Package 'crupR'

June 29, 2025

Type Package

Title An R package to predict condition-specific enhancers from ChIP-seq data

Version 1.0.0

Description
An R package that offers a workflow to predict condition-specific enhancers from ChIP-seq data. The prediction of regulatory units is done in four main steps:
Step 1 - the normalization of the ChIP-seq counts.
Step 2 - the prediction of active enhancers binwise on the whole genome.
Step 3 - the condition-specific clustering of the putative active enhancers.
Step 4 - the detection of possible target genes of the condition-specific clusters using RNA-seq counts.

License GPL-3

Encoding UTF-8

LazyData false

Imports bamsignals, Rsamtools, GenomicRanges, preprocessCore, randomForest, rtracklayer, GenomeInfoDb, S4Vectors, ggplot2, matrixStats, dplyr, IRanges, GenomicAlignments, GenomicFeatures, TxDb.Mmusculus.UCSC.mm10.knownGene, TxDb.Mmusculus.UCSC.mm9.knownGene, TxDb.Hsapiens.UCSC.hg19.knownGene, TxDb.Hsapiens.UCSC.hg38.knownGene, reshape2, magrittr, stats, utils, grDevices, SummarizedExperiment, BiocParallel, fs, methods

Depends R (>= 4.4.0)

Suggests testthat, BiocStyle, knitr, rmarkdown

biocViews DifferentialPeakCalling, GeneTarget, FunctionalPrediction, HistoneModification, PeakDetection

BiocType Software

URL https://github.com/akbariomgba/crupR

BugReports https://github.com/akbariomgba/crupR/issues

RoxygenNote 7.3.2

VignetteBuilder knitr

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

git_branch RELEASE_3_21

git_last_commit 64dc7c4

git_last_commit_date 2025-04-15

Repository Bioconductor 3.21

Date/Publication 2025-06-29

Author Persia Akbari Omgba [cre],
Verena Laupert [aut],
Martin Vingron [aut]

Maintainer Persia Akbari Omgba <omgba@molgen.mpg.de>

Contents

`crupR-package` *crupR: Condition-specific Regulatory Units Prediction*

Description

The crupR package is the re-engineered R package version of the enhancer prediction pipeline CRUP (Ramisch et. al, 2019). It is designed to provide a simple pipeline for the analysis and prediction of enhancers using ChIP-seq experiments of different histone modifications (H3K4me1, H3K4me3, H2K27ac) as input.

Details

The main functions are:

- `normalize` - normalize the ChIP-seq data of the three histone modifications by using the input experiment
- `getEnhancers` - apply the enhancer classifier to the genome-wide, normalized HM counts

- `getDynamics` - use the genome-wide enhancer predictions to identify condition-specific enhancers
- `getTargets` - correlate gene expression counts and enhancer activity over different conditions to find target genes of the enhancers.
- `plotSummary` - Plot the activity distributions of the condition-specific enhancers identified with `getDynamics`
- `getSE` - summarize the enhancer predictions into enhancer peaks and detect super-enhancers
- `saveFiles` - export the outputs of the main functions

For detailed information on usage, see the package vignette, by typing `vignette('crupR')`.

The code can be viewed at the GitHub repository:

https://github.com/akbariomgba/crupR

Author(s)

Persia Akbari Omgba, Verena Laupert, Martin Vingron

References

Ramisch, A., Heinrich, V., Glaser, L.V. et al. CRUP: a comprehensive framework to predict condition-specific regulatory units. Genome Biol 20, 227 (2019). https://doi.org/10.1186/s13059-019-1860-7

See Also

Useful links:

- https://github.com/akbariomgba/crupR
- Report bugs at https://github.com/akbariomgba/crupR/issues

`getDynamics` *Clusters the active enhancer into condition-specific groups*

Description

The genome-wide enhancer probabilities from the prior step (getEnhancers) for every sample of the different conditions are compared to find condition-specific (differential) enhancers. For this, replicates of every condition respectively are merged into a group sample. The group samples are compared in a pairwise manner and regions with significant differences are considered differential enhancers. Last, the activity patterns of the differential enhancers are indentified and enhancers with the same patterns are gouped into one cluster.

