Package 'ChIC'

April 12, 2022

Title Quality Control Pipeline for ChIP-Seq Data

Version 1.14.0

Author Carmen Maria Livi

Maintainer Carmen Maria Livi <carmen.livi@external.ifom.eu>

Description Quality control (QC) pipeline for ChIP-seq data using a comprehensive set of QC metrics, including previously proposed metrics as well as novel ones, based on local characteristics of the enrichment profile. The package provides functions to calculate a set of QC metrics, a compendium with reference values and machine learning models to score sample quality.

License GPL-2

Encoding UTF-8

LazyData TRUE

Imports ChIC.data (>= 1.11.1), caTools, methods, GenomicRanges, IRanges, parallel, progress, randomForest, caret, grDevices, stats, utils, graphics, S4Vectors, BiocGenerics, genomeIntervals, Rsamtools

Depends spp, R (>= 3.6)

biocViews ChIPSeq, QualityControl

RoxygenNote 7.1.1

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

git_branch RELEASE_3_14

git_last_commit 14ce4c1

git_last_commit_date 2021-10-26

Date/Publication 2022-04-12

R topics documented:

`chicWrapper` *ChIC analysis in one command*

Description

This function creates a single document (in pdf format) containing all the analysis plots produced by ChIC: the CrossCorrelation profile of the immunoprecipitation, the fingerprint plot and the different metagene profiles and returns the ChIC RF score.

chicWrapper

Usage

```
chicWrapper(
  chipName,
  inputName,
  read_length,
  savePlotPath = getwd(),
  target,
  annotationID,
  mc = 1,
  debug = FALSE
)
```

Arguments

`chipName`	String, filename and path to the ChIP bam file (without ".bam" extension)
`inputName`	String, filename and path to the Input bam file (without ".bam" extension)
`read_length`	Integer, length of the sequencing reads

savePlotPath	path, needs to be set to save the summary plot under "savePlotPath" (default=getwd()). In the current version of the code we MUST save the output summary plots in a PDF file. If the end users really needs to suppress the PDF output, they can use the "/dev/null" as output savePlotPath
target	String, chromatin mark or transcription factor to be analysed. Using the function "listAvailableElements" with the keywords "mark" and "TF" shows a list with the available elements.
annotationID	String, indicating the genome assembly
mc	Integer, the number of CPUs for parallelization (default=1)
debug	Boolean, to enter debugging mode. Intermediate files are saved in working directory

Value

ChIC RF score, returns the rnadom forest prediction score of the predictionScore() function, produces the summary report and saves it under "savePlotPath".

Examples

```
## This command is time intensive to run

## To run this example code the user MUST provide 2 bam files: one for
## ChIP and one for the input. Here we used ChIP-seq data from ENCODE.
## Two example files can be downloaded using the following link:
## https://www.encodeproject.org/files/ENCFF000BFX/
## https://www.encodeproject.org/files/ENCFF000BDQ/
## and save them in the working directory (here given in the temporary
## directory "filepath"

mc=5
## Not run:

filepath=tempdir()
setwd(filepath)

target= "H3K4me3"
system("wget
https://www.encodeproject.org/files/ENCFF000BFX/@download/ENCFF000BFX.bam")
system("wget
https://www.encodeproject.org/files/ENCFF000BDQ/@download/ENCFF000BDQ.bam")

chipName=file.path(filepath,"ENCFF000BFX")
inputName=file.path(filepath,"ENCFF000BDQ")

prediction=chicWrapper(chipName=chipName, inputName=inputName,
target= "H3K4me3", read_length=36, mc=mc , savePlotPath=filepath)
print(prediction)

## End(Not run)
```

createMetageneProfile *Wrapper function to create scaled and non-scaled metageneprofiles*

Description

Metagene plots show the signal enrichment around a region of interest like the TSS or over a predefined set of genes. The tag density of the immunoprecipitation is taken over all RefSeg annotated human genes, averaged and log2 transformed. The same is done for the input. The normalized profile is calculated as the signal enrichment (immunoprecipitation over the input). Two objects are created: a non-scaled profile for the TSS and TES, and a scaled profile for the entire gene, including the gene body. The non-scaled profile is constructed around the TSS/TES, with 2KB up- and downstream regions respectively.

CreateMetageneProfile

Usage

```
createMetageneProfile(
  selectedTagsChip,
  selectedTagsInput,
  smoothed.densityChip = NULL,
  smoothed.densityInput = NULL,
  tag.shift,
  annotationID,
  debug = FALSE,
  mc = 1
)
```

Arguments

selectedTagsChip	Data-structure with selected tag information for ChIP (returned by qualityScores_EM).
selectedTagsInput	Data-structure with selected tag information for Input (returned by qualityScores_EM)
smoothed.densityChip	Optional, output of tagDensity on the ChIP taglist object. It is just used as part of the ChIC_wrapper workflow to skip the call to tagDensity function as it is already performed in the qualityScores_GM function.
smoothed.densityInput	Optional, output of tagDensity on the Input taglist object. It is just used as part of the ChIC_wrapper workflow to skip the call to tagDensity function as it is already performed in the qualityScores_GM function.
tag.shift	Integer containing the value of the tag shif, calculated by getCrossCorrelation-Scores()
annotationID	String indicating the genome assembly
debug	Boolean, to enter debugging mode. Intermediate files are saved in working directory
mc	Integer, the number of CPUs for parallelization (default=1)

Value

list with 3 objects: scaled profile ("geneBody"), non-scaled profile for TSS (TSS) and TES (TES). Each object is made of a list containing the chip and the input profile

