Package 'BindingSiteFinder'

August 12, 2022

Type Package

Title Binding site defintion based on iCLIP data

Version 1.3.0

Description Precise knowledge on the binding sites of an RNA-binding protein (RBP) is key to understand (post-) transcriptional regulatory processes. Here we present a workflow that describes how exact binding sites can be defined from iCLIP data. The package provides functions for binding site definition and result visualization. For details please see the vignette.

License Artistic-2.0

Encoding UTF-8

VignetteBuilder knitr

Imports tidyr, plyr, matrixStats, stats, ggplot2, methods, rtracklayer, S4Vectors, ggforce

Depends GenomicRanges, R (>= 4.1)

Suggests testthat, BiocStyle, knitr, rmarkdown, dplyr, GenomicAlignments, ComplexHeatmap, GenomeInfoDb, forcats, scales

RoxygenNote 7.1.1

Collate 'AllClasses.R' 'AllGenerics.R' 'Functions.R' 'methods.R' 'bindingsites.R' 'helper.R' 'PlotFunction.R' 'CoverageFunctions.R'

biocViews Sequencing, GeneExpression, GeneRegulation, FunctionalGenomics, Coverage, DataImport

BugReports https://github.com/ZarnackGroup/BindingSiteFinder/issues

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

git_branch master

git_last_commit 251ad52

git_last_commit_date 2022-04-26

Date/Publication 2022-08-12

Author Mirko Brüggemann [aut, cre] (<https://orcid.org/0000-0002-1778-0248>),
Kathi Zarnack [aut] (<https://orcid.org/0000-0003-3527-3378>)

Maintainer Mirko Brüggemann <mirko.brueggemann@mail.de>

R topics documented:

`annotateWithScore` *Annotation function for BSFDataSet object*

Description

This function can be used to annotate a `BSFDataSet` object with merged binding sites with scores from the initial ranges (eg. PureCLIP scores).

Usage

```
annotateWithScore(object, scoreRanges)
```

Arguments

`object`	a BSFDataSet object
`scoreRanges`	a GRanges object, with numeric column named 'score'

Value

an object of class BSFDataSet with updated meta columns of the ranges

Examples

```
if (.Platform$OS.type != "windows") {
    # load data
    csFile <- system.file("extdata", "PureCLIP_crosslink_sites_example.bed",
                        package="BindingSiteFinder")
    cs = rtracklayer::import(con = csFile, format = "BED")
    clipFiles <- system.file("extdata", package="BindingSiteFinder")
    # two experimental conditions
    meta = data.frame(
    id = c(1,2,3,4),
    condition = factor(c("WT", "WT", "KD", "KD"),
    levels = c("KD", "WT")),
    clPlus = list.files(clipFiles, pattern = "plus.bw$", full.names = TRUE),
    clMinus = list.files(clipFiles, pattern = "minus.bw$",
     full.names = TRUE))
    bds = BSFDataSetFromBigWig(ranges = cs, meta = meta, silent = TRUE)

    # merge binding sites
    bds <- makeBindingSites(object = bds, bsSize = 9, minWidth = 2,
    minCrosslinks = 2, minClSites = 1)

    # annotate with original pureCLIP score
    bdsRe = annotateWithScore(bds, cs)
}
```

`BSFDataSet` *BSFDataSet object and constructors*

Description

`BSFDataSet` contains the class `GenomicRanges`, which is used to store input ranges. Alongside with the iCLIP signal in `list` structure and additional meta data as `data.frame`.

Usage

```
BSFDataSet(ranges, meta, signal, forceEqualNames = TRUE)

BSFDataSet(ranges, meta, signal, forceEqualNames = TRUE)

BSFDataSetFromBigWig(ranges, meta, silent = FALSE)
```

Arguments

`ranges`	a `GenomicRanges` with the desired ranges to process. The strand slot must be either + or -.
`meta`	a `data.frame` with at least two columns. The first column should be a unique numeric id. The second column holds sample type information, such as the condition.

