for running each beads-only sample against all remaining samples, getAB for more information about valid parameters for estimating beads-only prior parameters, guessEnriched for more information about how clearly enriched peptides are identified, and [rjags::jags.model] for MCMC sampling parameters.

Examples

```
sim_data <- readRDS(system.file("extdata", "sim_data.rds", package = "beer"))

## Default back-end evaluation
brew(sim_data)

## Serial
brew(sim_data, BPPARAM = BiocParallel::SerialParam())

## Snow
brew(sim_data, BPPARAM = BiocParallel::SnowParam())
```

---

brewOne *Run BEER for one sample*

Description

This function is not really for external use. It's exported for parallelization purposes. For more detailed descriptions see brew.

Usage

```
brewOne(
  object,
  sample,
  prior.params,
  n.chains = 1,
  n.adapt = 1000,
  n.iter = 10000,
  thin = 1,
```

```
  na.rm = TRUE,
  ...,
  seed = as.numeric(format(Sys.Date(), "%Y%m%d"))
)
```

Arguments

object	PhIPData object
sample	sample name
prior.params	vector of prior parameters
n.chains	number of chains to run
n.adapt	number of iterations to use as burn-in.
n.iter	number of iterations for the MCMC chain to run (after n.adapt)
thin	thinning parameter
na.rm	what to do with NA values (for JAGS)
...	extra parameters for JAGS
seed	number/string for reproducibility purposes.

Value

nothing, saves the the results to an RDS in either a temp directory or the specified directory.

Examples

```
sim_data <- readRDS(system.file("extdata", "sim_data.rds", package = "beer"))

beads_prior <- getAB(subsetBeads(sim_data), "edgeR")
brewOne(sim_data, "9", list(
    a_0 = beads_prior[["a_0"]],
    b_0 = beads_prior[["b_0"]],
    a_pi = 2, b_pi = 300,
    a_phi = 1.25, b_phi = 0.1,
    a_c = 80, b_c = 20,
    fc = 1
))
```

---

edgeROne *Run edgeR for one sample against all the beads-only samples.*

Description

This function is not really for external use. It's exported for parallelization purposes. For more detailed descriptions see runEdgeR.

Usage

```
edgeROne(object, sample, beads, common.disp, tagwise.disp, trended.disp)
```

Arguments

object	PhIPData object
sample	sample name of the sample to compare against beads-only samples
beads	sample names for beads-only samples
common.disp	edgeR estimated common dispersion parameter
tagwise.disp	edgeR estimated tagwise dispersion parameter
trended.disp	edgeR estimated trended dispersion parameter

Value

list with sample name, log2 fc estimate, and log10 p-value

Examples

```
sim_data <- readRDS(system.file("extdata", "sim_data.rds", package = "beer"))

beads_disp <- beer:::.edgeRBeads(sim_data)
edgeROne(
    sim_data, "9", colnames(sim_data)[sim_data$group == "beads"],
    beads_disp$common.dispersion, beads_disp$tagwise.disp,
    beads_disp$trended.disp
)
```

---

edgeROneQLF *Run edgeR for one sample against all the beads-only samples using edgeR's QLF Test for differential expression.*

Description

This function is not really for external use. It's exported for parallelization purposes. For more detailed descriptions see runEdgeR.

Usage

```
edgeROneQLF(object, sample, beads, common.disp, tagwise.disp, trended.disp)
```

Arguments

object	PhIPData object
sample	sample name of the sample to compare against beads-only samples
beads	sample names for beads-only samples
common.disp	edgeR estimated common dispersion parameter
tagwise.disp	edgeR estimated tagwise dispersion parameter
trended.disp	edgeR estimated trended dispersion parameter

Value

list with sample name, log2 fc estimate, and log10 p-value

Examples

```
sim_data <- readRDS(system.file("extdata", "sim_data.rds", package = "beer"))

beads_disp <- beer:::.edgeRBeadsQLF(sim_data)
edgeROneQLF(
    sim_data, "9", colnames(sim_data)[sim_data$group == "beads"],
    beads_disp$common.dispersion, beads_disp$tagwise.disp,
    beads_disp$trended.disp
)
```

`getAB` *Estimate beads-only shape parameters*

Description

Beta shape parameters are estimated using the proportion of reads-pulled per peptide across the beads-only samples. Currently, only three estimation methods are supported: edgeR, method of moments (MOM), maximum likelihood (MLE). Note that edgeR can only be used on `PhIPData` objects while MOM and MLE methods can also be applied to vectors of values between 0 and 1. Parameters that can be passed to each method are listed in the details.