Examples

```
## This command is time intensive to run
##
## To run the example code the user must provide two bam files for the ChIP
## and the input and read them with the readBamFile() function. To make it
## easier for the user to run the example code we provide tow bam examples
## (chip and input) in our ChIC.data package that have already been loaded
##with the readBamFile() function.

mc=4
finalTagShift=98
## Not run:

filepath=tempdir()
setwd(filepath)

data("chipSubset", package = "ChIC.data", envir = environment())
chipBam=chipSubset
data("inputSubset", package = "ChIC.data", envir = environment())
inputBam=inputSubset

## calculate binding characteristics

chip_binding.characteristics<-spp::get.binding.characteristics(
    chipBam, srange=c(0,500), bin=5,accept.all.tags=TRUE)
input_binding.characteristics<-spp::get.binding.characteristics(
    inputBam, srange=c(0,500), bin=5,accept.all.tags=TRUE)

##get chromosome information and order chip and input by it
chrl_final=intersect(names(chipBam$tags),names(inputBam$tags))
chipBam$tags=chipBam$tags[chrl_final]
chipBam$quality=chipBam$quality[chrl_final]
inputBam$tags=inputBam$tags[chrl_final]
inputBam$quality=inputBam$quality[chrl_final]

##remove sigular positions with extremely high read counts with
##respect to the neighbourhood
selectedTags=removeLocalTagAnomalies(chipBam, inputBam,
chip_binding.characteristics, input_binding.characteristics)

inputBamSelected=selectedTags$input.dataSelected
chipBamSelected=selectedTags$chip.dataSelected

##smooth input and chip tags
smoothedChip <- tagDensity(chipBamSelected,
    tag.shift = finalTagShift, mc = mc)
smoothedInput <- tagDensity(inputBamSelected,
```

```
    tag.shift = finalTagShift, mc = mc)

##calculate metagene profiles
Meta_Result <- createMetageneProfile(
    smoothed.densityChip = smoothedChip,
    smoothed.densityInput = smoothedInput,
    tag.shift = finalTagShift, mc = mc)

## End(Not run)
```

`downsample_ChIPpeaks` *Function for downsampling peaks in ChIP-data to simulate failed antibody*

Description

Function to downsample bam files: reads ChIP and Input bam files, calles peaks and removes randomly 60percent of the reads from the peaks. Returns a downsampled ChIP dataframe.

downsample_ChIPpeaks

Usage

```
downsample_ChIPpeaks(
  chip.data,
  input.data,
  read_length,
  annotationID = "hg19",
  mc = 1,
  debug = FALSE
)
```

Arguments

`chip.data`	data-structure with tag information reads from bam file (see readBamFile())
`input.data`	data-structure with tag information reads from bam file (see readBamFile())
`read_length`	Integer, length of the reads
`annotationID`	String, indicating the genome assembly (Default="hg19")
`mc`	Integer, the number of CPUs for parallelization (default=1)
`debug`	Boolean, to enter debugging mode. Intermediate files are saved in working directory

Value

chip.dataDownSampeld, data-structure with downsampled tags

Examples

```
## This command is time intensive to run

## To run this example code the user MUST provide 2 bam files: one for ChIP
## and one for the input". Here we used ChIP-seq data from ENCODE. Two
## example files can be downloaded using the following link:
## https://www.encodeproject.org/files/ENCFF000BFX/
## https://www.encodeproject.org/files/ENCFF000BDQ/
## and save them in the working directory (here given in the temporary
## directory "filepath"

mc=4
## Not run:

filepath=tempdir()
setwd(filepath)

data("chipSubset", package = "ChIC.data", envir = environment())
chipBam=chipSubset
data("inputSubset", package = "ChIC.data", envir = environment())
inputBam=inputSubset
chip.dataNew=downsample_ChIPpeaks(chip.data=chipBam,
   input.data=inputBa, read_length=read_length,
   annotationID="hg19", mc=mc, debug=FALSE)

message(" downsampling from", sum(unlist(lapply(chipBam$tags,length))),"to ",
sum(unlist(lapply(chip.dataNew$tags,length))))

## End(Not run)
```

`getCrossCorrelationScores` *QC-metrics from cross-correlation profile, phantom peak and general QC-metrics*

Description

We use cross-correlation analysis to obtain QC-metrics proposed for narrow-binding patterns. After calculating the strand cross-correlation coefficient (Kharchenko et al., 2008), we take the following values from the profile: coordinates of the ChIP-peak (fragment length, height A), coordinates at the phantom-peak (read length, height B) and the baseline (C), the strand-shift, the number of uniquely mapped reads (unique_tags), uniquely mapped reads corrected by the library size, the number of reads and the read lengths. We calculate different values using the relative and absolute height of the cross-correlation peaks: the relative and normalized strand coefficient RSC and NSC (Landt et al., 2012), and the quality control tag (Marinov et al., 2013). Other values regarding the library complexity (Landt et al., 2012) like the fraction of non-redundant mapped reads (NRF; ratio between the number of uniquely mapped reads divided by the total number of reads), the NRF adjusted by library size and ignoring the strand direction (NRF_nostrand), and the PCR bottleneck

coefficient PBC (number of genomic locations to which exactly one unique mapping read maps, divided by the number of unique mapping reads).

getCrossCorrelationScores

Usage

```
getCrossCorrelationScores(
  data,
  bchar,
  annotationID = "hg19",
  read_length,
  savePlotPath = NULL,
  mc = 1,
  tag = "ChIP"
)
```