`signal` a `list` with the two entries 'signalPlus' and 'signalMinus', following a special representation of `SimpleRleList` for counts per replicates (see details for more information).

`forceEqualNames` to maintain the integrity of chromosome names (TRUE/ FALSE). The option ensures that chromosome names present in the GRanges are also all present in the signal list and vice versa. Chromosomes names present in only the signal list or the ranges are removed.

`silent` suppress loading message (TRUE/ FALSE)

Details

The ranges are enforced to have to have a "+" or "-" strand annotation,"*" is not allowed. They are expected to be of the same width and a warning is thrown otherwise.

The meta information is stored as `data.frame` with at least two required columns, 'id' and 'condition'. They are used to build the unique identifier for each replicate split by '_' (eg. id = 1 and condition = WT will result in 1_WT).

The meta data needs to have the additional columns 'clPlus' and 'clMinus' to be present if `BSFDataSetFromBigWig` is called. It is used to provide the location to the iCLIP coverage files to the import function. On object initialization these files are loaded and internally represented in the signal slot of the object (see `BSFDataSet`).

The iCLIP signal is stored in a special list structure. At the lowest level crosslink counts per nucleotide are stored as `Rle` per chromosome summarized as a `SimpleRleList`. Such a list exits for each replicate and must be named by the replicate identifier (eg. 1_WT). Therefore this list contains always exactly the same number of entries as the number of replicates in the dataset. Since we handle strands initially seperated from each other this list must be given twice, once for each strand. The strand specific entries must be named 'signalPlus' and 'signalMinus'.

Value

A BSFDataSet object.

Examples

```
# load data
files <- system.file("extdata", package="BindingSiteFinder")
load(list.files(files, pattern = ".rda$", full.names = TRUE))
rng = getRanges(bds)
sgn = getSignal(bds)
mta = getMeta(bds)
bdsNew = BSFDataSet(ranges = rng, signal = sgn, meta = mta)
```

coverageOverRanges *Coverage function for BSFDataSet objects*

Description

The crosslink coverage is computed for all ranges in the the given BSFDataSet object (see BSFDataSet for details). Depending on the returnOptions the resulting coverage information is summarized, suitable for diverse computation and plotting tasks. The coverage can only be compute for objects with identical ranges.

Usage

```
coverageOverRanges(
  object,
 returnOptions = c("merge_ranges_keep_positions", "merge_replicates_per_condition",
    "merge_all_replicates", "merge_positions_keep_replicates"),
  method = "sum",
  allowNA = FALSE,
  silent = FALSE
)
```

Arguments

object	a BSFDataSet object
returnOptions	one of merge_ranges_keep_positions, merge_replicates_per_condition, merge_all_replicates, merge_positions_keep_replicates
method	sum/ mean, select how replicates/ ranges should be summarized
allowNA	TRUE/ FALSE, allow NA values in the output if input ranges are of different width
silent	TRUE/ FALSE, suppress warning messages

Details

If returnOptions is set to merge_ranges_keep_positions: Returns a matrix with ncol being the nucleotides of the ranges (equal to the width of the input ranges) and nrow being the number of replicates in the meta information.

If returnOptions is set to merge_replicates_per_condition: Returns a list of matrices. Each list corresponds to one condition set in the meta information. The matrix in each entry has ncols equal to the ranges width and nrow equal to the number of ranges. Counts per ranges and position are summed.

If returnOptions is set to merge_all_replicates: Returns a matrix with ncols equal to the range width and nrow equal to the number of ranges. Counts per range and position are summed.

If returnOptions is set to merge_positions_keep_replicates: Returns a GRanges object where the counts are summed for each replicate and added to the original granges object.

Value

an object of class specified in `returnOptions`

Examples

```
# load data
files <- system.file("extdata", package="BindingSiteFinder")
load(list.files(files, pattern = ".rda$", full.names = TRUE))

rng = coverageOverRanges(
bds, returnOptions = "merge_ranges_keep_positions", silent = TRUE)
rng = coverageOverRanges(
bds, returnOptions = "merge_replicates_per_condition", silent = TRUE)
rng = coverageOverRanges(
bds, returnOptions = "merge_all_replicates", silent = TRUE)
rng = coverageOverRanges(
bds, returnOptions = "merge_positions_keep_replicates", silent = TRUE)
```

---

getMeta *Accessor method for the meta data of the BSFDataSet object*

Description

Meta data is stored as a `data.frame` and must contain the columns "condition", "clPlus" and "clMinus".

