Package 'FourCSeq'

October 16, 2018

Type Package

Title Package analyse 4C sequencing data

Version 1.14.0

Date 2015-06-16

Author Felix A. Klein, EMBL Heidelberg

Maintainer Felix A. Klein <daggoth@gmx.de>

Depends R (>= 3.0), GenomicRanges, ggplot2, DESeq2 (>= 1.9.11), splines, methods, LSD

Imports DESeq2, Biobase, Biostrings, GenomicRanges, SummarizedExperiment, Rsamtools, ggbio, reshape2, rtracklayer, fda, GenomicAlignments, gtools, Matrix

Suggests BiocStyle, knitr, TxDb.Dmelanogaster.UCSC.dm3.ensGene

VignetteBuilder knitr

Description FourCSeq is an R package dedicated to the analysis of (multiplexed) 4C sequencing data. The package provides a pipeline to detect specific interactions between DNA elements and identify differential interactions between conditions. The statistical analysis in R starts with individual bam files for each sample as inputs. To obtain these files, the package contains a python script (extdata/python/demultiplex.py) to demultiplex libraries and trim off primer sequences. With a standard alignment software the required bam files can be then be generated.

License GPL (>= 3)

biocViews Software, Preprocessing, Sequencing

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

git_branch RELEASE_3_7

git_last_commit 4fb923f

git_last_commit_date 2018-04-30

Date/Publication 2018-10-15

R topics documented:

`addFragments` *Add the restriction fragment information*

Description

`addFragments` adds the genomic position and information of the restriction fragments as `GRanges` object to `rowRanges` of the `FourC` object.

Usage

```
addFragments(object, minSize = 20, filter = TRUE, save = FALSE)
```

Arguments

`object`	A `FourC` object.
`minSize`	Minimum size of a restriction fragment end to be valid. Default is `20` bases.
`filter`	Defines whether fragments that do not contain a cutting site of the second restriction enzyme or are smaller than `minSize` should be filtered out.
`save`	Defines if the fragment information should be saved as txt and bed files in the fragmentDir folder of the projectPath.

Value

Updated FourC object that contains the information about the restriction fragments in rowRanges.

Author(s)

Felix A. Klein, <felix.klein@embl.de>

See Also

FourC, findViewpointFragments

Examples

```
metadata <- list(projectPath=tempdir(),
                 fragmentDir="re_fragments",
                 referenceGenomeFile=system.file("extdata/dm3_chr2L_1-6900.fa",
                                                 package="FourCSeq"),
                 reSequence1="GATC",
                 reSequence2="CATG",
                 primerFile=system.file("extdata/primer.fa",
                                        package="FourCSeq"),
                 bamFilePath=system.file("extdata/bam", package="FourCSeq"))

colData <- DataFrame(viewpoint = "testdata",
                     condition = factor(rep(c("WE_68h", "MESO_68h", "WE_34h"),
                                            each=2),
                                        levels = c("WE_68h", "MESO_68h", "WE_34h")),
                     replicate = rep(c(1, 2),
                                     3),
                     bamFile = c("CRM_ap_ApME680_WE_6-8h_1_testdata.bam",
                                 "CRM_ap_ApME680_WE_6-8h_2_testdata.bam",
                                 "CRM_ap_ApME680_MESO_6-8h_1_testdata.bam",
                                 "CRM_ap_ApME680_MESO_6-8h_2_testdata.bam",
                                 "CRM_ap_ApME680_WE_3-4h_1_testdata.bam",
                                 "CRM_ap_ApME680_WE_3-4h_2_testdata.bam"),
                     sequencingPrimer="first")

fc <- FourC(colData, metadata)
fc

fc <- addFragments(fc)
fc
```

---

addPeaks *Add peaks based on z-scores and adjusted p-values*

Description

Fragments are annotated to be a z-score peak (TRUE or FALSE) based on z-scores and fdr values for the given fragment

Usage

```
addPeaks(object, zScoreThresh = 2, fdrThresh = 0.1)
```

Arguments

object	FourC object for which z-scores have been calculated.
zScoreThresh	Threshold on z-score that has to met by all replicates of a condition.
fdrThresh	Threshold on adjusted p-value that has to be met in at least one replicated of a condition.

