

Package ‘epigenomix’

April 5, 2014

Type Package

Title Epigenetic and gene expression data normalization and integration with mixture models

Version 1.2.0

Date 2013-06-10

Author Hans-Ulrich Klein, Martin Schaefer

Maintainer Hans-Ulrich Klein <h.klein@uni-muenster.de>

biocViews ChIPseq, GeneExpression, DifferentialExpression,Classification

Depends R (>= 2.12.0), methods, Biobase, IRanges, GenomicRanges

Imports methods, BiocGenerics, Biobase, IRanges, GenomicRanges,beadarray

Description A package for the integrative analysis of microarray based gene expression and histone modification data obtained by ChIP-seq. The package provides methods for data preprocessing and matching as well as methods for fitting bayesian mixture models in order to detect genes with differences in both data types.

License LGPL-3

R topics documented:

bayesMixModel	2
ChIPseqSet-class	5
eSet	6
integrateData	7
mappedReads	8
matchProbeToPromoter	9
MixModel-class	10
MixModelBayes-class	12
MixModelML-class	13
MixtureComponent-class	14
mlMixModel	15
normalizeChIP	16

plotChains	18
plotClassification	19
plotComponents	20
summarizeReads	21
transToTSS	22

Index	24
--------------	-----------

bayesMixModel	<i>Fits a Bayesian mixture model using Markov Chain Monte Carlo (MCMC) methods</i>
---------------	--

Description

This method estimates the posterior distribution of a Bayesian mixture model using Markov Chain Monte Carlo (MCMC) methods and calculates measures of this distribution. The mixture model may consist of normal components (with a fixed expectation of 0), exponential components and gamma components, which may be mirrored in order to model negative values.

Usage

```
bayesMixModel(z, normNull = c(), expNeg = c(), expPos = c(), gamNeg = c(), gamPos = c(), sdNormNullInit =
```

Arguments

<code>z</code>	Observed values
<code>normNull</code>	Indices of the normal components (that have $\mu = 0$).
<code>expNeg</code>	Indices of the mirrored exponential components.
<code>expPos</code>	Indices of the exponential components.
<code>gamNeg</code>	Indices of the mirrored gamma components.
<code>gamPos</code>	Indices of the mirrored components.
<code>sdNormNullInit</code>	Initial standard deviations of the normal components.
<code>rateExpNegInit</code>	Initial rates of the mirrored exponential components.
<code>rateExpPosInit</code>	Initial rates of the exponential components.
<code>shapeGamNegInit</code>	Initial shape parameters of the mirrored gamma components.
<code>scaleGamNegInit</code>	Initial scale parameters of the mirrored gamma components.
<code>shapeGamPosInit</code>	Initial shape parameters of the gamma components.
<code>scaleGamPosInit</code>	Initial scale parameters of the gamma components.
<code>piInit</code>	Initial weights of the components. If missing, all k components get the same initial weight 1/k.

classificationsInit	Initial classifications of the data points. If missing, all data points are assigned to class floor($k/2$) with k = number of components.
dirichletParInit	Initial parameter of Dirichlet process.
shapeDir	Prior shape parameter of Gamma distribution for parameter of Dirichlet process.
scaleDir	Prior scale parameter of Gamma distribution for parameter of Dirichlet process.
shapeNorm0	Prior shape parameter of Gamma distribution for precision of normal components.
scaleNorm0	Prior scale parameter of Gamma distribution for precision of normal components.
shapeExpNeg0	Prior shape parameter of Gamma distribution for parameter of mirrored exponential components.
scaleExpNeg0	Prior scale parameter of Gamma distribution for parameter of mirrored exponential components.
shapeExpPos0	Prior shape parameter of Gamma distribution for parameter of exponential components.
scaleExpPos0	Prior scale parameter of Gamma distribution for parameter of exponential components.
shapeGamNegAlpha0	Prior shape parameter of Gamma distribution for shape parameter of mirrored Gamma components.
shapeGamNegBeta0	Prior scale parameter of Gamma distribution for shape parameter of mirrored Gamma components.
scaleGamNegAlpha0	Prior shape parameter of Gamma distribution for scale parameter of mirrored Gamma components.
scaleGamNegBeta0	Prior scale parameter of Gamma distribution for scale parameter of mirrored Gamma components.
shapeGamPosAlpha0	Prior shape parameter of Gamma distribution for shape parameter of Gamma components.
shapeGamPosBeta0	Prior scale parameter of Gamma distribution for shape parameter of Gamma components.
scaleGamPosAlpha0	Prior shape parameter of Gamma distribution for scale parameter of Gamma components.
scaleGamPosBeta0	Prior scale parameter of Gamma distribution for scale parameter of Gamma components.
itb	Number of iterations used for burn-in.

nmc	Number of iterations after burn-in used for analysis.
thin	Thinning value for the iterations after burn-in.
average	Way of averaging across the posterior distribution to obtain estimates of model parameters. Either <code>average="mean"</code> or <code>average="median"</code> . Note: For the allocation to components, results are given for posterior mean, median and maximum density regardless of the specification.
gammaProposalFactor	Factor used in defining the variance of the proposal distribution for the shape parameters of the Gamma distributions.
gammaShapeGrid	Grid of values in which to search for the point around which the proposal distribution is centered.

