Package 'PEA'

April 5, 2018

Type Package

Title An integrated R toolkit for plant epitranscriptome analysis.

Version 1.1

Date 2018-04-05

Author Jingjing Zhai, Chuang Ma and Jie Song

Maintainer Jingjing Zhai <zhaijingjing603@gmail.com>

Depends R (>= 3.2.3), seqinr (>= 3.3-6), stringr (>= 1.2.0), randomForest (>= 4.6-12), bigmemory (>= 4.5.19), Rsamtools (>= 1.22.0), rGADEM (>= 2.18.0), snowfall (>= 1.84-6.1), ggplot2 (>= 2.2.1), seqLogo (>= 1.36.0), pROC (>= 1.5.4), ROCR (>= 1.0-5), exomePeak (>= 2.2.2), MeTPeak (>= 1.0.0), BayesPeak (>= 1.22.0), biomaRt (>= 2.26.1)

Suggests cairoDevice, gplots

Description PEA is an integrated R toolkit that aims to facilitate plant epitranscriptome data analysis. This toolkit contains a comprehensive set of functions for read mapping, CMR (chemical modifications of RNA) calling from epitranscriptome sequencing data, CMR prediction at the transcriptome scale, and CMR annotation (location distribution analysis, motif scanning and discovery, and gene functional enrichment analysis.)

LazyData true

License GPL (>= 2)

URL http://bioinfo.nwafu.edu.cn/software

NeedsCompilation no

Repository CRAN

RoxygenNote 6.0.1

R topics documented:

`classifier` *Building machine learning-based classification models*

Description

This function builds classification models with different machine learning algorithms including random forest (randomForest) and support vector machine (svm)

Usage

```
classifier( method = c("randomForest", "svm"), featureMat,
            positiveSamples, negativeSamples, ...)
```

Arguments

`method`	a character string specifying machine learning method used to construct prediction models. Currently the user-optional values are "randomForest" and "svm".
`featureMat`	a numeric matrix recording feature values for each sample.
`positiveSamples`	a character vector recording positive samples used to train classification model.
`negativeSamples`	a character vector recording negative samples used to train classification model.
`...`	Further parameters passed to train classification model.

Details

For the random forest algorithm, the important parameters are mtry (number of features randomly selected for bulding the decision tree. Default:sqrt(ncol(featureMat))) and ntree (number of trees to be built. Default:500). More information about the parameters related to random forest can be found in R package RandomForest.

For the svm algorithm, the default kernel function is the "radial"(radial basis function; RBF). Other optional kernels are the "linear", the "polynomial", and the "sigmoid" fuctions. The range of degree (for the kernel type of polynomial), gamma (for radial), coef0 (for polynomial and sigmoid) and cost (the cost parameters for all kernels) should be designated. Please refer the description of "svm" function in R package e1071 for more information about the parameters of svm.

Value

An object of class model.

Author(s)

Chuang Ma, Jingjing Zhai

Examples

```
## Not run:

   ##Load samples and feature matrix
   data(sampleData)

   positiveSamples <- sampleData$positives
   negativeSamples <- sampleData$negatives
   featureMat <- sampleData$featureMat

   ##Using ramdom forest algorithm to build machine learning model
   cl <- classifier( method = "randomForest", featureMat = featureMat,
                     positiveSamples = positiveSamples,
                     negativeSamples = negativeSamples)

## End(Not run)
```

`CMRAnnotation` *CRM Annotation*

Description

This function is designed to perform annotation analysis including CMR location and enrichment profiling, CMR-related motif scanning and motif discovery, and CMR-related gene function enrichment analysis.