Value

FourC object with an additional assay 'peaks' of boolean values. The parameters of the call to addPeaks are stored as DataFrame peakParameter in the metadata slot.

Author(s)

Felix A. Klein, <felix.klein@embl.de>

data(fcf, package="FourCSeq")

fcf <- addPeaks(fcf)

---

addViewpointFrags *Add the information of the viewpoint fragments*

Description

addViewpointFrags adds the genomic position and information of the viewpoint primer fragments to colData of the FourC object.

Usage

```
addViewpointFrags(object, primerFragFile)
```

Arguments

object	A FourC object.
primerFragFile	character string defining the file that contains the information about the primer fragments. Defaults to "primerFragments.rda" which is the default output file of findViewpointFragments in the provided fragmentDir.

Value

Updated FourC object with information about the viewpoint fragments added to colData.

Author(s)

Felix A. Klein, <felix.klein@embl.de>

See Also

FourC, findViewpointFragments

Examples

```
metadata <- list(projectPath=tempdir(),
                 fragmentDir="re_fragments",
                 referenceGenomeFile=system.file("extdata/dm3_chr2L_1-6900.fa",
                                                 package="FourCSeq"),
                 reSequence1="GATC",
                 reSequence2="CATG",
                 primerFile=system.file("extdata/primer.fa",
                                        package="FourCSeq"),
                 bamFilePath=system.file("extdata/bam", package="FourCSeq"))

colData <- DataFrame(viewpoint = "testdata",
                     condition = factor(rep(c("WE_68h", "MESO_68h", "WE_34h"),
                                            each=2),
                                        levels = c("WE_68h", "MESO_68h", "WE_34h")),
                     replicate = rep(c(1, 2),
                                     3),
                     bamFile = c("CRM_ap_ApME680_WE_6-8h_1_testdata.bam",
                                 "CRM_ap_ApME680_WE_6-8h_2_testdata.bam",
                                 "CRM_ap_ApME680_MESO_6-8h_1_testdata.bam",
                                 "CRM_ap_ApME680_MESO_6-8h_2_testdata.bam",
                                 "CRM_ap_ApME680_WE_3-4h_1_testdata.bam",
                                 "CRM_ap_ApME680_WE_3-4h_2_testdata.bam"),
                     sequencingPrimer="first")

fc <- FourC(colData, metadata)
fc

fc <- addFragments(fc)

findViewpointFragments(fc)

fc <- addViewpointFrags(fc)
fc
```

`combineFragEnds` *Combine the counts of both fragment ends.*

Description

`combineFragEnds` combines the counts of both fragment ends. A multiplication factor can be used for fragments that only have counts for one valid fragment end.

Usage

```
combineFragEnds(object, multFactor = 1, filter = FALSE)
```

Arguments

`object`	A `FourC` object.
`multFactor`	Multiplication factor that can be used to multiply the counts of fragments which only have one valid end. Default is `1`. `filter` is automatically set to `TRUE` if `multFactor` is different from 1.

filter	If filter is TRUE, only reads from valid fragment ends are summed up for each fragment. This means if only one fragment end is valid only counts from this end are considered. If filter is FALSE, counts from both fragment ends are summed up without any filtering.

Value

Returns an updated FourC object with a new assay counts containing the combined count data of both fragment ends for all viewpoints.

Author(s)

Felix A. Klein, <felix.klein@embl.de>

See Also

FourC, countFragmentOverlaps

Examples

```
data(fc, package="FourCSeq")

fc <- combineFragEnds(fc)
fc
```

---

countFragmentOverlaps *Count fragment overlaps*

Description

countFragmentOverlaps counts the number of reads mapping to each fragment end in rowRanges of the FourC object.

Usage

```
countFragmentOverlaps(object, trim=0, minMapq=0, shift=0)
```

Arguments

object	A FourC object.
trim	Number of bases that should be trimmed at the start of a read. This is necessary if the read still contains the restriction enzyme sequence. Default is 0 bases.
minMapq	Minimum mapping quality required for counting the read. Default is 0. If set to negative values the filtering step is skipped.
shift	Maximum difference in starts or ends between read and fragment positions. Default is 0.