Details

The convergence of Markov chains must be assessed prior to an interpretation of results. Inspection of trace plots via `plotChains` is therefore urgently recommended. Iterations during which one of the chains has not yet reached stationarity should not be taken into account for analysis and can be excluded by setting an appropriate burn-in value `itb`. Autocorrelation between subsequent chain values can be reduced by thinning the chain, setting an appropriate value for `thin`. The number of iterations for the chains after the burn-in should be increased when the thinning is increased to ensure a sufficient number of iterations, as specified by `nmc` is used for analysis. For the histone modification ChIP-seq and gene expression microarray example data, we used `itb=2000`, `nmc=75000` and `thin=10` in the final analyses.

Value

An object of class `MixModelBayes-class` storing results, data, priors, initial values and information about convergence.

Author(s)

Martin Schaefer (martin.schaefer@udo.edu)

See Also

[plotChains](#), [MixModelBayes-class](#)

Examples

```
set.seed(1000)
z <- c(rnorm(1000, 0, 0.5), rnorm(1000, 0, 1))
mm <- bayesMixModel(z, normNull=1:2, sdNormNullInit=c(0.1, 0.2),
  piInit=c(1/2, 1/2), shapeNorm0=c(1, 1), scaleNorm0=c(1, 1),
  shapeExpNeg0=c(), scaleExpNeg0=c(),
  shapeExpPos0=c(), scaleExpPos0=c(), itb=100, nmc=500, thin=10)
mm
plotComponents(mm)
plotChains(mm, chain="pi")
```

```

z <- c(rnorm(200, 0, 1), rnorm(200, 0, 5), rexp(200, 0.1), -rexp(200, 0.1))
mm <- bayesMixModel(z, normNull=1:2, gamNeg=3, gamPos=4,
  sdNormNullInit=c(1, 1),
  shapeGamNegInit=1, scaleGamNegInit=1, shapeGamPosInit=1, scaleGamPosInit=1,
  shapeNorm0=c(1,3), scaleNorm0=c(1,3),
  shapeGamNegAlpha0=1, shapeGamNegBeta0=1,
  scaleGamNegAlpha0=1, scaleGamNegBeta0=1,
  shapeGamPosAlpha0=1, shapeGamPosBeta0=1,
  scaleGamPosAlpha0=1, scaleGamPosBeta0=1,
  itb=10, nmc=50, thin=1)
mm
plotComponents(mm)
plotChains(mm, chain="pi")

```

ChIPseqSet-class	<i>Class "ChIPseqSet"</i>
------------------	---------------------------

Description

A class for storing count data obtained from ChIP-seq experiments by counting the number of reads lying within regions. The class extends [eSet](#).

Objects from the Class

Objects can be created by calls of the form `new("ChIPseqSet", assayData, phenoData, featureData, experimentData)`. However, usually objects are generated by calling [summarizeReads](#).

Slots

assayData: Object of class "AssayData". Contains a matrix with the number of ChIP-seq reads.

phenoData: Object of class "AnnotatedDataFrame". Contains experimenter-supplied variables describing sample (i.e., columns in assayData) phenotypes.

featureData: Object of class "AnnotatedDataFrame". Contains variables describing features (i.e., rows in assayData) unique to this experiment.

experimentData: Object of class "MIAxE". Contains details of experimental methods.

annotation: Object of class "character". Label associated with the annotation package used in the experiment.

protocolData: Object of class "AnnotatedDataFrame". Contains ChIP-seq equipment-generated variables describing sample (i.e., columns in assayData) phenotypes.

.___classVersion__: Object of class "Versions". Describes the R and Biobase version numbers used to create the instance. Intended for developer use.

Extends

Class "[eSet](#)", directly. Class "[VersionedBiobase](#)", by class "eSet", distance 2. Class "[Versioned](#)", by class "eSet", distance 3.

Methods

chipVals<- signature(object = "ChIPseqSet", value = "matrix"): Set element named chipVals in the AssayData-class slot.

chipVals signature(object = "ChIPseqSet"): Access element named chipVals in the AssayData-class slot.

Author(s)

Hans-Ulrich Klein (h.klein@uni-muenster.de)

See Also

[summarizeReads](#) linkS4class{eSet}

Examples

```
showClass("ChIPseqSet")
```

eSet

Example gene expression data set.

Description

The [ExpressionSet](#) stores 2 replicates for each of two different conditions. Data were obtained from Affymetrix MouseGene 1.0 ST arrays.

Usage

```
data(eSet)
```

Format

An object of class [ExpressionSet](#).

Details

The example data contains a subset of 200 probesets located on chromosome 1. Data were RMA normalized.