Arguments

data	data-structure with tag information read from bam file (see readBamFile())
bchar	binding.characteristics is a data-structure containing binding information for binding preak separation distance and cross-correlation profile (see spp::get.binding.characteristics).
annotationID	String, indicating the genome assembly (Default="hg19")
read_length	Integer, read length of "data" (Defaul="36")
savePlotPath	if set the plot will be saved under "savePlotPath". Default=NULL and plot will be omitted.
mc	Integer, the number of CPUs for parallelization (default=1)
tag	String,can be used to personalize the prefix of the filename for the cross- correltion plot (default="ChIP" and "Input" in case of cross-correlation plot for the input')

Value

finalList List with QC-metrics

Examples

```
## This command is time intensive to run

## To run the example code the user must provide a bam file and read it
## with the readBamFile() function. To make it easier for the user to run
## the example code we provide a bam file in our ChIC.data package that has
## already been loaded with the readBamFile() function.

mc=4
print("Cross-correlation for ChIP")
## Not run:
filepath=tempdir()
setwd(filepath)
```

```
data("chipSubset", package = "ChIC.data", envir = environment())
chipBam=chipSubset

## calculate binding characteristics

chip_binding.characteristics<-spp::get.binding.characteristics( chipBam,
    srange=c(0,500), bin = 5, accept.all.tags = TRUE)

crossvalues_Chip<-getCrossCorrelationScores( chipBam ,
    chip_binding.characteristics, read_length = 36,
    annotationID="hg19",
    savePlotPath = filepath, mc = mc)

## End(Not run)
```

`getPeakCallingScores` *Calculating QC-values from peak calling procedure*

Description

QC-metrics based on the peak calling are the fraction of usable reads in the peak regions (FRiP) (Landt et al., 2012), for which the function calls sharp- and broad-binding peaks to obtain two types: the FRiP_sharpsPeak and the FRiP_broadPeak. The function takes the number of called of peaks using an FDR of 0.01 and an evalue of 10 (Kharchenko et al., 2008). And count the number of peaks called when using the sharp- and broad-binding option.

getPeakCallingScores

Usage

```
getPeakCallingScores(
  chip,
  input,
  chip.dataSelected,
  input.dataSelected,
  annotationID = "hg19",
  tag.shift = 75,
  mc = 1,
  chrorder = NULL,
  debug = FALSE
)
```

Arguments

`chip`	data-structure with tag information for the ChIP (see readBamFile())
`input`	data-structure with tag information for the Input (see readBamFile())
`chip.dataSelected`	selected ChIP tags after running removeLocalTagAnomalies() which removes local tag anomalies

input.dataSelected
: selected Input tags after running removeLocalTagAnomalies() which removes local tag anomalies

annotationID
: String indicating the genome assembly (Default="hg19")

tag.shift
: Integer containing the value of the tag shift, calculated by getCrossCorrelationScores(). Default=75

mc
: Integer, the number of CPUs for parallelization (default=1)

chrorder
: chromosome order (default=NULL)

debug
: Boolean, to enter debugging mode. Intermediate files are saved in working directory

Value

QCscoreList List with 6 QC-values

Examples

```
mc=4
finalTagShift=98
print("Cross-correlation for ChIP")

## Not run:
filepath=tempdir()
setwd(filepath)

data("chipSubset", package = "ChIC.data", envir = environment())
chipBam=chipSubset
data("inputSubset", package = "ChIC.data", envir = environment())
inputBam=inputSubset

## calculate binding characteristics

chip_binding.characteristics<-spp::get.binding.characteristics(
    chipBam, srange=c(0,500), bin = 5, accept.all.tags = TRUE)

input_binding.characteristics<-spp::get.binding.characteristics(
    inputBam, srange=c(0,500), bin = 5, accept.all.tags = TRUE)

##get chromosome information and order chip and input by it
chrl_final <- intersect(names(chipBam$tags), names(inputBam$tags))
chipBam$tags <- chipBam$tags[chrl_final]
chipBam$quality <- chipBam$quality[chrl_final]
inputBam$tags <- inputBam$tags[chrl_final]
inputBam$quality <- inputBam$quality[chrl_final]

##remove sigular positions with extremely high read counts with
##respect to the neighbourhood
selectedTags <- removeLocalTagAnomalies(chipBam, inputBam,
    chip_binding.characteristics, input_binding.characteristics)
```

```
inputBamSelected <- selectedTags$input.dataSelected
chipBamSelected <- selectedTags$chip.dataSelected

##Finally run function
bindingScores <- getPeakCallingScores(chip = chipBam,
    input = inputBam, chip.dataSelected = chipBamSelected,
    input.dataSelected = inputBamSelected,
    annotationID="hg19",
    tag.shift = finalTagShift, mc = mc)

## End(Not run)
```

`listAvailableElements` *Shows available chromatin marks and factors*

Description

Lists chromatin marks and transcription factors that are available to be used for the comparison analysis by the fuctions metagenePlotsForComparison(), plotReferenceDistribution() and predictionScore().

listAvailableElements

Usage

```
listAvailableElements(target)
```

Arguments

`target`	String, chromatin mark or transcription factor to be analysed. With the keywords "mark" and "TF" the respective lists with the available elements are listed.

Value

List of elements or single string

Examples

```
listAvailableElements(target="CTCF")
listAvailableElements(target="H3K36me3")
listAvailableElements(target="TF")
listAvailableElements(target="mark")
```

listDatasets *Lists the IDs of samples included in the compendium*

Description

Shows the IDs of all analysed ChIP-seq samples included in the compendium from ENCODE and Roadmap.

listDatasets

Usage

```
listDatasets(dataset)
```

Arguments

dataset	String, to specify the dataset for which the IDs have to be returned. Valid keywords are "ENCODE" and "Roadmap".

Value

"ENCODE" returns a vecor of transcription factor, chromatin mark and RNAPol2 sample IDs from ENCODE, "Roadmap" returns a vector of chromatin mark IDs from Roadmap that have been included in the compendium.