Value

Updated FourC object that contains two new assays countsLeftFragmentEnd and countsRightFragmentEnd with the count values at the respective fragment end.

Author(s)

Felix A. Klein, <felix.klein@embl.de>

See Also

FourC, findViewpointFragments, countFragmentOverlapsSecondCutter

Examples

```
metadata <- list(projectPath=tempdir(),
                 fragmentDir="re_fragments",
                 referenceGenomeFile=system.file("extdata/dm3_chr2L_1-6900.fa",
                                                 package="FourCSeq"),
                 reSequence1="GATC",
                 reSequence2="CATG",
                 primerFile=system.file("extdata/primer.fa",
                                        package="FourCSeq"),
                 bamFilePath=system.file("extdata/bam", package="FourCSeq"))

colData <- DataFrame(viewpoint = "testdata",
                     condition = factor(rep(c("WE_68h", "MESO_68h", "WE_34h"),
                                            each=2),
                                        levels = c("WE_68h", "MESO_68h", "WE_34h")),
                     replicate = rep(c(1, 2),
                                     3),
                     bamFile = c("CRM_ap_ApME680_WE_6-8h_1_testdata.bam",
                                 "CRM_ap_ApME680_WE_6-8h_2_testdata.bam",
                                 "CRM_ap_ApME680_MESO_6-8h_1_testdata.bam",
                                 "CRM_ap_ApME680_MESO_6-8h_2_testdata.bam",
                                 "CRM_ap_ApME680_WE_3-4h_1_testdata.bam",
                                 "CRM_ap_ApME680_WE_3-4h_2_testdata.bam"),
                     sequencingPrimer="first")

fc <- FourC(colData, metadata)
fc

fc <- addFragments(fc)

findViewpointFragments(fc)

fc <- addViewpointFrags(fc)
fc

fc <- countFragmentOverlaps(fc, trim=4, minMapq=30)
fc
```

---

countFragmentOverlapsSecondCutter

Count fragment overlaps when sequencing was performed from the second cutting site

---

Description

countFragmentOverlapsSecondCutter counts the number of reads mapping to each cutting site of the second cutter and then summarizes them over the fragment ends of the first cutter stored in rowRanges of the FourC object.

Usage

```
countFragmentOverlapsSecondCutter(object, extend = TRUE, minMapq = 0,
  shift = 0)
```

Arguments

object	A FourC object.
extend	Defines whether the read start should be extended by the length of the second cutter recognition sequence to overlap the cutting sites which is required for counting. If the cutting site has been trimmed with the primer sequence this has to be set to TRUE. Default is TRUE.
minMapq	Minimum mapping quality required for counting the read. Default is 0. If set to negative values the filtering step is skipped.
shift	Maximum difference in starts or ends between read and fragment positions. Default is 0.

Value

Updated FourC object that contains two new assays countsLeftFragmentEnd and countsRightFragmentEnd with the count values at the respective fragment end.

Author(s)

Felix A. Klein, <felix.klein@embl.de>

See Also

FourC, findViewpointFragments, countFragmentOverlaps

Examples

```
## not run:
## countFragmentOverlapsSecondCutter(fc, extend=TRUE, minMapq=30)
```

---

distFitMonotone *Fit the distance dependency*

Description

distFitMonotone takes the variance stabilized count values and calculates a monotone fit for the distance dependency. The position information about the viewpoint is stored in fragData. The signal trend is fitted for the data left and right to the viewpoint separately using the fda package.

Usage

```
distFitMonotone(count, fragData, alpha = 20, penalty = 0.1,
  removeZeros = FALSE, ...)
```

Arguments

count	Matrix of variance stabilized count data in the defined interval around the viewpoint.
fragData	Data frame with all the information on restriction fragments and the interval around the viewpoint.
alpha	Approximate number of fragments between two breaks of the B-spline basis. `floor((number of fragments))/alpha)` is passed to `create.bspline.basis` as `nbasis` argument (if this value is smaller than 4, 4 is the default value for `nbasis`). Default is `20`.
penalty	Penalty term passed to `fdPar` as `lambda` argument. Default is `0.1`.
removeZeros	Defines if zero values should be removed from the fit. Default is `FALSE`.
...	Additional parameters.

Value

Returns a matrix with z-score values of all fragments around the given viewpoint.