Usage

```
CMRAnnotation(cmrMat, genomic = F, UTRMat = NULL, GTF = NULL, SNR = T,
              annotation = c("location", "motifScan", "motifDetect", "GO"),
              cmrSeq = NULL, RNAseq = NULL, motifPos = NULL, plot = T, ...)
```

Arguments

cmrMat	A BED format matrix representing the CMR position.
genomic	Logical, whic TRUE indicates the genomic position.
UTRMat	A matrix with six columns representing the relative position of 5'UTR, CDS and 3'UTR.
GTF	A string vector of file name, which specifies the reference genome annotation in GTF format.
SNR	Logical, where TRUE indicates the CMR is at single nucleotide resolution.
annotation	A string vector specified which annotation method would be used.
cmrSeq	A list containing CMR-realted sequences.
RNAseq	A string vector representing the RNA sequence in FASTA format.
motifPos	If SNR = F, the motifPos will be used, which indicates the relative position of specified consensus motif.
plot	Logical, where TRUE indicates perform a visualization for the results.
...	Further parameters used CMR annotation.

Value

For location analysis: A list containing:

cmrMat	The input CMR matrix.
position	The location of each CMR (e.g. 5'UTR, CDS or 3'UTR)
distribution	The relative position of CMR in the transcript.

For motif scan: A PWM matrix and motif logo.

For motif detect: A list containing consensus motifs in PWM (position weight matrix) format.

For GO enrichment: A list recording enrichment results for three subcategories (MF, BP and CC):

BP	The GO enrichment results for BP(biological process).
MF	The GO enrichment results for MF(molecular function).
CC	The GO enrichment results for CC(cellular component).

Author(s)

Jingjing Zhai, Chuang Ma, Jie Song

Examples

```
## Not run:

  ######Load data
  GTF <- system.file("extdata/chromosome1.gtf", package = "PEA")
  input.bam <- system.file("extdata/chr1_input_test.bam", package = "PEA")
  RIP.bam <- system.file("extdata/chr1_RIP_test.bam", package = "PEA")
  refGenome <- system.file("extdata/chromosome1.fa", package = "PEA")
  cDNA <- system.file("extdata/chr1_cdna.fa", package = "PEA")

  ##Extract the UTR position information from GTF file and perform location
  ##distribution analysis
  UTRMat <- getUTR(GTF = GTF)


  cmrMat <- CMRCalling(CMR = "m6A", IPBAM = RIP.bam, inputBAM = input.bam,
                       method = "SlidingWindow", mappedInput = 17472,
                       mappedRIP = 20072, refGenome = refGenome)

  results <- CMRAnnotation(cmrMat = cmrMat, SNR = F, UTRMat = UTRMat,
                           genomic = T, annotation = "location", GTF = GTF,
                           RNAseq = cDNA)

  ########################Motif analysis####################
  testSeq <- system.file("extdata/test.fa", package = "PEA")
  results.scan <- CMRAnnotation(cmrSeq = testSeq, annotation = "motifScan")
  results.detect <- CMRAnnotation(cmrSeq = testSeq, annotation = "motifDetect")

  ######################GO analysis#######################
  library(topGO)
  peaks <- G2T(bedPos = cmrMat, GTF = GTF)
  enrichment <- CMRAnnotation(cmrMat = peaks, GTF = GTF, annotation = "GO",
                              topNodes = 20, dataset = "athaliana_eg_gene")

## End(Not run)
```

`CMRCalling` *Identify CMRs (chemical modifications of RNA) from epitranscriptome sequencing data*

Description

CMRCalling is designed to identify different types of CMRs from epitranscriptome sequencing data.

Usage

```
CMRCalling(CMR = c("m6A", "m6Am", "m5C", "hm5C", "pseudoU", "m1A"),
           cpus = 1, IPBAM = NULL, inputBAM = NULL,
           GTF = NULL,  paired = FALSE, ...)
```

Arguments

CMR	A string, which specifies the CMR type.
cpus	A integer number specifying the number of cpus to be used for parallel computing.
IPBAM	A string vector of file name, which specifies the IP samples in BAM format.
inputBAM	A string vector of file name, which specifies the input control samples in BAM format.
GTF	A string vector of file name, which specifies the reference genome annotation in GTF format.
paired	Logical, if TRUE, the paired-end parameters will be used for CMR calling, otherwise, the single-end parameters would be used.
...	Further parameters used CMR calling.

Value

A matrix in BED format containing the CMRs.