Usage

```
getMeta(object)

## S4 method for signature 'BSFDataSet'
getMeta(object)
```

Arguments

`object`	a BSFDataSet object

Value

returns the meta data `data.frame` with the columns "condition", "clPlus" and "clMinus".

See Also

BSFDataSet

Examples

```
# load data
files <- system.file("extdata", package="BindingSiteFinder")
load(list.files(files, pattern = ".rda$", full.names = TRUE))

getMeta(bds)
```

---

getRanges *Accessor method for the ranges of the BSFDataSet object*

Description

The ranges slot holds the genomic ranges information of the sites currently in the object. They are encoded as a GRanges object with each binding site having a single ranges entry.

Usage

```
getRanges(object)

## S4 method for signature 'BSFDataSet'
getRanges(object)
```

Arguments

object	a BSFDataSet object

Value

returns the genomic ranges (GRanges) of the associated ranges

See Also

BSFDataSet

Examples

```
# load data
files <- system.file("extdata", package="BindingSiteFinder")
load(list.files(files, pattern = ".rda$", full.names = TRUE))

getRanges(bds)
```

getSignal *Accessor method for the signal data of the BSFDataSet object*

Description

Signal data is loaded from the path specified in getMeta columns "clPlus" and "clMinus" and stored as a list of RLE lists.

Usage

```
getSignal(object)

## S4 method for signature 'BSFDataSet'
getSignal(object)
```

Arguments

object	a BSFDataSet object

Value

returns the signal data, as list of RLE list for each strand, named after the meta data columns "clPlus" and "clMinus"

See Also

getMeta BSFDataSet

Examples

```
# load data
files <- system.file("extdata", package="BindingSiteFinder")
load(list.files(files, pattern = ".rda$", full.names = TRUE))

getSignal(bds)
```

getSummary *Accessor method for the summary slot of the BSFDataSet object*

Description

The summary slot is used to track information of the filtering steps applied in the makeBindingSites function

Usage

```
getSummary(object, ...)

## S4 method for signature 'BSFDataSet'
getSummary(object)
```

Arguments

object	a BSFDataSet object
...	additional arguments

Value

returns the summary information storted in the summary slot after makeBindingSites was run

See Also

BSFDataSet makeBindingSites

Examples

```
# load data
files <- system.file("extdata", package="BindingSiteFinder")
load(list.files(files, pattern = ".rda$", full.names = TRUE))

bds <- makeBindingSites(object = bds, bsSize = 9, minWidth = 2,
minCrosslinks = 2, minClSites = 1)

getSummary(bds)
```

---

makeBindingSites *Define equally sized binding sites from peak calling results and iCLIP crosslink events.*

Description

This function performs the merging of single nucleotide crosslink sites into binding sites of a user defined width (bsSize). Depending on the desired output width crosslink sites with a distance closer than bsSize -1 are concatenated. Initially all input regions are concatenated and then imperatively merged and extended. Concatenated regions smaller than minWidth are removed prior to the merge and extension routine. This prevents outlier crosslink pileup, eg. mapping artifacts to be integrated into the final binding sites. All remaining regions are further processed and regions larger than the desired output width are interactively split up by setting always the position with the highest number of crosslinks as center. Regions smaller than the desired width are symmetrically extended. Resulting binding sites are then filtered by the defined constraints.