Author(s)

Felix A. Klein, <felix.klein@embl.de>

---

distFitMonotoneSymmetric

Fit the distance dependency

Description

`distFitMonotoneSymmetric` takes the variance stabilized count values and calculates a symmetric monotone fit for the distance dependency. The position information about the viewpoint is stored in `fragData`. The signal trend is fitted for the combined data left and right to the viewpoint using the fda package.

Usage

```
distFitMonotoneSymmetric(count, fragData, alpha = 20, penalty = 0.1,
  removeZeros = FALSE, ...)
```

Arguments

count	Matrix of variance stabilized count data in the defined interval around the viewpoint.
fragData	Data frame with all the information on restriction fragments and the interval around the viewpoint.

alpha	Approximate number of fragments between two breaks of the B-spline basis. floor((max(number of fragments left or right))/alpha) is passed to create.bspline.basis as nbasis argument (if this value is smaller than 4, 4 is the default value). Default is 20.
penalty	Penalty term passed to fdPar as lambda argument. Default is 0.1.
removeZeros	Defines if zero values should be removed from the fit. Default is FALSE.
...	Additional parameters.

Value

Returns a matrix with z-score values of all fragments around the given viewpoint.

Author(s)

Felix A. Klein, <felix.klein@embl.de>

---

fc *FourC object with counts*

Description

This file contains the FourCSeq object fc. The reference fragments have been added and the overlaps of the reference fragments and aligned reads were counted. (See ?addFragments or the vignette of the 'FourCSeq' package for details).

Usage

```
data(fc)
```

Format

Formal class 'FourC' [package "FourCSeq"]

Examples

```
data(fc)
fc
```

fcf *FourC object with z-scores*

Description

This file contains the FourCSeq object fcf. For this object z-scores have been calculated. (See ?getZScores or the vignette of the 'FourCSeq' package for details).

Usage

```
data(fcf)
```

Format

Formal class 'FourC' [package "FourCSeq"]

Examples

```
data(fcf)
fcf
```

findViewpointFragments *Find the fragments to which the viewpoint primers map.*

Description

findViewpointFragments finds the position of the viewpoint primer in the reference genome and on which restriction fragment they fall. It saves the results in these files: - projectPath/fragmentDir/primerFragments.rda - projectPath/fragmentDir/primerFragments.txt

Usage

```
findViewpointFragments(object)
```

Arguments

object	A FourC object.

Author(s)

Felix A. Klein, <felix.klein@embl.de>

See Also

FourC getReferenceSeq

Examples

```
metadata <- list(projectPath=tempdir(),
                 fragmentDir="re_fragments",
                 referenceGenomeFile=system.file("extdata/dm3_chr2L_1-6900.fa",
                                                 package="FourCSeq"),
                 reSequence1="GATC",
                 reSequence2="CATG",
                 primerFile=system.file("extdata/primer.fa",
                                        package="FourCSeq"),
                 bamFilePath=system.file("extdata/bam", package="FourCSeq"))

colData <- DataFrame(viewpoint = "testdata",
                     condition = factor(rep(c("WE_68h", "MESO_68h", "WE_34h"),
                                            each=2),
                                        levels = c("WE_68h", "MESO_68h", "WE_34h")),
                     replicate = rep(c(1, 2),
                                     3),
                     bamFile = c("CRM_ap_ApME680_WE_6-8h_1_testdata.bam",
                                 "CRM_ap_ApME680_WE_6-8h_2_testdata.bam",
                                 "CRM_ap_ApME680_MESO_6-8h_1_testdata.bam",
                                 "CRM_ap_ApME680_MESO_6-8h_2_testdata.bam",
                                 "CRM_ap_ApME680_WE_3-4h_1_testdata.bam",
                                 "CRM_ap_ApME680_WE_3-4h_2_testdata.bam"),
                     sequencingPrimer="first")

fc <- FourC(colData, metadata)
fc

fc <- addFragments(fc)

findViewpointFragments(fc)

fc <- addViewpointFrags(fc)
fc
```

`FourC` *FourC-class*

Description

FourC-class

Usage

```
FourC(colData, metadata)
```

Arguments

`colData`	Column data that contains the required information for each library to set up the `FourC` object: 1. viewpoint, name of the viewpoint 2. condition, experimental condition 3. replicate, replicate number