Examples

```
listDatasets(dataset="ENCODE")
listDatasets(dataset="Roadmap")
```

listMetrics *Lists the metrics available in the compendium*

Description

Lists the metrics available in the compendium that can be used with plotReferenceDistribution()

listMetrics

Usage

```
listMetrics(category = "all")
```

Arguments

category	String, to specify the category for which the list of metrics should to be returned. Valid keywords are "all", "EM", "GM", "LM".

Value

returns a character vecor of metrics available in the compendium that can be used with plotReferenceDistribution() as "metricToBePlotted" parameter

Examples

```
listMetrics(category="all")
listMetrics(category="EM")
```

`metagenePlotsForComparison` *Function to create metage plots for comparison*

Description

QC-metrics of newly analysed ChIP-seq samples can be compared with the reference values of the compendium and enrichment profiles can be plotted against pre-computed profiles of published datasets. The metagene profiles show the problematic samples signal (red line) for the ChIP, for the input and their relative enrichment when compared to the compendium's mean signal (black line) and its 2 x standard error (blue shadow). Additionally the function plots the desired QC-metric as a red dashed line for the sample plotted against the reference distribution (density plots) of the compendium values stratified by chromatin marks.

metagenePlotsForComparison

Usage

```
metagenePlotsForComparison(
  data,
  target,
  tag,
  savePlotPath = NULL,
  plot = "all"
)
```

Arguments

`data`	metagene-object of metagene profile by createMetageneProfile() containing input and chip profile
`target`	String, chromatin mark or transcription factor to be analysed. Use listAvailableElements() function to check availability.
`tag`	indicating the kind of profile to plot. Can be either: geneBody, TES or TSS.
`savePlotPath`	if set the plot will be saved under 'savePlotPath'. Default=NULL and plot will be forwarded to stdout.
`plot`	character, possible values "norm", "chip", "input", "all" (default). To plot metaprofiles for normalized ChIP/input enrichment, or only ChIP reads density, or only input control reads density or all three plots (respectively)

Value

Creates a pdf figure under 'savePlotPath'

Examples

```
## This command is time intensive to run
##
## To run the example code the user must provide two bam files for the ChIP
## and the input and read them with the readBamFile() function. To make it
## easier for the user to run the example code we provide tow bam examples
## (chip and input) in our ChIC.data package that have already been loaded
## with the readBamFile() function.

mc=4
finalTagShift=98

## Not run:

filepath=tempdir()
setwd(filepath)

data("chipSubset", package = "ChIC.data", envir = environment())
chipBam=chipSubset
data("inputSubset", package = "ChIC.data", envir = environment())
inputBam=inputSubset

## calculate binding characteristics

chip_binding.characteristics<-spp::get.binding.characteristics(
    chipBam, srange=c(0,500), bin=5,accept.all.tags=TRUE)
input_binding.characteristics<-spp::get.binding.characteristics(
    inputBam, srange=c(0,500), bin=5,accept.all.tags=TRUE)

##get chromosome information and order chip and input by it
chrl_final=intersect(names(chipBam$tags),names(inputBam$tags))
chipBam$tags=chipBam$tags[chrl_final]
chipBam$quality=chipBam$quality[chrl_final]
inputBam$tags=inputBam$tags[chrl_final]
inputBam$quality=inputBam$quality[chrl_final]

##remove sigular positions with extremely high read counts with
##respect to the neighbourhood
selectedTags=removeLocalTagAnomalies(chipBam, inputBam,
chip_binding.characteristics, input_binding.characteristics)

inputBamSelected=selectedTags$input.dataSelected
chipBamSelected=selectedTags$chip.dataSelected

##smooth input and chip tags
smoothedChip <- tagDensity(chipBamSelected,
    tag.shift = finalTagShift, mc = mc)
```

```
smoothedInput <- tagDensity(inputBamSelected,
    tag.shift = finalTagShift, mc = mc)

##calculate metagene profiles
Meta_Result <- createMetageneProfile(
    smoothed.densityChip = smoothedChip,
    smoothed.densityInput = smoothedInput,
    tag.shift = finalTagShift, mc = mc)

##compare metagene features of the geneBody with the compendium
metagenePlotsForComparison(data = Meta_Result$geneBody,
    target = "H3K4me3", tag = "geneBody")

## End(Not run)
```

`plotReferenceDistribution` *Function to create reference distribution plot for comparison*

Description

Creates a density plot (in pdf) for the sample against the reference distribution (density plots) of the compendium values stratified by chromatin marks.

plotReferenceDistribution

Usage

```
plotReferenceDistribution(
  target,
  metricToBePlotted = "RSC",
  currentValue,
  savePlotPath = NULL
)
```

Arguments

`target`	String, chromatin mark or transcription factor to be analysed. Use listAvailableElements() function to check availability.
`metricToBePlotted`	The metric to be plotted (Default='RSC')
`currentValue`	The value of the current sample
`savePlotPath`	if set the plot will be saved under 'savePlotPath'. Default=NULL and plot will be forwarded to stdout.