Usage

```
makeBindingSites(
  object,
  bsSize,
  minWidth = 3,
  minCrosslinks = 2,
  minClSites = 1,
  centerIsClSite = TRUE,
  centerIsSummit = TRUE,
  sub.chr = NA
)
```

Arguments

`object`	a BSFDataSet object (see `BSFDataSet`)
`bsSize`	an odd integer value specifying the size of the output binding sites
`minWidth`	the minimum size of regions that are subjected to the iterative merging routine, after the initial region concatenation.
`minCrosslinks`	the minimal number of positions to overlap with at least one crosslink event in the final binding sites
`minClSites`	the minimal number of crosslink sites that have to overlap a final binding site
`centerIsClSite`	logical, whether the center of a final binding site must be covered by an initial crosslink site
`centerIsSummit`	logical, whether the center of a final binding site must exhibit the highest number of crosslink events
`sub.chr`	chromosome identifier (eg, chr1, chr2) used for subsetting the BSFDataSet object. This option can be used for testing different parameter options

Details

The `bsSize` argument defines the final output width of the merged binding sites. It has to be an odd number, to ensure that a binding site has a distinct center.

The `minWidth` parameter is used to describe the minimum width a ranges has to be after the initial concatenation step. For example: Consider bsSize = 9 and minWidth = 3. Then all initial crosslink sites that are closer to each other than 8 nucleotides (bsSize -1) will be concatenated. Any of these ranges with less than 3 nucleotides of width will be removed, which reflects about 1/3 of the desired binding site width.

The argument `minCrosslinks` defines how many positions of the binding sites are covered with at least one crosslink event. This threshold has to be defined in conjunction with the binding site width. A default value of 3 with a binding site width of 9 means that 1/3 of all positions in the final binding site must be covered by a crosslink event. Setting this filter to 1 deactivates it.

The `minClSites` argument defines how many positions of the binding site must have been covered by the original crosslink site input. If the input was based on the single nucleotide crosslink positions computed by PureCLIP than this filter checks for the number of positions originally identified by PureCLIP in the computed binding sites. The default of `minClSites` = 1 essentially deactivates this filter. Setting this filter to 1 deactivates it.

The options centerIsClSite and centerIsSummit ensure that the center of each binding site is covered by an initial crosslink site and represents the summit of crosslink events in the binding site, respectively.

The option sub.chr allows to run the binding site merging on a smaller subset (eg. "chr1") for improoved computational speed when testing the effect of various binding site width and filtering options.

Value

an object of type BSFDataSet with modified ranges

Examples

```
# load data
files <- system.file("extdata", package="BindingSiteFinder")
load(list.files(files, pattern = ".rda$", full.names = TRUE))

# standard options, no subsetting
bds <- makeBindingSites(object = bds, bsSize = 9, minWidth = 2,
minCrosslinks = 2, minClSites = 1)

# standard options, with subsetting
bds <- makeBindingSites(object = bds, bsSize = 9, minWidth = 2,
minCrosslinks = 2, minClSites = 1, sub.chr = "chr22")
```

---

mergeSummaryPlot *Plot summarized results of the different binding site merging and filtering steps*

Description

Bar charts produced for the different filter steps in the binding site merging routine. Depending on the selected option (select) all or only a user defined filter can be shown.

Usage

```
mergeSummaryPlot(
  object,
  select = c("all", "filter", "inputRanges", "minClSites", "mergeCrosslinkSites",
    "minCrosslinks", "centerIsClSite", "centerIsSummit"),
  ...
)
```

Arguments

object	a BSFDataObject, with the makeBindingSites function already run

select	one of "all", "filter", "inputRanges", "minCLSites", "mergeCrosslinkSites", "minCrosslinks", "centerIsClSite" or "centerIsSummit". Defines which parameter is selected for plotting.
...	further arguments passed to ggplot

Details

If object is a single BSFDataObject a single coverage plot will be drawn, whereas if it is a list of BSFDataObjects, then faceting is used to make a plot for each list element.