Usage

```
getAB(object, method = "mom", ...)
```

Arguments

`object`	a `PhIPData` object or a vector
`method`	one of `c("edgeR", "mle", "mom")` designating which method to use to estimate beads-only prior parameters. MOM is the default method used to estimate shape parameters.
`...`	parameters passed to specific estimating functions. See details for more information

Details

edgeR derived estimates rely on edgeR's peptide-specific dispersion estimates, denoted ϕ^{edgeR}. ϕ^{edgeR} corresponds to the squared coefficient of variation for the proportion of reads pulled for a given peptide. Using ϕ^{edgeR} to derive an estimate of the variance for the proportion of reads pulled by a single peptide, the mean and variance are transformed into shape parameters satisfying the lower and upper bounds. When `method = "edgeR"`, the following additional parameters can be specified.

- `threshold.cpm`: CPM threshold to be considered present in a sample.
- `threshold.prevalence`: proportion of beads-only samples that surpass `threshold.cpm`.
- `lower`: minimum value of the beta shape parameters.
- `upper`: maximum value of the beta shape parameters.

Method of Moments (MOM) estimates are derived by transforming the sample mean and variance to shape parameters of the beta distribution. For `method = "mom"`, the following parameters can be adjusted:

- `offsets`: vector defining the offset to use when the mean and/or variance are zero.
- `lower`: lower bound for the shape parameters.
- `upper`: upper bound for the shape parameters.
- `...`: parameters passed to `[base::mean]` and `[stats::var]`.

Maximum Likelihood (MLE) estimates rely on `[stats::optim]` to derive shape parameters that maximize the likelihood of observed data. By default the L-BFGS-B optimization method is used. Parameters for MLE estimates include:

- `prop.offset`: offset to use when the proportion of reads is 0.
- `optim.method`: optimization method passed to `[stats::optim]`.
- `lower`: lower bound for the shape parameters.
- `upper`: upper bound for the shape parameters.

Value

a data frame of beta shape parameters where each row corresponds to a peptide.

Examples

```
## PhIPData object
sim_data <- readRDS(system.file("extdata", "sim_data.rds", package = "beer"))

getAB(sim_data, method = "edgeR")
getAB(sim_data, method = "mle")
getAB(sim_data, method = "mom")

## Vector of proportions
prop <- rbeta(100, 2, 8)
getAB(prop, method = "mle")
getAB(prop, method = "mom")
```

`getBF` *Calculate Bayes Factors*

Description

Calculate Bayes Factors

Usage

```
getBF(
  object,
  assay.postprob = "prob",
  assay.name = "bayes_factors",
  prior.params = list(a_pi = 2, b_pi = 300)
)
```

Arguments

object	PhIPData object
assay.postprob	string indicating the assay where posterior probabilities are stored.
assay.name	name indicating where the results should be stored in the PhIPData object
prior.params	prior parameters for the probability of enrichment (a_pi, b_pi)

Value

PhIPData object with the results stored in the location specified by `assay.name`.

Examples

```
sim_data <- readRDS(system.file("extdata", "sim_data.rds", package = "beer"))

## Calculate Bayes Factors
getBF(sim_data, "prob", "bayes_factor")
```

---

getExpected *Calculate expected read counts or proportion of reads*

Description

Calculate expected read counts or proportion of reads

Usage

```
getExpected(
  object,
  type = c("rc", "prop"),
  assay.names = c("expected_rc", "expected_prop")
)
```

Arguments

object	PhIPData object
type	any of ‘rc‘ or ‘prop‘ indicating whether the function should return the expected read counts or expected proportion of reads, respectively
assay.names	name(s) indicating where the results should be stored in the PhIPData object

Value

PhIPData object with the results stored in the location specified by `assay.name`.