Examples

```
data(eSet)
eSet
pData(eSet)
```

integrateData	<i>Calculates a normalized correlation score from ChIP-seq and microarray gene expression data.</i>
---------------	---

Description

This function calculates the product of the standardized differences between two conditions in ChIP-seq data and the respective standardized differences in gene expression data. A score close to zero means that there are no (large) differences in at least one of the two data sets. If the score is positive, equally directed differences exist in both data sets. In case of a negative score, differences have unequal signs in the two data sets.

Usage

```
integrateData(expr, chipseq, factor, reference)
```

Arguments

expr	An ExpressionSet holding the gene expression data.
chipseq	A ChIPseqSet holding the ChIP-seq data.
factor	A character giving the name of the factor that describes the conditions to be compared. The factor must be present in the pheno data slot of the objects expr and chipseq. Further, the factor must have exactly two levels and the level names must be the same in both objects.
reference	Optionally, the name of the factor level that should be used as reference. If missing, the first level of factor in the object expr is used.

Details

Let A and B denote the gene expression value of one probe in the group of interest and in the reference group defined by the argument reference. And let X and Y be the ChIP-seq values assigned to that probe. This functions returns for each probe

$$Z = (A - B)/\sigma_{ge} \times (X - Y)/\sigma_{chip},$$

where σ_{ge} is the standard deviation estimated from all observed difference in the gene expression data and σ_{chip} the standard deviation in the ChIP-seq data.

If there is more than one sample in any group and data set, the average of the replicates is calculated first and than plugged into the formula above.

Not all features in expr must also be in chipseq and vice versa. Features present in only one of the two data types are omitted.

Value

A matrix with five columns. The first 4 columns store the (average) expression values and the (average) ChIP-seq values for each of the two conditions. The fifth column stores the correlation score. The row names equal common feature names of expr and chipseq.

Author(s)

Hans-Ulrich Klein (h.klein@uni-muenster.de)

See Also

[summarizeReads](#) [normalizeChIP](#)

Examples

```
ge <- matrix(c(5,12,5,11,11,10,12,11), nrow=2)
row.names(ge) <- c("100_at", "200_at")
colnames(ge) <- c("c1", "c2", "t1", "t2")
geDf <- data.frame(status=c("control", "control", "treated", "treated"),
  row.names=colnames(ge))
eSet <- ExpressionSet(ge, phenoData=AnnotatedDataFrame(geDf))

chip <- matrix(c(10,20,20,22), nrow=2)
row.names(chip) <- c("100_at", "200_at")
colnames(chip) <- c("c", "t")
chipDf <- data.frame(status=c("control", "treated"),
  totalCount=c(100, 100),
  row.names=colnames(chip))
cSet <- new("ChIPseqSet", chipVals=chip,
  phenoData=AnnotatedDataFrame(chipDf))

integrateData(eSet, cSet, factor="status", reference="control")
```

mappedReads

Mapped reads obtained from a anti-histone ChIP-seq experiment.

Description

The [GRangesList](#) contains two elements: "CEBPA_WT_1" and "CEBPA_KO_1". Both list elements are [GRanges](#) objects storing mapped reads from anti-H3K4me3 ChIP-seq experiments. The first sample was a wild-type mouse cell line. The second sample was obtained from the same cell line after CEPBA knock-out.

Usage

```
data(mappedReads)
```

Format

A [GRangesList](#) with two [GRanges](#).

Details

Duplicated reads and reads mapping to more than one genomic location were removed. Reads were extended to the estimated DNA fragment size of 200bp towards the 3 prime end. Further, only reads lying within certain regions of chromomse 1 were kept to reduce storage space.

Examples

```
data(mappedReads)
names(mappedReads)
mappedReads[[1]]
```

matchProbeToPromoter *A function assigning promoter regions to given probe IDs.*

Description

This function returns a GRangesList object assigning promoter regions to probes. The assignment of transcripts to probes and the transcriptional start sites must be given as arguments.

Usage

```
matchProbeToPromoter(probeToTranscript, transcriptToTSS, promWidth, mode)
```

Arguments

probeToTranscript	A list with character vectors as elements. The elements' names are probe IDs and the character vectors store the transcript IDs assigned to that probe.
transcriptToTSS	A data.frame with four columns: <ol style="list-style-type: none"> 1. Transcript ID as given in the argument probeToTranscript 2. Chromosome 3. Transcriptional start site in base pairs 4. Strand
promWidth	Width of the promoter regions in base pairs. Promoters are defined as promWidth base pairs upstream of the transcriptional start site. (default 4000bp)
mode	How probes with multiple transcripts should be handled. Must be either "union", "keepAll" or "dropMultiple". (default "union")

Details

More than one transcript can be assigned to one probe in the given probeToTranscript argument. Several options how to handle such cases can be chosen by argument mode. "union": The union of all promoters is calculated and assigned to the probe. "keepAll": All promoters of all transcripts are assigned to the probe. If some transcript have identical TSSs, the same promoter region occurs several times. "dropMultiple": All probes that have more than one transcript with different TSS are removed.