Value

nothing, creates a figure under 'savePlotPath'

Examples

```
print ('Plot distribution of RSC')
## Not run:
filepath=tempdir()
setwd(filepath)

plotReferenceDistribution(target="H3K4me1",
    metricToBePlotted="RSC", currentValue=0.49, savePlotPath=filepath)

## End(Not run)
```

---

predictionScore *Predict score*

Description

predictionScore

Usage

```
predictionScore(
  target,
  features_cc,
  features_global,
  features_TSS,
  features_TES,
  features_scaled
)
```

Arguments

target	String, chromatin mark or transcription factor to be analysed. Use listAvailableElements() function to check availability. If the specific transcription factor is not available the keyword "TF" can be used to call the TF-model.
features_cc	list, with QC-metrics returned from qualityScores_EM()
features_global	list, list with QC-metrics returned from qualityScores_GM()
features_TSS	list, list with QC-metrics returned from qualityScores_LM() with option TSS
features_TES	list, list with QC-metrics returned from qualityScores_LM() with option TES
features_scaled	list, list with QC-metrics returned from qualityScores_LMgenebody()

Value

predictions for positive and negative class

Examples

```
## To execute this command the user has to run the entire pipeline
## (time intensive to run)

## To run this example code the user MUST provide 2 bam files: one for ChIP
## and one for the input". Here we used ChIP-seq data from ENCODE. Two
## example files can be downloaded using the following link:
## https://www.encodeproject.org/files/ENCFF000BFX/
## https://www.encodeproject.org/files/ENCFF000BDQ/
## and save them in the working directory (here given in the temporary
## directory "filepath"

mc=4

## Not run:
filepath=tempdir()
setwd(filepath)

system("wget
https://www.encodeproject.org/files/ENCFF000BFX/@@download/ENCFF000BFX.bam")

system("wget
https://www.encodeproject.org/files/ENCFF000BDQ/@@download/ENCFF000BDQ.bam")

chipName=file.path(filepath,"ENCFF000BFX")
inputName=file.path(filepath,"ENCFF000BDQ")

CC_Result=qualityScores_EM(chipName=chipName, inputName=inputName,
read_length=36, mc=mc,savePlotPath=filepath)

##save tag.shift value
finalTagShift=CC_Result$QCscores_ChIP$tag.shift

##save the smooted profile
smoothedDensityInput=CC_Result$TagDensityInput
smoothedDensityChip=CC_Result$TagDensityChip

##caluclate GM QC-metrics
Ch_Results=qualityScores_GM(densityChip=smoothedDensityChip,
densityInput=smoothedDensityInput,savePlotPath=filepath)

##caluclate metagene profiles
Meta_Result=createMetageneProfile(smoothedDensityChip,smoothedDensityInput,
finalTagShift,annotationID="hg19",mc=mc)

##get LM QC-values
TSSProfile=qualityScores_LM(Meta_Result$TSS,tag="TSS",savePlotPath=filepath)
TESProfile=qualityScores_LM(Meta_Result$TES,tag="TES",savePlotPath=filepath)
geneBody_Plot=qualityScores_LMgenebody(Meta_Result$geneBody,
savePlotPath=filepath)

##Finally use all calculated QC-metrics to predict the final score
```

```
##example for chromatin mark H3K4me3
predictionScore(target="H3K4me3", features_cc=CC_Result,
features_global=Ch_Results,features_TSS=TSSProfile, features_TES=TESProfile,
features_scaled=geneBody_Plot)

##example for TF not available in compendium
predictionScore(target="TF", features_cc=CC_Result,
features_global=Ch_Results,features_TSS=TSSProfile, features_TES=TESProfile,
features_scaled=geneBody_Plot)

##example for CTCF
predictionScore(target="CTCF", features_cc=CC_Result,
features_global=Ch_Results,features_TSS=TSSProfile, features_TES=TESProfile,
features_scaled=geneBody_Plot)

## End(Not run)
```

`qualityScores_EM` *Wrapper function to calculate EM metrics*

Description

Wrapper that reads bam files and provides EM QC-metrics from cross-correlation analysis, peak calling and general metrics like for example the read-length or NRF. In total 22 features are calculated.

qualityScores_EM

Usage

```
qualityScores_EM(
  chipName,
  inputName,
  read_length,
  chip.data = NULL,
  input.data = NULL,
  readAlignerType = "bam",
  annotationID,
  mc = 1,
  crossCorrelation_Input = FALSE,
  downSamplingChIP = FALSE,
  writeWig = FALSE,
  savePlotPath = NULL,
  debug = FALSE
)
```

Arguments

chipName	Character, filename (and optional path) for the ChIP bam file (without the .bam extension)
inputName	Character, filename (and optional path) for the Input control bam file (without the .bam extension)
read_length	Integer, length of the reads
chip.data	Optional, taglist object for ChIP reads as returned by spp or readBamFile() function. If not set (NULL) the data will be read from the BAM file with name specified by "chipName"
input.data	Optional, taglist object for Input control reads as returned by spp or readBamFile() function. If not set (NULL) the data will be read from the BAM file with name specified by "inputName"
readAlignerType	string, bam (default) tagAlign file format are supported
annotationID	Character, indicating the genome assembly
mc	Integer, the number of CPUs for parallelization (default=1)
crossCorrelation_Input	Boolean, calculates cross-correlation and and EM metrics for the input. The default=FALSE as the running time increases and the metrics are not used in quality prediction.
downSamplingChIP	Boolean, to be used to downsample reads within enrichment peaks. This option was used for generating simulated low quality (low enrichment) profiles for testing the prediction models. The default is FALSE and should generally not be used by end users.
writeWig	Boolean, saves smoothed tag density in wig format in working directory for Input and ChIP
savePlotPath,	set if Cross-correlation plot should be saved under "savePlotPath". Default=NULL and plot will be forwarded to stdout
debug	Boolean, to enter debugging mode. Intermediate files are saved in working directory

Value

returnList, contains QCscores_ChIP List of QC-metrics with crosscorrelation values for the ChIP QCscores_Input List of QC-metrics with crosscorrelation values for the Input if "crossCorrelation_Input" parameter was set to TRUE, NULL otherwise QCscores_binding List of QCscores from peak calls TagDensityChip Tag-density profile, smoothed by the Gaussian kernel (for further details see "spp" package) TagDensityInput Tag density-profile, smoothed by the Gaussian kernel (for further details see "spp" package)