Examples

```
sim_data <- readRDS(system.file("extdata", "sim_data.rds", package = "beer"))

## Calculate expected read counts
getExpected(sim_data, "rc", "expected_rc")

## Calculate expected proportion of reads
getExpected(sim_data, "prop", "expected_prop")

## Calculate both
getExpected(sim_data)
```

---

guessEnriched *Identifying clearly enriched peptides*

Description

As clearly enriched peptides will always have a 100% posterior probability of enrichment, BEER removes these peptides a priori to running the model. Clearly enriched peptides can be identified using edgeR estimated fold-changes or maximum likelihood estimates based on the specified prior parameters. Additional parameters for each method can be found in the details below.

Usage

```
guessEnriched(object, method = "mle", ...)
```

Arguments

`object`	a `PhIPData` object
`method`	one of "mle" or "edgeR", specifying which method to use to identify clearly enriched peptides
`...`	additional parameters dependent on the method used. See details for more information

Details

edgeR. Identification of clearly enriched peptides relies on edgeR fold-change estimates, so `edgeR` must be run on the `PhIPData` object beforehand. Additional parameters for identifying clearly enriched peptides based on edgeR estimated fold-changes are listed below:

- `object`: a `PhIPData` object.
- `threshold`: minimum estimated fc for a peptide to be considered super-enriched. The default value is 15.
- `fc.name`: assay name corresponding to the assay that stores the edgeR estimated log2 fold-changes.

MLE. As the number of reads tends to be quite large, the estimates for the proportion of reads pulled are generally accurate. Clearly enriched peptides are identified by first comparing the observed read count to the expected read count based on the beads-only prior parameters. Peptides with observed read counts larger than 5 times the expected read counts are temporarily labeled as

enriched, and attenuation constants are estimated by regressing the observed read counts on the expected read counts for all non-enriched peptides. Using this attenuation constant, peptides with fold-changes above some predefined threshold after adjusting for the attenuation constant are considered enriched. Parameters for identifying clearly enriched peptides using the MLE approach are listed below.

- `object`: a `PhIPData` object.
- `threshold`: minimum estimated fc for a peptide to be considered super-enriched.
- `beads.prior`: data.frame of prior parameters for beads-only samples.

Value

a logical matrix of the with the same dimensions as `object` indicating which peptides are considered super-enriched.

Examples

```
sim_data <- readRDS(system.file("extdata", "sim_data.rds", package = "beer"))
edgeR_out <- runEdgeR(sim_data)

guessEnriched(edgeR_out, method = "edgeR", threshold = 15, fc.name = "logfc")
guessEnriched(edgeR_out,
    method = "mle",
    beads.prior = getAB(edgeR_out, method = "edgeR"),
    threshold = 15
)
```

---

guessInits *Derive initial estimates of unknown model parameters*

Description

To reduce converge time and to reduce the likelihood of the slice sampler getting stuck, we use maximum likelihood to derive initial estimates for unknown model parameters.

Usage

```
guessInits(object, beads.prior)
```

Arguments

`object`	a `PhIPData` object
`beads.prior`	a data frame with two columns (named a_0, b_0) containing estimated shape parameters from beads-only samples.

Details

Briefly initial values are defined as follows:

1. `theta_guess[i, j]` = `Y[i, j]/n[j]`, or the the MLE for theta.
2. `Z_guess[i, j]` = 1 if j is a serum sample, and the observed read count is >2x the expected read count assuming `c[j]` = 1.
3. `pi_guess[j]` is the mean of column j in `Z_guess`.
4. `c_guess[j]` is the estimated slope from regressing the observed read counts against the expected read counts (without adjusting for the attenuation constant) for non-enriched peptides only.
5. `phi_guess[i,j]` is the ratio of the observed read counts to the expected read counts multiplied by the attenuation constant.