The argument transcriptToTSS must have at least 4 columns giving the information as described above. The column names are not decisive, but their position.

Value

An object of class `GRangesList` with one element for each probe. If `mode` is not set to "dropMultiple", `GRanges` may consist of more than one range. The names of the lists' elements are the probe IDs and additionally, each `GRanges` has a meta data column "probe" giving the corresponding probe ID.

Author(s)

Hans-Ulrich Klein (h.klein@uni-muenster.de)

See Also

[summarizeReads](#)

Examples

```
probeToTrans <- list("101"="ENST00011",
                    "102"=c("ENST00021", "ENST00022"),
                    "103"=NA)
transToTSS <- data.frame(
  transID=c("ENST00011", "ENST00021", "ENST00022"),
  chr=c("1", "1", "1"),
  tss=c(100000, 200000, 201000),
  strand=c("-", "+", "+"))

matchProbeToPromoter(probeToTrans, transToTSS,
  promWidth=4000, mode="union")
matchProbeToPromoter(probeToTrans, transToTSS,
  promWidth=4000, mode="keepAll")
```

MixModel-class

Class "MixModel"

Description

This class stores a fitted mixture model.

Objects from the Class

A virtual Class: No objects may be created from it.

Slots

`mmData`: Object of class "numeric" storing the data.

`configuration`: Object of class "list" storing configuration. See notes for details.

`results`: Object of class "list" storing results. See notes for details.

Methods

- classification** signature(object = "MixModel", method = "character"): Assess classification results.
- classification** signature(object = "MixModel", method = "missing"): Assess classification results.
- components** signature(object = "MixModel"): Assess mixture components.
- mmData** signature(object = "MixModel"): Assess data.
- dim** signature(x = "MixModel"): Assess dimension, i.e. numer of data points and number of components.
- length** signature(x = "MixModel"): Number of data points.
- listClassificationMethods** signature(object = "MixModel"): List available classification methods.
- show** signature(object = "MixModel"): Print an object of MixModel on screen.
- summary** signature(object = "MixModel"): Returns a list of data frames summarizing the parameter estimations for each component.
- weights** signature(object = "MixModel"): Asses the components weights.

Note

Slots configuration and results are lists with named elements. The following elements make up the minimum set of element that must be present. Depending on the method that was used to fit the mixture model, more elements may be present.

Slot configuration has at least one element.

1. `inits`A list with at least two elements: `component` and `pi`. `components` contains a list of objects of [MixtureComponent-class](#) storing the inital parameters of the mixture components. `pi` is a vector storing the initial components' weights.

Slot results has at least three elements.

1. `components`A list of objects of [MixtureComponent-class](#) storing the fitted mixture components.
2. `pi`A numeric vector holding the estimated components' weights.
3. `classification`A list of numeric vectors of the same length as data storing the classification results.

Author(s)

Hans-Ulrich Klein (h.klein@uni-muenster.de)

See Also

[m1MixModel](#) [bayesMixModel](#) [MixModelML](#) [MixModelBayes](#)

Examples

```
showClass("MixModel")
```

MixModelBayes-class *Class "MixModelBayes"*

Description

This class stores a Bayesian mixture model fitted by MCMC methods.

Objects from the Class

Objects can be created by calls of the form `new("MixModelBayes", ...)`.

Slots

chains: Object of class "list" storing the course of the Markov chains for each parameter.

mmData: Object of class "numeric" storing the data.

configuration: Object of class "list" storing configuration. See notes for details.

results: Object of class "list" storing results. See notes for details.

Extends

Class "[MixModel](#)", directly.

Methods

chains signature(object = "MixModelBayes"): Gives access to the chains slot of the object.

Note

In addition to the content described in [MixModel](#), the following elements are present: Slot configuration:

1. `initsAs` in [MixModel](#).
2. `priorsA` list specifying the prior distributions for the parameters of the components and the parameter of the Dirichlet process.
3. `chainA` list with the technical specifications for the Markov Chains.

Slot results is exactly like in [MixModel](#). Slot chains:

1. `componentsA` list giving the values for the parameters of the components in each iteration after burn-in and application of thinning.
2. `piA` matrix giving the values for the weights π of the components in each iteration after burn-in and application of thinning.
3. `dirichletParameterA` vector giving the values for dirichlet Parameter in each iteration after burn-in and application of thinning.
4. `classificationA` matrix giving the number of genes classified to each components in each iteration after burn-in and application of thinning.

Author(s)

Hans-Ulrich Klein (h.klein@uni-muenster.de)

See Also

[bayesMixModel](#) [MixModel](#)

Examples

```
showClass("MixModelBayes")
```

MixModelML-class	Class "MixModelML"
------------------	--------------------

Description

This class stores a mixture model fitted by a maximum likelihood approach.

Objects from the Class

Objects can be created by calls of the form `new("MixModelML", ...)`. Usually, objects are created by [mlMixModel](#).