Value

a plot of type ggplot after the makeBindingSites function was run

See Also

makeBindingSites

Examples

```
# load data
files <- system.file("extdata", package="BindingSiteFinder")
load(list.files(files, pattern = ".rda$", full.names = TRUE))

# plotting a single object
bds0 <- makeBindingSites(object = bds, bsSize = 9, minWidth = 2,
minCrosslinks = 2, minClSites = 1)
mergeSummaryPlot(bds0)

# plotting mulitple obejcts
bds1 <- makeBindingSites(object = bds, bsSize = 9, minWidth = 2,
minCrosslinks = 2, minClSites = 1, sub.chr = "chr22")
bds2 <- makeBindingSites(object = bds, bsSize = 9, minWidth = 2,
minCrosslinks = 2, minClSites = 3, sub.chr = "chr22")
l = list(`1. bsSize = 3` = bds1, `2. bsSize = 9` = bds2)
rangeCoveragePlot(l, width = 20)
```

---

rangeCoveragePlot *Plot crosslink events coverage over range*

Description

A diagnostic plot function that allows to check the coverage of crosslink events over different merged regions. The coverage is shown as mean over all replicates and conditions, with a standard deviation corridor.

Usage

```
rangeCoveragePlot(object, width, name = "Coverage Plot", ...)
```

Arguments

object	a BSFDataSet, or a list of BSFDataSet
width	a numeric value that defines the plotting ranges
name	plot title
...	further arguments passed to ggplot

Details

If object is a single BSFDataObject a single coverage plot will be drawn, whereas if it is a list of BSFDataObjects, then faceting is used to make a plot for each list element.

Value

a plot of type ggplot2 displaying the crosslink coverage over the ranges of the given BSFDataSet

See Also

BSFDataSet

Examples

```
# load data
files <- system.file("extdata", package="BindingSiteFinder")
load(list.files(files, pattern = ".rda$", full.names = TRUE))

# plotting a single object
bds <- makeBindingSites(object = bds, bsSize = 9, minWidth = 2,
minCrosslinks = 2, minClSites = 1)
rangeCoveragePlot(bds, width = 20)

# plotting multiple objects
bds1 <- makeBindingSites(object = bds, bsSize = 3, minWidth = 2,
minCrosslinks = 2, minClSites = 1, sub.chr = "chr22")
bds2 <- makeBindingSites(object = bds, bsSize = 9, minWidth = 2,
minCrosslinks = 2, minClSites = 1, sub.chr = "chr22")
l = list(`1. bsSize = 3` = bds1, `2. bsSize = 9` = bds2)
rangeCoveragePlot(l, width = 20)
```

---

reproducibilityFilter *Replicate reproducibility filter function*

Description

For each replicate the number of binding sites with a certain number of crosslinks is calculated. A quantile based threshold (cutoff) is applied to each replicate. This indicates how many of the merged binding sites are supported by crosslinks from the respective replicate. Next, one can specify how many replicates need to pass the defined threshold for a binding site to be considered reproducible.

Usage

```
reproducibilityFilter(
  object,
  cutoff = 0.05,
  n.reps = 1,
  min.crosslinks = 1,
  returnType = c("BSFDataSet", "data.frame")
)
```

Arguments

object	a BSFDataSet object
cutoff	a vector of length = 1, or of length = levels(getMeta(object)$conditions) with a single number (between 0-1) indicating the quantile cutoff
n.reps	a vector of length = 1, or of length = l evels(getMeta(object)$conditions) indicating how many replicates need to meet their threshold for a binding site to be called reproducible.
min.crosslinks	numeric of length = 1, defines the lower boundary for the minimum number of crosslinks a binding site has to be supported by all replicates, regardless of the replicate specific quantile threshold
returnType	one of "BSFDataSet" or "data.frame". "BSFDataSet" is the default and "matrix" can be used for easy plotting.

Details

If cutoff is a single number then the indicated cutoff will be applied to all replicates. If it is a vector then each element in the vector is applied to all replicates of the respective condition. The order is hereby given by the levels of the condition column of the meta data (see BSFDataSet,getMeta). If the condition specific filter is applied, a meta column is added to the GRanges of the BSFDataSet object, indicating the support for each condition.