Usage

```
getDynamics(
  data,
  w_0 = 0.5,
  cutoff = 0.05,
  W = 10,
  BPPARAM = BiocParallel::SerialParam()
)
```

Arguments

data	List containing the bin-wise enhancer prediction values as a GRanges object (output of getEnhancers() or getSE()) for every condition
w_0	The minimum difference between the normalized prediction means that two enhancers need in order to be included in the clustering. Default is 0.5 ([0,1]).
cutoff	cutoff for the p-values calculated during the clustering. Default is 0.05 ([0,1]).
W	Number of bins +/- the current bin that should be included when calculating the p-values. Default is 10 ([5, 50]).
BPPARAM	An object of class SerialParam that is used as input for the BiocParallel functions.

Details

First, for every condition, the genome-wide enhancer probabilities of all replicates are merged into one genome-wide vector by taking their mean and normalizing it by the group variance: mean_weighted = (mean over all replicates)/(1 - group variance + K). K is a pseudo-count of 1, if the group variance is 1. Otherwise, K is 0. After forming the weighted mean for every condition, the summarized genome-wide probabilities are compared in a pairwise manner: A window is formed by extending the current bin by W bins to the left and right (default W = 10) and is then slid over the genome. For every position the distribution of the summarized proabilities in the window are compared using a Kolmogorov Smirnov (KS) test. If the test detects a significant difference between the conditions, the given region is considered a differential enhancer. The pairwise comparisons of every condition results in a list of differential enhancers. These are further divided by their activity patterns. The activtiy pattern of a differential enhancer is inferred by the conditions in which it shows differential activity and the direction of said difference (is it differentially active or inactive). The pattern is encoded in a binary manner, meaning a bit to represent the outcome of one pairwise comparison. For i.e. 3 conditions (cond1, cond2, cond3), there would be three comparisons (cond1-cond2, cond1-cond3, cond2-cond3) resulting in 6 possible outcomes. The first three bits represent the following outcomes: cond1>cond2, cond1>cond3, cond2>cond3. The last three bits represent the opposite outcomes: cond1<cond2, cond1<cond3, cond2<cond3. If any of these outcomes are true, the respective bit is set to 1, otherwise it is 0. For example, if one differential enhancer is active in cond1 and cond3, but inactive in cond2, only 2 outcomes would be true (cond1>cond2, cond2<cond3), resulting in following acitvity pattern: 100001.

The enhancers are divided according to their pattern into different clusters. To reduce the number of comparisons, only windows in which the mean difference between the enhancer probabilities of the conditions surpasses a pre-defined threshold w_0 (default: 0.5) are compared.

Value

GRanges object containing the condition-specific clusters

Examples

```
#recreate the outputs of getEnhancers()
files <- c( system.file('extdata', 'Condition1.H3K4me1.bam', package='crupR'),
            system.file('extdata', 'Condition1.H3K4me3.bam', package='crupR'),
            system.file('extdata', 'Condition1.H3K27ac.bam', package='crupR'),
            system.file('extdata', 'Condition2.H3K4me1.bam', package='crupR'),
            system.file('extdata', 'Condition2.H3K4me3.bam', package='crupR'),
            system.file('extdata', 'Condition2.H3K27ac.bam', package='crupR'))
inputs <- rep(system.file('extdata', 'Condition1.Input.bam',
package='crupR'), 3)
inputs2 <- rep(system.file('extdata', 'Condition2.Input.bam',
package='crupR'), 3)
metaData <- data.frame( HM = rep(c('H3K4me1', 'H3K4me3', 'H3K27ac'),2),
                        condition = c(1,1,1,2,2,2), replicate = c(1,1,1,1,1,1),
                        bamFile = files, inputFile = c(inputs, inputs2))
metaData1 <- subset(metaData, condition == 1)
metaData2 <- subset(metaData, condition == 2)
pred1 <- readRDS(system.file(  'extdata', 'condition1_predictions.rds',
                               package = 'crupR'))
S4Vectors::metadata(pred1) <- metaData1
pred2 <- readRDS(system.file(  'extdata', 'condition2_predictions.rds',
                              package = 'crupR'))
S4Vectors::metadata(pred2) <- metaData2
#put the outputs in a list
predictions <- list(pred1, pred2)
#run the function
getDynamics(data = predictions)
```

`getEnhancers` *Predicts the occurence of enhancers using the normalized HMs counts.*

Description

This function gets the output of the prior normalization step as input and uses the three histone modifications to predict the probability of an active enhancer being present.