Value

a list of estimated initial values.

See Also

Methods in [Chen et. al 2022](https://www.biorxiv.org/content/10.1101/2022.01.19.476926v1)

`runEdgeR` *Run edgeR on PhIP-Seq data*

Description

Run edgeR on PhIP-Seq data

Usage

```
runEdgeR(
  object,
  threshold.cpm = 0,
  threshold.prevalence = 0,
  assay.names = c(logfc = "logfc", prob = "prob"),
  beadsRR = FALSE,
  de.method = "exactTest",
  BPPARAM = BiocParallel::bpparam()
)
```

Arguments

`object`	`PhIPData` object
`threshold.cpm`	CPM threshold to be considered present in a sample
`threshold.prevalence`	proportion of beads-only samples that surpass `threshold.cpm`.
`assay.names`	named vector specifying the assay names for the log2(fold-change) and exact test p-values. If the vector is not names, the first and second entries are used as defaults

beadsRR	logical value specifying whether each beads-only sample should be compared to all other beads-only samples.
de.method	character describing which edgeR test for differential expression should be used. Must be one of 'exactTest' or 'glmQLFTest'
BPPARAM	[BiocParallel::BiocParallelParam] passed to BiocParallel functions.

Value

PhIPData object with log2 estimated fold-changes and p-values for enrichment stored in the assays specified by 'assay.names'.

See Also

[BiocParallel::BiocParallelParam], beadsRR

Examples

```
sim_data <- readRDS(system.file("extdata", "sim_data.rds", package = "beer"))

## Default back-end evaluation
runEdgeR(sim_data)

## Serial
runEdgeR(sim_data, BPPARAM = BiocParallel::SerialParam())

## Snow
runEdgeR(sim_data, BPPARAM = BiocParallel::SnowParam())

## With glmQLFTest
runEdgeR(sim_data, de.method = "glmQLFTest")
```

---

summarizeRun *Summarize MCMC chain and return point estimates for BEER parameters*

Description

Posterior means are used as point estimates for c, π, ϕ, and Z. As super-enriched peptides are tossed out before MCMC sampling, super-enriched peptides return NA for the ϕ and Z point estimates. Indices corresponding to a particular peptide in the MCMC sampler are mapped back to the original peptide names.

Usage

```
summarizeRun(
  object,
  jags.files,
  se.matrix,
  burn.in = 0,
  post.thin = 1,
  assay.names = c(phi = NULL, phi_Z = "logfc", Z = "prob", c = "sampleInfo", pi =
```

```
    "sampleInfo"),
  BPPARAM = BiocParallel::bpparam()
)
```

Arguments

object	a PhIPData object
jags.files	list of files containing MCMC sampling results
se.matrix	logical matrix indicating which peptides were identified as super-enriched peptides
burn.in	number of iterations to be burned
post.thin	thinning parameter
assay.names	named vector of specifying where to store point estimates
BPPARAM	[BiocParallel::BiocParallelParam] passed to BiocParallel functions.

Value

PhIPData object with point estimates stored in the assays specified by 'assay.names'.

---

summarizeRunOne *Derive point estimates for c, pi, phi, and Z for a particular sample*

Description

Posterior means are used as point estimates for c, π, ϕ, and Z. As super-enriched peptides are tossed out before MCMC sampling, super-enriched peptides return NA for the ϕ and Z point estimates. Indices corresponding to a particular peptide in the MCMC sampler are mapped back to the original peptide names.

Usage

```
summarizeRunOne(object, file, se.matrix, burn.in = 0, post.thin = 1)
```

Arguments

object	a PhIPData object
file	path to rds file
se.matrix	logical matrix indicating which peptides were identified as super-enriched peptides
burn.in	number of iterations to be burned
post.thin	thinning parameter

Value

list of point estimates for c, pi, phi and Z

Index