Slots

convergence: Object of class "list" storing information about the convergence of the EM algorithm.

mmData: Object of class "numeric" storing the data.

configuration: Object of class "list" storing configuration. See notes for details.

results: Object of class "list" storing results. See notes for details.

Extends

Class "[MixModel](#)", directly.

Methods

convergence signature(object = "MixModelML"): Access to the convergence information.

Note

In addition to the content described in [MixModel](#), the following elements are present: Slot configuration:

1. convergence A list storing the maximum number of allowed iterations. And delta log likelihood limit, that is interpreted as convergence, if the delta log likelihood falls below that limit.

Slot results is exactly like in [MixModel](#). Slot convergence:

1. iterations Number of iterations ran.
2. deltaLogLik Delta of log likelihood observed in the last iteration.
3. logLik Log likelihood of the model fit.

Author(s)

Hans-Ulrich Klein (h.klein@uni-muenster.de)

See Also

[mlMixModel](#) [MixModel](#)

Examples

```
showClass("MixModelML")
```

MixtureComponent-class

Class "MixtureComponent"

Description

A class representing a mixture component.

Objects from the Class

Objects can be created by calls of the form `new("MixtureComponent", ...)`.

Slots

name: Object of class "character" giving the name or type of the mixture component.

parameters: Object of class "list" storing the parameters of corresponding distribution.

pdf: Object of class "function" giving the pdf of the mixture component.

color: Object of class "character" giving the color of the component that is used by plotting methods.

Methods

show `signature(object = "MixtureComponent")`: A method plotting a summary of the component on screen.

Note

The element in `parameters` should be named by the argument names of `pdf` such that this call works: `do.call(object@pdf, c(list(x=data), object@parameters))`

Author(s)

Hans-Ulrich Klein (h.klein@uni-muenster.de)

See Also

[MixModel](#)

Examples

```
showClass("MixtureComponent")
```

mlMixModel

Fits a mixture model using the maximum likelihood principle.

Description

This method calculates the maximum likelihood estimations of a mixture model using the expectation-maximization (EM) algorithm. The mixture model may consists of normal components (with a fixed expectation of 0) and exponential components, which may be mirrored in order to model negative values.

Usage

```
mlMixModel(z, normNull = c(), expNeg = c(), expPos = c(), sdNormNullInit = c(), rateExpNegInit = c(), rateExpPosInit = c(), piInit = c(), maxIter = c(), tol = c())
```

Arguments

<code>z</code>	Observed values.
<code>normNull</code>	Indices of the normal components (that have $\mu = 0$).
<code>expNeg</code>	Indices of the mirrored exponential components.
<code>expPos</code>	Indices of the exponential components.
<code>sdNormNullInit</code>	Initial standard deviations of the normal components.
<code>rateExpNegInit</code>	Initial rates ("lambda") of the exponential components.
<code>rateExpPosInit</code>	Initial rates ("lambda") of the exponential components.
<code>piInit</code>	Initial weights of the components.
<code>maxIter</code>	Maximum number of iterations.
<code>tol</code>	Threshold for convergence. The minimum log likelihood gain between two iterations that must be achieved to continue.

Details

The EM algorithm is known to converge slowly in some cases and local maxima may avoid finding the optimal solution. Users should try different initial values and different convergence criteria.

The components' indices do not influence the result, but may influence the order in which components are listed or plotted by downstream methods. Indices must be successive integers from 1 to n .

Value

An object of `MixModelML-class` storing results, data, initial values and information about the convergence.

Author(s)

Hans-Ulrich Klein (h.klein@uni-muenster.de)

See Also

[MixModelML-class](#)

Examples

```
z <- c(rnorm(1000, 0, 0.5), rnorm(1000, 0, 1))
mm <- mlMixModel(z, normNull=1:2, sdNormNullInit=c(0.1, 0.2),
  pi=c(1/2, 1/2), maxIter=500, tol=0.001)
mm

z <- c(rnorm(1000, 0, 3), rnorm(1000, 0, 5), rexp(1000, 5), -rexp(1000, 5))
mm <- mlMixModel(z, normNull=1:2, expNeg=3, expPos=4,
  sdNormNullInit=c(1, 2), rateExpNegInit=8, rateExpPosInit=8,
  pi=c(1/4, 1/4, 1/4, 1/4), maxIter=500, tol=0.001)
mm
```

normalizeChIP

Normalization of ChIP-seq count data.

Description

This method implements some normalization approaches for ChIP-seq data after counting reads within regions or bins. Similar methods are often applied to RNA-seq data after counting reads within genes.