Usage

```
getEnhancers(data, classifier = NULL, all = FALSE)
```

Arguments

data	normalized ChIP-seq counts in a GRanges object (output of the normalize() step)
classifier	The path of the classifier to use for the prediction. When set to NULL (default), the default classifier is used. Both classifiers are objects of class 'randomForest' as implemented by the randomForest package. For more information on this object, please check the documentation of randomForest.
all	[LOGICAL] Whether to include the probabilities of the two individual random forests in the output. Default is FALSE.

Details

First, the input-normalized histone modification counts are quantile-normalized. This way their distribution is matching to what the classifiers have been trained on. Next, the two random forest classifiers go through every bin and use the 5 upstream and downstream flanking bins to make predictions. One classifiers predicts whether the current bin is an active region based on the surrounding histone modification patterns. The other predicts whether the current bin is an enhancer or promoter, assuming it's already active. Last, the predicted probabilities of each classifier for every bin are mulitplied and their product forms the final binwise enhancer activity probability.

Value

GRanges object containing the enhancer probabilities for each 100bp bin

Examples

```
#first recreate the output of crupR::normalize (so skip this)
files <- c(system.file('extdata', 'Condition2.H3K4me1.bam', package='crupR'),
          system.file('extdata', 'Condition2.H3K4me3.bam', package='crupR'),
          system.file('extdata', 'Condition2.H3K27ac.bam', package='crupR'))
inputs <- rep(system.file('extdata', 'Condition2.Input.bam', package='crupR'))
metaData <- data.frame(HM = c('H3K4me1','H3K4me3','H3K27ac'),
                   condition = c(2,2,2), replicate = c(1,1,1),
                   bamFile = files, inputFile = inputs)
norm <- readRDS(system.file('extdata', 'condition2_normalized.rds', package='crupR'))
S4Vectors::metadata(norm) <- metaData
#let's run the actual function
getEnhancers(data = norm)
```

---

getSE *Find enhancer peaks and super enhancers*

Description

An optional intermediate step to summarize the genome-wide enhancer predictions from the prior step (getEnhancers). First, enhancer peaks are detected. Next, neighbouring enhancer peaks are further grouped together into super-enhancers, which we define as clusters of proximal enhancer regions.

Usage

```
getSE(
  data,
  cutoff = 0.5,
  distance = 12500,
  BPPARAM = BiocParallel::SerialParam()
)
```

Arguments

data	enhancer prediction values in a GRanges object (output of getEnhancers())
cutoff	Cut-off for enhancer probabilities. Default is 0.5.
distance	Maximum distance (in bp) for clustering. Default is 12500.
BPPARAM	An object of class SerialParam that is used as input for the BiocParallel functions.

Details

First, enhancer peaks are identified. All bins with an enhancer probability >=0.5 are sorted in a descending manner by their probabilities. The bins are then extended by 5 bins up and downstream resulting in regions of size 1100bp. Overlapping regions are discarded, while keeping the region with the higher enhancer probability. The resulting list of enhancer peaks is further summarized into clusters by grouping peaks in close vicinity (default: max. 12.5kb up-/downstream distance) together. These clusters are supposed to reflect super-enhancers.