If n.reps is a single number then this number is used as treshold for all binding sites. If it is a vector then it is applied to the replicates of the respective condition (like in cutoff). This allows the application of different thresholds for experiments of different experimental conditions. If the condition specific filter is applied, a meta column is added to the GRanges of the BSFDataSet object, indicating the support for each condition.

Value

an object of type BSFDataSet

Examples

```
# load data
files <- system.file("extdata", package="BindingSiteFinder")
load(list.files(files, pattern = ".rda$", full.names = TRUE))

# merge binding sites
bds <- makeBindingSites(object = bds, bsSize = 9, minWidth = 2,
```

```
minCrosslinks = 2, minClSites = 1)

# use default return with single threshold
s = reproducibilityFilter(bds, cutoff = c(0.05), n.reps = c(3))

# use default return with condition specific threshold
s = reproducibilityFilter(bds, cutoff = c(0.1, 0.05), n.reps = c(1, 2))

# use data.frame return type for plotting
s = reproducibilityFilter(bds, cutoff = c(0.1, 0.05), n.reps = c(1, 2),
returnType = "data.frame")
library(ComplexHeatmap)
m = make_comb_mat(s)
UpSet(m)
```

`reproducibiliyCutoffPlot` *Plot to that shows how many replicates support each binding site*

Description

Plotting function for settings specified in `reproducibilityFilter`.

Usage

```
reproducibiliyCutoffPlot(
  object,
  cutoff = 0.05,
  min.crosslinks = 1,
  max.range = 20,
  ...
)
```

Arguments

`object`	a BSFDataSet object
`cutoff`	a vector of length = 1, or of length = levels(meta$conditions) with a single number (between 0-1) indicating the quantile cutoff
`min.crosslinks`	numeric of length = 1, defines the lower boundary for the minimum number of crosslinks a binding site has to be supported by all replicates, regardless of the replicate specific quantile threshold
`max.range`	maximum number of crosslinks per sites that should be shown
`...`	further arguments passed to ggplot

Value

a plot of type `ggplot2` showing the per replicate reproducibility cutoffs based on a given quantile threshold

See Also

reproducibilityFilter

Examples

```
# load data
files <- system.file("extdata", package="BindingSiteFinder")
load(list.files(files, pattern = ".rda$", full.names = TRUE))

# merge binding sites
bds <- makeBindingSites(object = bds, bsSize = 9, minWidth = 2,
minCrosslinks = 2, minClSites = 1)

# use a single quantile cutoff
reproducibiliyCutoffPlot(bds, max.range = 20, cutoff = c(0.05))

# use condition specific quantile cutoffs
reproducibiliyCutoffPlot(bds, max.range = 20, cutoff = c(0.1, 0.05))
```

---

setRanges *Setter method for the ranges of the BSFDataSet object The GRanges object that holds the genomic ranges information can be replaced.*

Description

Setter method for the ranges of the BSFDataSet object The GRanges object that holds the genomic ranges information can be replaced.

Usage

```
setRanges(object, ...)

## S4 method for signature 'BSFDataSet'
setRanges(object, newRanges)
```

Arguments

object	a BSFDataSet object
...	additional arguments
newRanges	an object of type GRanges

Value

object of type BSFDataSet with updated ranges

See Also

BSFDataSet

Examples

```
# load data
files <- system.file("extdata", package="BindingSiteFinder")
load(list.files(files, pattern = ".rda$", full.names = TRUE))

rng = getRanges(bds)
rng = rng + 10
bdsNew = setRanges(bds, rng)
```

---

setSignal *Setter method for the signal data of the BSFDataSet object*

Description

Signal data is loaded from the path specified in getMeta columns "clPlus" and "clMinus" and stored as a list of RLE lists.