Usage

```
normalizeChIP(object, method)
```


Arguments

object	A ChIPseqSet object as generated by summarizeReads
method	Normalization method, either "scaleTotal", "scaleRegion", "scaleMedianRegion" or "quantile"

Details

The following normalization methods are implemented:

1. `scaleTotal` Samples are scaled by a factor such that all samples have the same number of reads (the median number of reads observed across all samples before normalization). All reads are used for calculating the scaling factor.
2. `scaleRegion` Samples are scaled by a factor such that all samples have the same number of reads (the median number of reads observed across all samples before normalization). In contrast to `scaleTotal`, only reads falling into the regions (genes, promoters) that were used to create the given [ChIPseqSet](#) object are used for calculating the scaling factor. Hence, the sum of all columns of the returned [ChIPseqSet](#) are equal after applying this method.
3. `scaleMedianRegion` The scaling factor s_j for the j -th sample is defined as:

$$s_j = \text{median}_i \frac{k_{ij}}{\prod_{v=1}^m k_{iv}}$$

k_{ij} is the value of region i in sample j . See Anders and Huber (2010) for details.

4. `quantileQuantile` normalization is applied to the ChIP-seq values such that each sample has the same cdf after normalization.

Value

An [ChIPseqSet-class](#) object with normalized ChIP-seq values.

Author(s)

Hans-Ulrich Klein (h.klein@uni-muenster.de)

References

Anders and Huber; Differential expression analysis for sequence count data; Genome Biology 2010, 11:R106

See Also

[summarizeReads](#)

Examples

```
cSet <- new("ChIPseqSet",
  chipVals=matrix(c(5,6,5,6,10,12,10,12), nrow=4),
  phenoData=AnnotatedDataFrame(data.frame(totalCount=c(30,20))))
chipVals(cSet)
chipVals(normalizeChIP(cSet, method="scaleMedianRegion"))
chipVals(normalizeChIP(cSet, method="quantile"))
```

plotChains

Produces trace plots for a Bayesian mixture model

Description

This function method draws trace plots for a Bayesian mixture model, e.g. visualizes the course of the Markov Chains. Inspection of the Markov Chains is important to determine convergence of the chains, which is necessary for sensible results.

Usage

```
plotChains(object, chain, component, itb = 1, thin = 1, cols, ...)
```

Arguments

object	An object of <code>MixModelBayes-class</code>
chain	A character of length one giving the name of the parameter, which chain should be plotted. Can be omitted, if component is given. Then, all parameters of the given components are plotted.
component	An integer specifying the components, which parameter chains should be plotted. Can be omitted, if chain is given. Then, all trace plots are generated for all components having the parameter specified via argument chain.
itb	Number of iterations used for burn-in. The burn-in is relative to the output of <code>bayesMixModel</code> , e.g., any burn-in specified here is added to the burn-in that was specified when calling <code>bayesMixModel</code> .
thin	Thinning value for the iterations after burn-in. The thinning is relative to the output of <code>bayesMixModel</code> , e.g., any thinning specified here multiplies by the thinning that was specified in <code>bayesMixModel</code> .
cols	Number of columns to be used in the plot. Optional, if omitted, the number of columns and rows are chosen by the method itself.
...	Further arguments passed to <code>plot</code> .

Details

The number of iterations necessary until a Markov chain reaches stationarity depends on the specific model and data. For any inference based on Markov Chain Monte Carlo methods, it is therefore necessary to inspect the convergence of Markov Chains. One way to do this is visual inspection of trace plots using this method.

If argument `main` is passed to this method, it should have as many elements as chains are plotted. Otherwise, vector `main` is recycled.

Author(s)

Hans-Ulrich Klein (h.klein@uni-muenster.de) Martin Schaefer (martin.schaefer@udo.edu)

See Also

[bayesMixModel](#), [MixModelBayes-class](#)

Examples

```
z <- c(rnorm(1000, 0, 3), rnorm(1000, 0, 5), rexp(1000, 5), -rexp(1000, 5))
mm <- bayesMixModel(z, normNull=1:2, expNeg=3, expPos=4,
  sdNormNullInit=c(1, 2), rateExpNegInit=8, rateExpPosInit=8,
  shapeNorm0=c(1, 1), scaleNorm0=c(1, 1),
  shapeExpNeg0=c(1, 1), scaleExpNeg0=c(1, 1),
  shapeExpPos0=c(1, 1), scaleExpPos0=c(1, 1),
  itb=200, nmc=1000, thin=10)
plotChains(mm, chain="pi")
plotChains(mm, component=c(2,3))
```

plotClassification *Plot classification obtained from a mixture model.*

Description

This method visualizes the assignment of data points to the mixture components of the given mixture model. The components are plotted on the y-axis and the data on the x-axis. Data points are plotted in the color of the respective mixture component.

Usage

```
plotClassification(object, method, ...)
```

Arguments

object	An object of MixModel-class .
method	Depending on the type of the mixture model (ML, Bayes), different approaches to obtain a classification are available. Also the default approach may vary.
...	Further arguments passed to plot .

Details

If method is given, it must be a valid option for method classification. E.g., if [bayesMixModel](#) was used to create the mixture model, valid options are "maxDens", "median" and "mode".