Value

A list containing the enhancer prediction values as a GRanges object (like the input data), the enhancer peak calls as a GRanges object (can be exported as a bedGraph) and the clusters of peaks (super-enhancers) in a GRanges object (can be exported as BED file)

Examples

```
# first recreate the output of crupR:getPredictions
files <- c(system.file('extdata', 'Condition2.H3K4me1.bam', package='crupR'),
           system.file('extdata', 'Condition2.H3K4me3.bam', package='crupR'),
           system.file('extdata', 'Condition2.H3K27ac.bam', package='crupR'))
inputs <- rep(system.file('extdata', 'Condition2.Input.bam', package='crupR'), 3)
#create the metaData frame
metaData <- data.frame(HM = c('H3K4me1','H3K4me3','H3K27ac'),
                       condition = c(2,2,2), replicate = c(1,1,1),
                       bamFile = files, inputFile = inputs)
prediction <- readRDS(system.file('extdata', 'condition2_predictions.rds', package='crupR'))
S4Vectors::metadata(prediction) <- metaData
#run the function
se <- getSE(data = prediction)
```

getTargets *Find the target genes of the regulatory, condition-specific clusters*

Description

This functions aims to connect the differential enhancers identified in the prior step (getDynamics) to the potential target genes. It exploits normalized gene expression counts for every samples, derived from RNA-seq experiments, and correlates the gene activities with predicted enhancer probabilities over each replicate of all conditions. If there is high correlation between a gene and an enhancer, the respective gene is considered a target gene of the enhancer. The number of comparisons is narrowed down by only comparing enhancers to genes within the same topologically associating domain (TAD). Alternatively, one can also just check the nearest gene for every enhancer.

Usage

```
getTargets(
  data,
  expr = NULL,
  genome,
  TAD.file = NULL,
  cutoff = 0.9,
  nearest = FALSE,
  BPPARAM = BiocParallel::SerialParam()
)
```

Arguments

data	condition-specific clusters in a GRanges object (output of getDynamics())
expr	SummarizedExperiment object containing the gene expression counts of RNA-seq experiments for each condition and its replicates
genome	Genome used in the .bam files of the RNA-seq experiments. Possible options are 'mm9', 'mm10', 'hg19' and 'hg38'.
TAD.file	Path to the TAD file to use for finding the target genes. If set to NULL, the default file is used (only if the 'mm10' genome was used)
cutoff	cut-off for correlation between cluster and gene. Default is 0.9.
nearest	[LOGICAL] If set, the nearest gene is taken to build the regulatory regions.
BPPARAM	An object of class SerialParam that is used as input for the BiocParallel functions.

Details

This functions depends on the presence of gene expression counts for every every sample. These can be created by using e.g. DESeq2. For an example code of how to get the counts, check /inst/script/crupR_example_files.txt.

This functions compares the gene expression counts of the candidate target genes for a differential enhancer and correlates them to the enhancer probabilities as computed in getEnhancers. If the correlation surpasses a threshold (default: 0.9), the candidate gene is considered a target gene. There are two ways to define candidate target genes: Either by using the topologically associating domains (TADs) as boundaries for potential interactions or in case no TAD annotations are available, the nearest gene of the respective differential enhancer is considered.

Value

GRanges object containing the dynamic regulatory units

Examples

```
#first get the output of crupR::getDynamics so skip this
files <- c(system.file('extdata', 'Condition1.H3K4me1.bam', package='crupR'),
           system.file('extdata', 'Condition1.H3K4me3.bam', package='crupR'),
           system.file('extdata', 'Condition1.H3K27ac.bam', package='crupR'),
           system.file('extdata', 'Condition2.H3K4me1.bam', package='crupR'),
           system.file('extdata', 'Condition2.H3K4me3.bam', package='crupR'),
           system.file('extdata', 'Condition2.H3K27ac.bam', package='crupR'))
inputs <- rep(system.file('extdata', 'Condition1.Input.bam', package='crupR'), 3)
inputs2 <- rep(system.file('extdata', 'Condition2.Input.bam', package='crupR'), 3)
metaData <- data.frame(HM = rep(c('H3K4me1', 'H3K4me3', 'H3K27ac'),2),
                       condition = c(1,1,1,2,2,2), replicate = c(1,1,1,1,1,1),
                       bamFile = files, inputFile = c(inputs, inputs2))
clusters <- readRDS(system.file('extdata', 'differential_enhancers.rds', package='crupR'))
S4Vectors::metadata(clusters) <- metaData
#load your SummarizedExperiments object containing the gene expressions counts and run the function
expr <- readRDS(system.file('extdata', 'expressions.rds', package='crupR'))
getTargets(data = clusters, expr = expr, genome = 'mm10')
```

---

normalize *Normalization of ChIP-seq counts by input*

Description

This function normalizes the ChIP-seq counts for the three histone modifications (H3K4me3, H3K4me1, H2K27ac) using an input experiment, as they are needed for the enhancer prediction in the next step. This step is optional, but recommended.