Examples

```
## This command is time intensive to run
```

```
## To run this example code the user MUST provide 2 bam files: one for ChIP
## and one for the input". Here we used ChIP-seq data from ENCODE. Two
## example files can be downloaded using the following link:
## https://www.encodeproject.org/files/ENCFF000BFX/
## https://www.encodeproject.org/files/ENCFF000BDQ/
## and save them in the working directory (here given in the temporary
## directory "filepath"

mc=4
## Not run:

filepath=tempdir()
setwd(filepath)

system("wget
https://www.encodeproject.org/files/ENCFF000BFX/@download/ENCFF000BFX.bam")
system("wget
https://www.encodeproject.org/files/ENCFF000BDQ/@download/ENCFF000BDQ.bam")

chipName=file.path(filepath,"ENCFF000BFX")
inputName=file.path(filepath,"ENCFF000BDQ")

CC_Result=qualityScores_EM(chipName=chipName, inputName=inputName,
read_length=36, mc=mc, annotationID = "hg19")

## End(Not run)
```

`qualityScores_GM` *Wrapper function to calculate GM metrics from global read distribution*

Description

This set of values is based on the global read distribution along the genome for immunoprecipitation and input data (Diaz et al., 2012). The genome is binned and the read coverage counted for each bin. Then the function computes the cumulative distribution of reads density per genomic bin and plots the fraction of the coverage on the y-axis and the fraction of bins on the x-axis. Then different values can be sampled from the cumulative distribution: like the fraction of bins without reads for in immunoprecipitation and input,the point of the maximum distance between the ChIP and the input (x-axis, y-axis for immunoprecipitation and input, distance (as absolute difference), the sign of the differences), the fraction of reads in the top 1 percent bin for immunoprecipitation and input. Finally, the funciton returns 9 QC-measures.

qualityScores_GM

Usage

```
qualityScores_GM(
  selectedTagsChip,
  selectedTagsInput,
```

```
  tag.shift,
  annotationID,
  savePlotPath = NULL,
  mc = 1,
  returnDensities = FALSE
)
```

Arguments

selectedTagsChip	Data-structure with selected tag information for ChIP (returned by qualityScores_EM).
selectedTagsInput	Data-structure with selected tag information for Input (returned by qualityScores_EM)
tag.shift,	Integer containing the value of the tag shif, calculated by getCrossCorrelation-Scores()
annotationID	String, indicating the genome assembly
savePlotPath	if set the plot will be saved under "savePlotPath". Default=NULL and plot will be forwarded to stdout.
mc	Integer, the number of CPUs for parallelization (default=1)
returnDensities	Boolean, default FALSE. Whether smoothed Chip and Input reads densities should be returned. This is used only for optimizing the flow od fata in the ChIC_wrapper function

Value

finalList List with 9 QC-values

Examples

```
## This command is time intensive to run
##
## To run the example code the user must provide two bam files for the ChIP
## and the input and read them with the readBamFile() function. To make it
## easier for the user to run the example code we provide tow bam examples
## (chip and input) in our ChIC.data package that have already been loaded
## with the readBamFile() function.

mc=4
finalTagShift=98
## Not run:

filepath=tempdir()
setwd(filepath)

data("chipSubset", package = "ChIC.data", envir = environment())
chipBam=chipSubset
data("inputSubset", package = "ChIC.data", envir = environment())
inputBam=inputSubset
```

```
## calculate binding characteristics

chip_binding.characteristics<-spp::get.binding.characteristics(
    chipBam, srange=c(0,500), bin=5,accept.all.tags=TRUE)
input_binding.characteristics<-spp::get.binding.characteristics(
    inputBam, srange=c(0,500), bin=5,accept.all.tags=TRUE)

##get chromosome information and order chip and input by it
chrl_final=intersect(names(chipBam$tags),names(inputBam$tags))
chipBam$tags=chipBam$tags[chrl_final]
chipBam$quality=chipBam$quality[chrl_final]
inputBam$tags=inputBam$tags[chrl_final]
inputBam$quality=inputBam$quality[chrl_final]

##remove sigular positions with extremely high read counts with
##respect to the neighbourhood
selectedTags=removeLocalTagAnomalies(chipBam, inputBam,
chip_binding.characteristics, input_binding.characteristics)

inputBamSelected=selectedTags$input.dataSelected
chipBamSelected=selectedTags$chip.dataSelected

##smooth input and chip tags
smoothedChip <- tagDensity(chipBamSelected,
    tag.shift = finalTagShift, mc = mc)
smoothedInput <- tagDensity(inputBamSelected,
    tag.shift = finalTagShift, mc = mc)

Ch_Results <- qualityScores_GM(densityChip = smoothedChip,
    densityInput = smoothedInput, savePlotPath = filepath)

## End(Not run)
```

`qualityScores_LM` *Wrapper function that plots non-scaled profiles for TSS of TES and to collect the QC-metrics*

Description

The non-scaled profile is constructed around the TSS/TES, with 2KB up- and downstream regions respectively. Different values are taken at the TSS/TES and surroundings with +/-2KB, +/-1KB and +/-500 sizes. For all the genomic positions, we kept the values for the ChIP and the normalized profile, as the normalization already contains information from the input. Additionally, we calculated for all of the intervals between the predefined positions the area under the profile, the local maxima (x, y coordinates), the variance, the standard deviation and the quantiles at 0 the function returns 43 QC-metrics.