Author(s)

Hans-Ulrich Klein (h.klein@uni-muenster.de)

See Also

[MixModel-class listClassificationMethods](#)

Examples

```
z <- c(rnorm(100, 0, 10), rnorm(100, 0, 2), rexp(100, 1/2), -rexp(100, 1/2))
mm <- mlMixModel(z, normNull=1:2, expNeg=3, expPos=4,
  sdNormNullInit=c(1, 2), rateExpNegInit=c(1/2), rateExpPosInit=c(1/2),
  pi=c(1/4, 1/4, 1/4, 1/4), maxIter=50, tol=0.01)
plotClassification(mm)
```

plotComponents	<i>Plots the mixture density together with the densities of all single components.</i>
----------------	--

Description

This function plots the mixture pdf, the estimated data pdf and the weighted pdfs of all components of the given mixture model. The plot is useful to assess the fit of the model.

Usage

```
plotComponents(object, density = FALSE, ...)
```

Arguments

object	A MixModel-class object to be plotted.
density	A logical indicating whether the data distribution should be plotted as histogram (FALSE) or as density (TRUE) estimated using kernel density estimation.
...	Further arguments passed to plot .

Details

If the argument "col" is given, the first color is used for the mixture pdf. The following colors (2 to n+1) are used for the n mixture components' pdfs. If density is set to TRUE, a further color (n+2) must be given, that is used for the data pdf.

Author(s)

Hans-Ulrich Klein (h.klein@uni-muenster.de)

See Also

[MixModel-class](#)

Examples

```
z <- c(rnorm(100, 0, 1), rnorm(100, 0, 2), rexp(100, 1/2), -rexp(100, 1/2))
mm <- mlMixModel(z, normNull=1:2, expNeg=3, expPos=4,
  sdNormNullInit=c(1, 2), rateExpNegInit=c(1/2), rateExpPosInit=c(1/2),
  pi=c(1/4, 1/4, 1/4, 1/4), maxIter=50, tol=0.01)
plotComponents(mm)
```

summarizeReads	<i>Count reads lying within given regions.</i>
----------------	--

Description

This function takes reads from e.g. ChIP-seq experiments and regions, e.g. promoters of genes, and assigns the number of overlapping reads to that region. This method was written particularly with regard to histone ChIP-seq experiments. Some histone modifications mainly occur at transcriptional start sites and thus ChIP-seq values can be assigned to genes by counting the number of reads within genes' promoter regions. However, some genes may have several transcripts and hence several promoters. Different options for handling multiple promoters are implemented. This method is also useful when integrating microarray expression data and ChIP-seq data, since most array platforms are gene centric and have probes that measure several transcripts.

Usage

```
summarizeReads(object, regions, summarize)
```

Arguments

object	A GRangesList with one GRanges object for each sample storing the ChIP-seq reads. The names of the GRangesList elements are used as sample names.
regions	An object of type GRangesList storing the promoter regions. Each element can be interpreted as gene or probe that has one or more promoters. The names of the lists' elements are used as feature names.
summarize	Defines how regions with several ranges are handled. "average" means that the mean count of reads across all ranges is assigned to the region whereas "add" means that all counts are simply added (default).

Details

This function is usually applied after calling [matchProbeToPromoter](#). When [matchProbeToPromoter](#) is used with mode "union", it is recommended to use "add". If the option "keepAll" had been used, one might want to use "average".

This method uses [countOverlaps](#) and counts each read that overlaps with at least one base. If more flexibility is needed, [summarizeOverlaps](#) might be an option. In contrast to this method, [summarizeOverlaps](#) ensures that each read is counted only once, even if it overlaps with different regions.

Value

Returns a ChIPseqSet with number of rows equal to the length of regions and number of samples equal to the length of object.

Author(s)

Hans-Ulrich Klein (h.klein@uni-muenster.de)

See Also

[matchProbeToPromoter](#) [ChIPseqSet-class](#) [summarizeOverlaps](#)

Examples

```
chipSeq <- GRangesList()
chipSeq[[1]] <- GRanges(seqnames=c("1", "1", "1", "1"),
  ranges=IRanges(start=c(97900, 198200, 198600, 202500),
    end=c(98100, 198400, 198800, 202700)),
  strand=c("+", "+", "+", "+"))
chipSeq[[2]] <- GRanges(seqnames=c("1", "1", "1", "1"),
  ranges=IRanges(start=c(97900, 198200, 198600, 300000),
    end=c(98100, 198400, 198800, 300200)),
  strand=c("+", "+", "+", "+"))
names(chipSeq) = c("sample1", "sample2")

promoters <- GRanges(seqnames=c("1", "1", "1"),
  ranges=IRanges(start=c(98000, 198000, 202000),
    end=c(101999, 201999, 205999)),
  strand=c("-", "+", "+"),
  probe=c("101", "102", "102"))
promoters <- split(promoters, elementMetadata(promoters)$probe)

chipSet <- summarizeReads(chipSeq, promoters, summarize="add")
chipVals(chipSet)
```

transToTSS

A data frame with Ensemble transcript IDs and transcriptional start sites.