Usage

```
normalize(
  metaData,
  condition,
  replicate,
  genome,
  mapq = 10,
```

```
  sequencing,
  input.free = FALSE,
  chroms = NULL,
  BPPARAM = BiocParallel::SerialParam()
)
```

Arguments

metaData	A data frame containing all important information about the ChIP-seq experiments, i.e, for which histone modification they were conducted, path to the file, condition or replicate, path to control/input experiments.
condition	The condition of the sample that is supposed to be normalized
replicate	The replicate number of the sample that is supposed to be normalized
genome	Reference genome. Either a character (one of: mm9, mm10, hg19, hg38) or a SeqInfo object
mapq	Minimum mapping quality of the reads that should be included. Default is 10.
sequencing	The type of sequencing that was used. Possible options are paired and single.
input.free	[LOGICAL] Whether or not to run the function in the input free mode, in case there is no input experiment available. Default is FALSE.
chroms	A vector of strings. Specify the relevant chromosomes if needed. Then, only the reads on these chromosomes are considered for normalization and used in further analysis. Default is 'all'.
BPPARAM	An object of class SerialParam that is used as input for the BiocParallel functions.

Details

The function normalizes the ChIP-seq profiles of the three histone modifications using the counts of the provided control/input experiment: First, the binned counts (bin size: 100bp) are computed for every target (H3K4me3, H3K4me1, H2K27ac, Input) with low-quality reads being fitlered. Next, the counts of the histone modifications are normalized by input by forming the log2 fold change of the counts: counts_norm(bin) = log2((counts_raw(bin) + 1)/(counts_input(bin) + 1)). In case input experiments are not available, the input free mode will calculate the binned counts, but won't further normalize them.

Value

GRanges object containing the normalized counts

Examples

```
files <- c(system.file('extdata', 'Condition1.H3K4me1.bam', package='crupR'),
          system.file('extdata', 'Condition1.H3K4me3.bam', package='crupR'),
          system.file('extdata', 'Condition1.H3K27ac.bam', package='crupR'),
          system.file('extdata', 'Condition2.H3K4me1.bam', package='crupR'),
          system.file('extdata', 'Condition2.H3K4me3.bam', package='crupR'),
          system.file('extdata', 'Condition2.H3K27ac.bam', package='crupR'))
```

```
inputs <- c(rep(system.file('extdata', 'Condition1.Input.bam', package='crupR'), 3),
           rep(system.file('extdata', 'Condition2.Input.bam', package='crupR'),3))

metaData <- data.frame(HM = rep(c('H3K4me1','H3K4me3','H3K27ac'),2),
                       condition = c(1,1,1,2,2,2), replicate = c(1,1,1,1,1,1),
                       bamFile = files, inputFile = inputs)

normalize(metaData = metaData, condition = 1, replicate = 1,
    genome = 'mm10', sequencing = 'paired')

#Example for a customized genome:
genome = GenomeInfoDb::Seqinfo( seqnames=c('chr3', 'chr4', 'chrM'),
                                seqlengths=c(1000, 2000, 500))
```

`plotSummary` *plots boxplots of the max. enhancer prediction values of the enhancers in the condition-specific clusters*

Description

This functions provides an easy way to get an overview of the differential enhancer clusters that were identified in the getDynamics step. The boxplots depict the maximum enhancer probabilities within each enhancer region in the cluster. By examining the the distribution of predicted enhancer activties, the user can gain more insight into the clusters.