qualityScores_LM

Usage

```
qualityScores_LM(data, tag, savePlotPath = NULL, plot = "all")
```

Arguments

data	metagene-list for input and chip samples as returned by createMetageneProfile()
tag	String that can be 'TSS' or 'TES',indicating if the TSS or the TES profile should be calcualted (Default='TSS')
savePlotPath	if set the plot will be saved under 'savePlotPath'. Default=NULL and plot will be forwarded to stdout.
plot	character, possible values "norm", "split", "all" (default) or "none" to plot metaprofiles for normalized ChIP/input enrichment, or the two samples reads densities separated, or both plots, or none (respectively)

Value

result Dataframe with QC-values for chip, input and normalized metagene profile

Examples

```
## This command is time intensive to run
##
## To run the example code the user must provide two bam files for the ChIP
## and the input and read them with the readBamFile() function. To make it
## easier for the user to run the example code we provide tow bam examples
## (chip and input) in our ChIC.data package that have already been loaded
## with the readBamFile() function.

mc=4
finalTagShift=82
## Not run:

filepath=tempdir()
setwd(filepath)

data("chipBam", package = "ChIC.data", envir = environment())
data("inputBam", package = "ChIC.data", envir = environment())

## calculate binding characteristics
chip_binding.characteristics<-spp::get.binding.characteristics(
    chipBam, srange=c(0,500), bin=5,accept.all.tags=TRUE)

input_binding.characteristics<-spp::get.binding.characteristics(
    inputBam, srange=c(0,500), bin=5,accept.all.tags=TRUE)

##get chromosome information and order chip and input by it
chrl_final=intersect(names(chipBam$tags), names(inputBam$tags))
chipBam$tags=chipBam$tags[chrl_final]
chipBam$quality=chipBam$quality[chrl_final]
```

```
inputBam$tags=inputBam$tags[chrl_final]
inputBam$quality=inputBam$quality[chrl_final]

##remove sigular positions with extremely high read counts with
##respect to the neighbourhood
selectedTags=removeLocalTagAnomalies(chipBam, inputBam,
chip_binding.characteristics, input_binding.characteristics)

inputBamSelected=selectedTags$input.dataSelected
chipBamSelected=selectedTags$chip.dataSelected

##smooth input and chip tags
smoothedChip=tagDensity(chipBamSelected, tag.shift=finalTagShift)
smoothedInput=tagDensity(inputBamSelected, tag.shift=finalTagShift)

##calculate metagene profiles
Meta_Result=createMetageneProfile(smoothed.densityChip=smoothedChip,
    smoothedInput,tag.shift=finalTagShift, mc=mc)

##estract QC-values and plot metageneprofile for TSS
TSS_Scores=qualityScores_LM(data=Meta_Result$TSS, tag="TSS",
savePlotPath=filepath))

## End(Not run)
```

---

qualityScores_LMgenebody *Wrapper function to plot the scaled metagene- profile and to collect the QC-metrics*

Description

The scaled metagene profile that includes the gene body, the signal is captured on a real scale from the TSS and an upstream region of 2KB. From the TSS, the gene body is constructed with 0.5KB in real scale at the gene start (TSS + 0.5KB) and the gene end (TES - 0.5KB), whereas the remaining gene body is scaled to a virtual length of 2000. Considering the length of these regions, the minimum gene length required is 3KB and shorter genes are filtered out. From the profile, we take enrichment values at different coordinates: at -2KB, at the TSS, inner margin (0.5KB), gene body (2KB + 2 * inner margin), gene body+1KB. We collect in total 42 QC-metrics from the ChIP and normalized profile.

qualityScores_LMgenebody

Usage

```
qualityScores_LMgenebody(
  data,
  savePlotPath = NULL,
  tag = "geneBody",
  plot = "all"
)
```

Arguments

data	metagene-list for input and chip sample of the genebody profile returned by createMetageneProfile()
savePlotPath	if set the plot will be saved under 'savePlotPath'. Default=NULL and plot will be forwarded to stdout.
tag,	character, it should always be "geneBody" in the current impplementation. This is just for simmetry with qualityScores_LM function
plot	character, possible values "norm", "split", "all" (default) or "none" to plot metaprofiles for normalized ChIP/input enrichment, or the two samples reads densities separated, or both plots, or none (respectively)

Value

returnList

Examples

```
## This command is time intensive to run
##
## To run the example code the user must provide two bam files for the ChIP
## and the input and read them with the readBamFile() function. To make it
## easier for the user to run the example code we provide tow bam examples
## (chip and input) in our ChIC.data package that have already been loaded
## with the readBamFile() function.

mc=4
finalTagShift=82
## Not run:

filepath=tempdir()
setwd(filepath)

data("chipBam", package = "ChIC.data", envir = environment())
data("inputBam", package = "ChIC.data", envir = environment())

## calculate binding characteristics
chip_binding.characteristics<-spp::get.binding.characteristics(
    chipBam, srange=c(0,500), bin=5,accept.all.tags=TRUE)

input_binding.characteristics<-spp::get.binding.characteristics(
    inputBam, srange=c(0,500), bin=5,accept.all.tags=TRUE)

##get chromosome information and order chip and input by it
chrl_final=intersect(names(chipBam$tags), names(inputBam$tags))
chipBam$tags=chipBam$tags[chrl_final]
chipBam$quality=chipBam$quality[chrl_final]
inputBam$tags=inputBam$tags[chrl_final]
inputBam$quality=inputBam$quality[chrl_final]

##remove sigular positions with extremely high read counts with
```

```
##respect to the neighbourhood
selectedTags=removeLocalTagAnomalies(chipBam, inputBam,
chip_binding.characteristics, input_binding.characteristics)

inputBamSelected=selectedTags$input.dataSelected
chipBamSelected=selectedTags$chip.dataSelected

##smooth input and chip tags
smoothedChip=tagDensity(chipBamSelected, tag.shift=finalTagShift)
smoothedInput=tagDensity(inputBamSelected, tag.shift=finalTagShift)

##calculate metagene profiles
Meta_Result=createMetageneProfile(smoothed.densityChip=smoothedChip,
    smoothedInput,tag.shift=finalTagShift, mc=mc)

#create scaled metagene profile
geneBody_Scores=qualityScores_LMgenebody(Meta_Result$geneBody,
savePlotPath=filepath)

## End(Not run)
```