4. bamFile, file name of the bam file
5. sequencingPrimer, was the 4C library sequenced from the side of the first restriction enzyme cutting site or second

`metadata` Experimental data information required for the `FourC` object:

1. projectPath, directory where the project will be saved
2. fragmentDir, directory in the project directory where the information about restriction fragments will be saved
3. referenceGenomeFile, path to the reference genome or a `BSgenome` object.
4. reSequence1, restriction enzyme recognition pattern of the first restriction enzyme
5. reSequence2, restriction enzyme recognition pattern of the second restriction enzyme
6. primerFile, path to the file containing the primer sequences used for preparing the 4C libraries
7. bamFilePath, path to the directory where the bam files are stored

Note

The `FourC` object extends the `DESeqDataSet` class.

Examples

```
metadata <- list(projectPath=tempdir(),
                 fragmentDir="re_fragments",
                 referenceGenomeFile=system.file("extdata/dm3_2L_1-6900.fa",
                                                 package="FourCSeq"),
                 reSequence1="GATC",
                 reSequence2="CATG",
                 primerFile=system.file("extdata/primer.fa",
                                        package="FourCSeq"),
                 bamFilePath=system.file("extdata/bam", package="FourCSeq"))

colData <- DataFrame(viewpoint = "testdata",
                     condition = factor(rep(c("WE_68h", "MESO_68h", "WE_34h"),
                                            each=2),
                                        levels = c("WE_68h", "MESO_68h", "WE_34h")),
                     replicate = rep(c(1, 2),
                                     3),
                     bamFile = c("CRM_ap_ApME680_WE_6-8h_1_testdata.bam",
                                 "CRM_ap_ApME680_WE_6-8h_2_testdata.bam",
                                 "CRM_ap_ApME680_MESO_6-8h_1_testdata.bam",
                                 "CRM_ap_ApME680_MESO_6-8h_2_testdata.bam",
                                 "CRM_ap_ApME680_WE_3-4h_1_testdata.bam",
                                 "CRM_ap_ApME680_WE_3-4h_2_testdata.bam"),
                     sequencingPrimer="first")

fc <- FourC(colData, metadata)
fc
```

getAllResults *FourCSeq analysis results*

Description

Using the DESEeq2 function results, getAllResults extracts results from a FourCSeq analysis giving base means across samples, log2 fold changes, standard errors, test statistics, p-values and adjusted p-values for all pair-wise conditions tested.

Usage

```
getAllResults(object, ...)
```

Arguments

object	FourC object for which z-scores and differences have been calculated.
...	Additional parameters that can be passed to results.

Value

DataFrame of results columns for all pairwise tests, with metadata columns of coefficient and test information

Author(s)

Felix A. Klein, <felix.klein@embl.de>

See Also

results

Examples

```
data(fcf, package="FourCSeq")

fcf <- getDifferences(fcf, referenceCondition="WE_68h")

results <- getAllResults(fcf)
results
```

getDifferences *Detect differences*

Description

getDifferences detects differences in the interaction frequencies between different conditions. For each viewpoint all possible combinations of conditions are tested using the DESeq2 package.

Usage

```
getDifferences(object, referenceCondition = NULL, fitNormFactors = TRUE)
```

Arguments

object	A FourC object.
referenceCondition	Reference condition to be used for testing.
fitNormFactors	Defines if the distance dependency should be used to estimate normalization-Factors. Default is TRUE.

Value

Returns an updated FourC object that contains the results and information from the differential testing. Normalization factors are added to the assays slot.

Author(s)

Felix A. Klein, <felix.klein@embl.de>

See Also

FourC, getZScores, estimateDispersions, nbinomWaldTest

Examples

```
data(fcf, package="FourCSeq")

fcf <- getDifferences(fcf, referenceCondition="WE_68h")

results <- getAllResults(fcf)
results
```

---

getDistAroundVp *getDistAroundVp*

Description

getDistAroundVp Add the distance information for a given viewpoint to the fragment data.

Usage

```
getDistAroundVp(vp, vpData, fragData)
```

Arguments

vp	A single character string with the selected viewpoint.
vpData	DataFrame containing the column data of the FourC object.
fragData	GRanges object containing the row data (fragment data) of the FourC object.