Usage

```
plotSummary(D, num_plots = 9)
```

Arguments

`D`	The getDynamics() output file of containing the GRanges object with the differential enhaners
`num_plots`	Maximal number of clusters whose plots should be displayed (clusters are sorted by their sizes). This parameter can be set in case the number of clusters is very high. Default is 9.

Value

ggplot2 object containing a boxplot for each cluster

Examples

```
#get the output of getDynamics()
files <- c(system.file('extdata', 'Condition1.H3K4me1.bam', package='crupR'),
           system.file('extdata', 'Condition1.H3K4me3.bam', package='crupR'),
           system.file('extdata', 'Condition1.H3K27ac.bam', package='crupR'),
```

```
            system.file('extdata', 'Condition2.H3K4me1.bam', package='crupR'),
            system.file('extdata', 'Condition2.H3K4me3.bam', package='crupR'),
            system.file('extdata', 'Condition2.H3K27ac.bam', package='crupR'))
inputs <- rep(system.file('extdata', 'Condition1.Input.bam', package='crupR'), 3)
inputs2 <- rep(system.file('extdata', 'Condition2.Input.bam', package='crupR'), 3)
metaData <- data.frame(HM = rep(c('H3K4me1', 'H3K4me3', 'H3K27ac'),2),
                       condition = c(1,1,1,2,2,2), replicate = c(1,1,1,1,1,1),
                       bamFile = files, inputFile = c(inputs, inputs2))
dynamics <- readRDS(system.file('extdata', 'differential_enhancers.rds', package='crupR'))
S4Vectors::metadata(dynamics) <- metaData
plotSummary(dynamics)
```

`saveFiles` *Function to save the different output objects of each step*

Description

This functions provides an easy way to save the outcomes of every step in a suitable format. The type of format available depends on the outcome itself.

Usage

```
saveFiles(data, modes, outdir, nearest = FALSE)
```

Arguments

`data`	An output object of getEnhancers(), getSE(), getDynamics() or getTargets()
`modes`	Formats in which the GRanges object should be saved. Following modes are available: for the output of getEnhancers(): 'bigWig' for a bigWig file, 'rds' for an .rds file for the output of getDynamics(): 'beds' for saving each cluster in a seperate BED file for the output of getTargets(): 'UCSC' for a UCSC interaction file for the output of getSE(): 'bedGraph' for saving the peak calls in a bedGraph file, 'bed' for saving the clusters of peaks in a BED file
`outdir`	Output directory in which the files should be saved
`nearest`	Only relevant, if you want to save the output of enhancerTargets. Specifies if the output was produced by the nearest gene mode of the function or not. Default is FALSE.

Value

True

Examples

```
# output directory
example_path <- file.path(tempdir(), 'crupR') #let's use a temporary direcotry for the outputs
dir.create(example_path) #create the directory
# recreate the output of getDynamics() to save
files <- c(system.file('extdata', 'Condition1.H3K4me1.bam', package='crupR'),
           system.file('extdata', 'Condition1.H3K4me3.bam', package='crupR'),
           system.file('extdata', 'Condition1.H3K27ac.bam', package='crupR'),
           system.file('extdata', 'Condition2.H3K4me1.bam', package='crupR'),
           system.file('extdata', 'Condition2.H3K4me3.bam', package='crupR'),
           system.file('extdata', 'Condition2.H3K27ac.bam', package='crupR'))
inputs <- rep(system.file('extdata', 'Condition1.Input.bam', package='crupR'), 3)
inputs2 <- rep(system.file('extdata', 'Condition2.Input.bam', package='crupR'), 3)
metaData <- data.frame(HM = rep(c('H3K4me1', 'H3K4me3', 'H3K27ac'),2),
                       condition = c(1,1,1,2,2,2), replicate = c(1,1,1,1,1,1),
                       bamFile = files, inputFile = c(inputs, inputs2))
clusters <- readRDS(system.file('extdata', 'differential_enhancers.rds',
package='crupR'))
S4Vectors::metadata(clusters) <- metaData
# save enhancer clusters as bed files
saveFiles(data = clusters, modes = 'beds', outdir = example_path)
```

Index