Description

The data frame stores Ensemble transcript IDs and respective chromosomes, transcriptional start sites and strands for *mus musculus* (mm10).

Usage

```
data(transToTSS)
```

Format

A data frame with 277 mouse transcripts with the following 4 variables:

`ensembl_transcript_id` A character giving the Ensemble transcript ID.

`chromosome_name` A character with the respective chromosome name.

`transcript_start` An integer storing the respective transcriptional start site.

`strand` An integer storing the respective strand information.

Details

Given a character vector transcripts with the Ensemble transcript IDs, a data frame like this can be obtained via biomaRt:

```
library("biomaRt") mart <- useMart("ensembl", dataset="mmusculus_gene_ensembl")
transToTSS <- getBM(attributes=c("ensembl_transcript_id", "chromosome_name", "transcript_start", "
```

Source

<http://www.ensembl.org>

Examples

```
data(transToTSS)
head(transToTSS)
```

Index

*Topic **classes**

- ChIPseqSet-class, 5
- MixModel-class, 10
- MixModelBayes-class, 12
- MixModelML-class, 13
- MixtureComponent-class, 14

*Topic **datasets**

- eSet, 6
- mappedReads, 8
- transToTSS, 22

bayesMixModel, 2, 11, 13, 18, 19

bayesMixModel, numeric-method
(bayesMixModel), 2

chains (MixModelBayes-class), 12

chains, MixModelBayes-method
(MixModelBayes-class), 12

ChIPseqSet, 7, 17

ChIPseqSet-class, 5

chipVals (ChIPseqSet-class), 5

chipVals, ChIPseqSet-method
(ChIPseqSet-class), 5

chipVals<- (ChIPseqSet-class), 5

chipVals<-, ChIPseqSet, matrix-method
(ChIPseqSet-class), 5

classification (MixModel-class), 10

classification, MixModel, character-method
(MixModel-class), 10

classification, MixModel, missing-method
(MixModel-class), 10

components (MixModel-class), 10

components, MixModel-method
(MixModel-class), 10

convergence (MixModelML-class), 13

convergence, MixModelML-method
(MixModelML-class), 13

countOverlaps, 21

dim, MixModel-method (MixModel-class), 10

eSet, 5, 6

ExpressionSet, 6, 7

GRanges, 8

GRangesList, 8

integrateData, 7

integrateData, ExpressionSet, ChIPseqSet, character, character
(integrateData), 7

integrateData, ExpressionSet, ChIPseqSet, character, missing-m
(integrateData), 7

integrateData, ExpressionSetIllumina, ChIPseqSet, character, c
(integrateData), 7

integrateData, ExpressionSetIllumina, ChIPseqSet, character, m
(integrateData), 7

length, MixModel-method
(MixModel-class), 10

listClassificationMethods, 19

listClassificationMethods
(MixModel-class), 10

listClassificationMethods, MixModel-method
(MixModel-class), 10

mappedReads, 8

matchProbeToPromoter, 9, 21, 22

matchProbeToPromoter, list, data.frame, missing, character-met
(matchProbeToPromoter), 9

matchProbeToPromoter, list, data.frame, missing, missing-metho
(matchProbeToPromoter), 9

matchProbeToPromoter, list, data.frame, numeric, character-met
(matchProbeToPromoter), 9

matchProbeToPromoter, list, data.frame, numeric, missing-metho
(matchProbeToPromoter), 9

MixModel, 12–15

MixModel-class, 10

MixModelBayes, 11

MixModelBayes-class, 12

MixModelML, 11

MixModelML-class, 13

MixtureComponent-class, [14](#)
mlMixModel, [11](#), [13](#), [14](#), [15](#)
mlMixModel, numeric-method (mlMixModel),
[15](#)
mmData (MixModel-class), [10](#)
mmData, MixModel-method
(MixModel-class), [10](#)

normalizeChIP, [8](#), [16](#)
normalizeChIP, ChIPseqSet, character-method
(normalizeChIP), [16](#)

plot, [18–20](#)
plotChains, [4](#), [18](#)
plotChains, MixModelBayes-method
(plotChains), [18](#)
plotClassification, [19](#)
plotClassification, MixModel-method
(plotClassification), [19](#)
plotComponents, [20](#)
plotComponents, MixModel-method
(plotComponents), [20](#)

show, MixModel-method (MixModel-class),
[10](#)
show, MixtureComponent-method
(MixtureComponent-class), [14](#)
summarizeOverlaps, [21](#), [22](#)
summarizeReads, [5](#), [6](#), [8](#), [10](#), [17](#), [21](#)
summarizeReads, GRangesList, GRangesList, character-method
(summarizeReads), [21](#)
summarizeReads, GRangesList, GRangesList, missing-method
(summarizeReads), [21](#)
summary, MixModel-method
(MixModel-class), [10](#)

transToTSS, [22](#)

Versioned, [5](#)
VersionedBiobase, [5](#)

weights, MixModel-method
(MixModel-class), [10](#)