Details

Mid positions of the fragments are defined as start + (end - start)

Value

Updated GRanges object that contains the distance from the viewpoint for the fragments on the viewpoint chromosome.

Examples

```
data(fc, package="FourCSeq")

fragmentDataWithDistance <- getDistAroundVp("ap", colData(fc), rowRanges(fc))
```

---

getNormalizationFactors

Get normalization factors for each fragment

Description

getNormalizationFactors uses the distance dependency to calculate normalization factors for each fragment.

Usage

```
getNormalizationFactors(object)
```

Arguments

object	A FourC object.

Value

Returns an updated FourC object that contains normalization factors.

Author(s)

Felix A. Klein, <felix.klein@embl.de>

See Also

getDifferences

Examples

```
data(fcf, package="FourCSeq")

normalizationFactors <- getNormalizationFactors(fcf)
head(normalizationFactors)
```

getReferenceSeq *Function to read reference sequences*

Description

This functions allows to retrieve the reference sequence.

Usage

```
getReferenceSeq(object)
```

Arguments

object	A FourC object.

Value

A DNAStringSet object containing the sequences of the reference genome from the FaFile or BSgenome object.

Author(s)

Felix A. Klein

Examples

```
metadata <- list(projectPath=tempdir(),
                 fragmentDir="re_fragments",
                 referenceGenomeFile=system.file("extdata/dm3_chr2L_1-6900.fa",
                                                 package="FourCSeq"),
                 reSequence1="GATC",
                 reSequence2="CATG",
                 primerFile=system.file("extdata/primer.fa",
                                        package="FourCSeq"),
                 bamFilePath=system.file("extdata/bam", package="FourCSeq"))

colData <- DataFrame(viewpoint = "testdata",
                     condition = factor(rep(c("WE_68h", "MESO_68h", "WE_34h"),
                                            each=2),
                                        levels = c("WE_68h", "MESO_68h", "WE_34h")),
                     replicate = rep(c(1, 2),
                                     3),
                     bamFile = c("CRM_ap_ApME680_WE_6-8h_1_testdata.bam",
                                 "CRM_ap_ApME680_WE_6-8h_2_testdata.bam",
                                 "CRM_ap_ApME680_MESO_6-8h_1_testdata.bam",
                                 "CRM_ap_ApME680_MESO_6-8h_2_testdata.bam",
                                 "CRM_ap_ApME680_WE_3-4h_1_testdata.bam",
                                 "CRM_ap_ApME680_WE_3-4h_2_testdata.bam"),
                     sequencingPrimer="first")


fc <- FourC(colData, metadata)
fc

refSeq <- getReferenceSeq(fc)
```

getZScores *Calculate z-scores using the residuals of the general trend fit.*

Description

getZScores calculates the z-score for each fragment within a distance from the viewpoint defined by distAroundVP. For the calculation the count data is first transformed with a variance stabilizing transformation from the DESeq2 package. The decay trend of this transformed data with distance from the viewpoint is fitted either with local regression model using locfit or a monotone decay fit using the fda package. z-scores are finally calculated from the residuals of the fit values.

Usage

```
getZScores(object, removeZeros = TRUE, minCount = 40, minDist = NULL,
  fitFun = "distFitMonotoneSymmetric", sdFun = mad, ...)
```

Arguments

object	A FourC object.
removeZeros	Parameter to define if fragments with zero counts across all replicates and conditions should be removed from the calculation. Default is TRUE.
minCount	The minimum median count value across replicates, a fragment has to have to be used for the calculation of z-scores. Default is 40
minDist	Define a minimum distance to both sides of the viewpoint that is discarded for analysis. If undefined (NULL), the borders of the viewpoint peak are estimated as the minimum values next to the viewpoint before the signal rises again.
fitFun	Fit function to be used. Either distFitMonotoneSymmetric (default), distFitMonotone or any self-defined function to fit the distance dependency.
sdFun	Function to calculate the variation of the data. Default is mad.
...	Additional Parameters passed to the distance fit function specified in fitFun (e.g. distFitMonotone, distFitMonotoneSymmetric).