`readBamFile` *Read bam file*

Description

Reading bam file format

readBamFile

Usage

```
readBamFile(filename, readAlignerType = "bam")
```

Arguments

`filename,`	name/path of the bam file to be read (without extension)
`readAlignerType,`	format of the input file. Currently only 'bam' (default) and 'tagAlign' are supported

Value

result list of lists, every list corresponds to a chromosome and contains a vector of coordinates of the 5' ends of the aligned tags.

Examples

```
## To run this example code the user MUST provide a bam file: The user can
## download a ChIP-seq bam file for example from ENCODE:
## https://www.encodeproject.org/files/ENCFF000BFX/
## and save it in the working directory

bamID="ENCFF000BFX"
## Not run:
filepath=tempdir()
setwd(filepath)
system("wget
https://www.encodeproject.org/files/ENCFF000BFX/@download/ENCFF000BFX.bam")

bamName=file.path(filepath,bamID)
chipBam=readBamFile(bamName)

## End(Not run)
```

`removeLocalTagAnomalies` *Removes loval anomalies*

Description

The removeLocalTagAnomalies function removes tags from regions with extremely high tag counts compared to the neighbourhood.

removeLocalTagAnomalies

Usage

```
removeLocalTagAnomalies(chip, input, chip_b.characteristics)
```

Arguments

`chip,`	data-structure with tag information for the ChIP (see readBamFile())
`input,`	data-structure with tag information for the Input (see readBamFile())
`chip_b.characteristics`	binding.characteristics of the ChIP. Is a data-structure containing binding information for binding peak separation distance and cross-correlation profile (see get.binding.characteristics)

Value

result A list containing filtered data structure for ChIP and Input

Examples

```
## This command is time intensive to run
##
## To run the example code the user must provide two bam files for the ChIP
## and the input and read them with the readBamFile() function. To make it
## easier for the user to run the example code we provide tow bam examples
## (chip and input) in our ChIC.data package that have already been loaded
##with the readBamFile() function.

mc=4
## Not run:

filepath=tempdir()
setwd(filepath)

##load the data
data("chipSubset", package = "ChIC.data", envir = environment())
chipBam=chipSubset
data("inputSubset", package = "ChIC.data", envir = environment())
inputBam=inputSubset

## calculate binding characteristics

chip_binding.characteristics<-spp::get.binding.characteristics(
chipBam, srange=c(0,500), bin=5,accept.all.tags=TRUE)

##get chromosome information and order chip and input by it
chrl_final=intersect(names(chipBam$tags),names(inputBam$tags))
chipBam$tags=chipBam$tags[chrl_final]
chipBam$quality=chipBam$quality[chrl_final]
inputBam$tags=inputBam$tags[chrl_final]
inputBam$quality=inputBam$quality[chrl_final]

##remove singular positions with extremely high read counts with
##respect to the neighbourhood
selectedTags=removeLocalTagAnomalies(chipBam, inputBam,
chip_binding.characteristics)

## End(Not run)
```

`tagDensity` *Smoothed tag density*

Description

Calculates the smoothed tag density using spp::get.smoothed.tag.density

tagDensity

Usage

```
tagDensity(data, tag.shift, annotationID = "hg19", mc = 1)
```

Arguments

data,	data-structure with tag information (see readBamFile())
tag.shift,	Integer containing the value of the tag shif, calculated by getCrossCorrelation-Scores()
annotationID	String, indicating the genome assembly (Default="hg19")
mc	Integer, the number of CPUs for parallelization (default=1)

Value

smoothed.density A list of lists, each list corresponds to a chromosome and contains a vector of coordinates of the 5' ends of the aligned tags

Examples

```
## This command is time intensive to run
##
## To run the example code the user must provide two bam files for the ChIP
## and the input and read them with the readBamFile() function. To make it
## easier for the user to run the example code we provide tow bam examples
## (chip and input) in our ChIC.data package that have already been loaded
## with the readBamFile() function.

mc=4
finalTagShift=98
## Not run:

filepath=tempdir()
setwd(filepath)

data("chipSubset", package = "ChIC.data", envir = environment())
chipBam=chipSubset
data("inputSubset", package = "ChIC.data", envir = environment())
inputBam=inputSubset

chip_binding.characteristics<-spp::get.binding.characteristics(
    chipBam, srange=c(0,500), bin=5,accept.all.tags=TRUE)
input_binding.characteristics<-spp::get.binding.characteristics(
    inputBam, srange=c(0,500), bin=5,accept.all.tags=TRUE)

##get chromosome information and order chip and input by it
chrl_final=intersect(names(chipBam$tags),names(inputBam$tags))
chipBam$tags=chipBam$tags[chrl_final]
chipBam$quality=chipBam$quality[chrl_final]
inputBam$tags=inputBam$tags[chrl_final]
inputBam$quality=inputBam$quality[chrl_final]
```

```
##remove sigular positions with extremely high read counts with
##respect to the neighbourhood
selectedTags=removeLocalTagAnomalies(chipBam, inputBam,
chip_binding.characteristics, input_binding.characteristics)

chipBamSelected=selectedTags$chip.dataSelected

smoothedChip=tagDensity(chipBamSelected, tag.shift=finalTagShift)

## End(Not run)
```

Index