Usage

```
setSignal(object, ...)

## S4 method for signature 'BSFDataSet'
setSignal(object, newSignal)
```

Arguments

object	a BSFDataSet object
...	additional arguments
newSignal	list of RLE lists

Value

an object of type BSFDataSet with updated signal

See Also

BSFDataSet

Examples

```
# load data
files <- system.file("extdata", package="BindingSiteFinder")
load(list.files(files, pattern = ".rda$", full.names = TRUE))

sgn = getSignal(bds)
sgn = lapply(sgn, function(selStrand){
   lapply(selStrand, function(chrList){
       chrList[names(chrList) == "chr22"]
   })
})
bdsNew = setSignal(bds, sgn)
```

`setSummary` *Setter method for the summary slot of the BSFDataSet object*

Description

The summary slot is used to track information of the filtering steps applied in the `makeBindingSites` function

Usage

```
setSummary(object, ...)

## S4 method for signature 'BSFDataSet'
setSummary(object, summary)
```

Arguments

`object`	a BSFDataSet object
`...`	additional arguments
`summary`	a data.frame with the summary information to be stored in `BSFDataSet`

Value

an object of type `BSFDataSet` with updated summary info

See Also

`BSFDataSet`

Examples

```
# load data
files <- system.file("extdata", package="BindingSiteFinder")
load(list.files(files, pattern = ".rda$", full.names = TRUE))

df = data.frame(processingStep = c(1,2),
parameter = c(3,4))
bds = setSummary(bds, df)
```

---

show *Show method to for the BSFDataSet*

Description

Prints the information for each of the slots in the BSFDataSet object. Ranges of the getRanges slot are shown, as well as the number of crosslinks per strand getSignal and the levels of the experimental conditions (getMeta).

Usage

```
## S4 method for signature 'BSFDataSet'
show(object)
```

Arguments

object	a BSFDataSet object

Value

shows the current object state

See Also

BSFDataSet

Examples

```
# load data
files <- system.file("extdata", package="BindingSiteFinder")
load(list.files(files, pattern = ".rda$", full.names = TRUE))

show(bds)
```

supportRatio *Support ratio function for BSFDataSet objects*

Description

Functions that computes a ratio to determine how well a given binding site with is supported by the crosslink coverage of the data. For a given BSFDataSet object binding sites are computed for each width indicated in the bsWidths vector (using the coverageOverRanges function). These coverages are compared to the coverage of regions flanking the binding sites. If not indicated in bsFlank these regions are of the same width as the binding sites.

Usage

```
supportRatio(object, bsWidths, bsFlank = NA, ...)
```

Arguments

object	a BSFDataSet object
bsWidths	a numeric vector indicating the different binding site width to compute the ratio for
bsFlank	optional; a numeric vector of the same length as bsWidth used to specify the width of the flanking regions
...	further arguments passed to makeBindingSites

Details

Testing the width of 3nt for example, would result in a coverage within all 3nt wide binding sites (c1) and a coverage computed on the adjacent 3nt flanking the binding sites up- and downstream (f1, f2). Based on these numbers the ratio is computed by: c1/(1/2(f1+f2)).

The median over all ratios is reported as representative value.

Value

an object of class data.frame

Examples

```
# load data
files <- system.file("extdata", package="BindingSiteFinder")
load(list.files(files, pattern = ".rda$", full.names = TRUE))

supportRatio(bds, bsWidths = c(3,7))
```

supportRatioPlot *Plot that shows the binding site support ratio*

Description

Function that shows a ratio to determine how well a given binding site with is supported by the crosslink coverage of the data. Ratios are computed using the supportRatio function.

Usage

```
supportRatioPlot(object, bsWidths, bsFlank = NA, ...)
```

Arguments

object	a BSFDataSet object
bsWidths	a numeric vector indicating the different binding site width to compute the ratio for
bsFlank	optional; a numeric vector of the same length as bsWidth used to specify the width of the flanking regions
...	further arguments passed to makeBindingSites

Details

The higher the ratio, the more does the given binding site width captures the enrichment of crosslinks compared the the local surrounding. A ratio equal to 1 would mean no enrichemnt at all.

Value

an object of class ggplot2

Examples

```
# load data
files <- system.file("extdata", package="BindingSiteFinder")
load(list.files(files, pattern = ".rda$", full.names = TRUE))

supportRatioPlot(bds, bsWidths = c(3,7),
minWidth = 1, minClSites = 1, minCrosslinks = 2)
```

Index