Value

Returns a FourC object for the selected viewpoint with z-score values for all fragments on the viewpoint chromosome that passed the minCount threshold and that were not too close to the viewpoint. All additional required data is saved in the object. Especially the following information is added to the FourC object:

1. metadata: parameters passed to the getZScore function.
2. colData: sdFun values calculated from the fit residuals
3. rowRanges: the distance information to the viewpoint and information calculated for the variance stabilizing data are added.
4. assays: variance stabilized count values (trafo), fit values (fit), z-score values (zScore), associated p-values (pValue) and adjusted p-values (pAdjusted)

Author(s)

Felix A. Klein, <felix.klein@embl.de>

See Also

FourC, countFragmentOverlaps, distFitMonotone, distFitMonotoneSymmetric

Examples

```
data(fc, package="FourCSeq")

fcf <- getZScores(fc)

fcf
```

---

normalizeRPM *Normalize count data to rpm*

Description

normalizeRPM Normalizes the counts of each experiment to RPM.

Usage

```
normalizeRPM(object, assay = "counts", normalized = "rpm",
  removeHighestValues = 0)
```

Arguments

object	A FourC object.
assay	Character vector selecting the assay of the FourC object that should be normalized.
normalized	Character vector specifying the assay name of the normalized data.
removeHighestValues	Defines the number of highest count values that should be removed before the library sizes are calculated in the denominator. normalized containing the normalized data.

Author(s)

Felix A. Klein, <felix.klein@embl.de>

See Also

FourC, combineFragEnds

Examples

```
data(fc, package="FourCSeq")

fc <- combineFragEnds(fc)
fc

fc <- normalizeRPM(fc)
fc
```

plotDifferences *Plot differences*

Description

plotDifferences generate plots to investigate the results of getDifferences

Usage

```
plotDifferences(object, plotWindows = c(1e+05, 1e+06), textsize = 20,
  diffThresh = 0.01, controls = NULL, txdb = NULL, conditionAB = NULL)
```

Arguments

object	A FourC object.
plotWindows	Window sizes around the viewpoint for which plots are generated.
textsize	Adjust text size.
diffThresh	Threshold on adjusted p-values calculated in the differential test.
controls	Auxiliary GRanges object that contains information about the viewpoint and closest gene TSS.
txdb	Auxiliary TranscriptDb object that contains gene models for the investigated organism.
conditionAB	Condition A and B, for which the comparison is plotted.

Author(s)

Felix A. Klein, <felix.klein@embl.de>

See Also

FourC, getDifferences

Examples

```
data(fcf, package="FourCSeq")

fcf <- getDifferences(fcf, referenceCondition="WE_68h")

plotDifferences(fcf)
```

plotFits *Plot fit results.*

Description

plotFits generates plots of the fits used to calculate the z-scores.

Usage

```
plotFits(object, viewpoint = NULL, main = NULL)
```

Arguments

object	A FourC object, after successfully calling getZScores.
viewpoint	A character vector of the viewpoint for which the plots are generated. If set to NULL the first viewpoint in the FourC object is used.
main	Main text for the plots. If set to NULL the column names are printed.

Details

Plots are generated to visualize the results of the fits used to calculate the z-scores.

Author(s)

Felix A. Klein, <felix.klein@embl.de>

See Also

FourC, getZScores, distFitMonotone, distFitMonotoneSymmetric

Examples

```
data(fcf, package="FourCSeq")

plotFits(fcf)
```

plotNormalizationFactors *Plot the estimated normalization factors.*

Description

plotNormalizationFactors generates plots of the fits used to calculate the z-scores.

Usage

```
plotNormalizationFactors(object, dist = TRUE, sizeFactors = TRUE)
```

Arguments

object	A FourC object, after successfully calling getZScores.
dist	If TRUE (default), the normalizationFactors are plotted as a function for distance from the viewpoint. Otherwise they are plotted as a function of ranks.
sizeFactors	If TRUE, sizeFactors are calculated and plotted as horizontal bars.

Details

Plots are generated to visualize the results of the fits used to calculate the z-scores.

Author(s)

Felix A. Klein, <felix.klein@embl.de>

See Also

FourC, getDifferences, estimateSizeFactors

Examples

```
data(fcf, package="FourCSeq")

fcf <- getDifferences(fcf)

plotNormalizationFactors(fcf)
```

---

plotScatter *plotScatter*

Description

plotScatter Plot scatter plots for all experiments with replicates.

Usage

```
plotScatter(object, assay = "counts", ...)
```

Arguments

object	A FourC object.
assay	Character vector selecting the assay of the FourC object for which the correlation between the replicates should be plotted.
...	Additional arguments passed to smoothScatter.

See Also

smoothScatter

Examples

```
data(fc, package="FourCSeq")

fc <- combineFragEnds(fc)
fc

plotScatter(fc)
```

---

plotZScores *Plot z-score results.*

Description

plotZScores generates plots to check the z-score calculation.

Usage

```
plotZScores(object, cols = NULL, plotWindows = c(1e+05, 1e+06),
  controls = NULL, textsize = 20, txdb = NULL, plotSingle = FALSE)
```

Arguments

object	A FourC object.
cols	A character or numeric vector used to subset the columns of object.
plotWindows	Window sizes around the viewpoint for which plots are generated.
controls	Auxiliary GRanges object that contains information about the viewpoint and closest gene TSS.
textsize	Plot parameter passed to the plotting function if
txdb	Auxiliary TranscriptDb object that contains gene models for the investigated organism.
plotSingle	If set to true each selected column will be plotted in a single plot.

Details

Plots are generated to visualize the results of the getZScores function

Author(s)

Felix A. Klein, <felix.klein@embl.de>

See Also

FourC, getZScores, distFitMonotone, distFitMonotoneSymmetric

Examples

```
data(fcf, package="FourCSeq")

plotZScores(fcf)
```

smoothCounts *Smooth the counts of neighboring fragments*

Description

Counts are smoothed using the number of fragments provided by binWidth. binWidth has to be an odd number so that an equal number of fragments to each side of the current fragment are used for smoothing.

Usage

```
smoothCounts(object, assay = "counts", binWidth = 5)
```

Arguments

object	A FourC object.
assay	Assay name that will be smoothed.
binWidth	Integer vector of odd numbers.

Value

Returns an updated FourC object with smoothed counts for each binWidth as new assays.

Author(s)

Felix A. Klein, <felix.klein@embl.de>

See Also

FourC

Examples

```
data(fc, package="FourCSeq")

fc <- smoothCounts(fc)
fc
```

smoothHitPerCent *Smooth the hits of neighboring fragments*

Description

Counts are transformed to a hit if they exceed the given threshold. Hits are then smoothed using the number of fragments provided by binWidth. binWidth has to be an odd number so that an equal number of fragments to each side of the current fragment are used for smoothing.

Usage

```
smoothHitPerCent(object, binWidth = 101, thresh = 1)
```

Arguments

object	A FourC object.
binWidth	Integer vector of odd numbers.
thresh	Single integer defining the threshold for calling a fragment a hit.

Value

Returns an updated FourC object with smoothed counts for each binWidth as new assays.

Author(s)

Felix A. Klein, <felix.klein@embl.de>

See Also

FourC

Examples

```
data(fc, package="FourCSeq")

fc <- smoothHitPerCent(fc)
fc
```

---

writeTrackFiles *Write track files of an selected* assay

Description

The files are saved in the specified folder. The filenames are the combination of the assay name, the selected column name and the corresponding file extension.

Usage

```
writeTrackFiles(object, assay = "counts", folder = "tracks",
  format = "bw", removeZeros = TRUE)
```

Arguments

object	A FourC object.
assay	Character vector selecting the assay of the FourC object that should be saved as track file.
folder	Path relative to the project folder, where the results are track files should be saved.
format	Character vector specifying the format of the output. Can either be 'bedGraph' or 'bw'. 'bw' is the default.
removeZeros	Define whether fragments with zero counts should be included with value 0 or not. On default zeros are removed.

Details

`writeTrackFiles`

Value

Message whether the track export of assay was successful.

Author(s)

Felix A. Klein, <felix.klein@embl.de>

Examples

```
data(fc, package="FourCSeq")
metadata(fc)$projectPath = tempdir()

fc <- combineFragEnds(fc)
fc

writeTrackFiles(fc)
```

Index