Author(s)

Jingjing Zhai, Chuang Ma, Jie Song

Examples

```
## Not run:

  #Preparing sample data for peak calling
  input.bam <- system.file("extdata/chr1_input_test.bam", package = "PEA")
  RIP.bam <- system.file("extdata/chr1_RIP_test.bam", package = "PEA")
  refGenome <- system.file("extdata/chromosome1.fa", package = "PEA")
  GTF <- system.file("extdata/chromosome1.gtf", package = "PEA")


  #m6A peak calling using sliding window-based method
  #mappedInput: represents the number of mapped reads number in the input control samples.
  #mappedRIP: represents the number of mapped reads number in the RIP samples.
  cmrMat <- CMRCalling(CMR = "m6A", IPBAM = RIP.bam, inputBAM = input.bam,
                       method = "SlidingWindow", mappedInput = 17472,
                       mappedRIP = 20072, refGenome = refGenome)

  #m6A peak calling using exomePeak
  results <- CMRCalling(CMR = "m6A", method = "exomePeak", IPBAM = RIP.bam,
                       inputBAM = input.bam, GTF = GTF)

  #m6A peak calling using BayesPeak
  results <- CMRCalling(CMR = "m6A", method = "BayesPeak", IPBAM = RIP.bam,
                       inputBAM = input.bam, GTF = GTF)

  #m6A peak calling using MetPeak
  results <- CMRCalling(CMR = "m6A", method = "MetPeak", IPBAM = RIP.bam,
```

```
                      inputBAM = input.bam, GTF = GTF)

  #m6A peak calling using MACS2
  results <- CMRCalling(CMR = "m6A", method = "MACS2", IPBAM = RIP.bam,
                      inputBAM = input.bam, GTF = GTF, ...="--nomodel")

  #Pseudo uridine ratio
  inputBAM <- system.file("extdata/pseudoU_test.bam", package = "PEA")
  refGenome <- system.file("extdata/chr1_cdna.fa", package = "PEA")
  results <- CMRCalling(CMR = "pseudoU", inputBAM = inputBAM,
                       cpus = 4, refGenome = refGenome)

  #m5C calling
  fq <- system.file("extdata/test.fq", package = "PEA")
  refGenome <- system.file("extdata/chromosome1.fa", package = "PEA")
  results <- CMRCalling(CMR = "m5C", fq = fq, GTF = GTF,
                       cpus = 4, refGenome = refGenome)


## End(Not run)
```

`cross_validation` *Cross validation method*

Description

The ML-based classification model is trained and tested with N-fold cross validation method.

Usage

```
cross_validation(seed = 1, method = c("randomForest", "svm"),
                 featureMat, positives, negatives, cross = 5, cpus = 1, ...)
```

Arguments

`seed`	an integer number specifying a random seed for randomly partitioning dataset.
`method`	a character string specifying machine learning method. Possible values are "randomForest" or "nnet".
`featureMat`	a numeric feature matrix.
`positives`	a character vector reocrding positive samples
`negatives`	a character vector recording negative samples.
`cross`	number of fold for cross validation.
`cpus`	an integer number specifying the number of cpus to be used for parallel computing.
`...`	Further parameters used to cross validation. Same with the parameters used in the classifer function.

Details

In machine learning, the cross validation method has been widely used to evaluate the performance of ML-based classification models (classifiers).

For N-fold cross validation, positive and negative samples are randomly partitioned into N groups with approximately equal amount of samples, and each group is successively used for testing the performance of the ML-based classifier trained with the other N-1 groups of positive and negative samples.

For each round of cross validation, the prediction accuracy of the ML-based classifier was assessed using the receiver operating characteristic (ROC) curve analysis.The ROC curve is a two-dimensional plot of the false positive rate (FPR, x-axis) against the true positive rate (TPR, y-axis) at all possible thresholds. The value of area under the ROC curve (AUC) was used to quantitatively score the prediction accuracy of the ML-based classifer. The AUC value is ranged from 0 to 1.0, with higher AUC value indicates a better prediction accuracy of the ML-based classifer.

After N groups have been successively used as the testing set, the N sets of (FPR, TPR) pairs were imported into R package ROCR to visualize the ROC curves. The mean value of N AUCs was then computed as the overall performance of the ML-based classification model.

Value

A list recording results from each fold cross validation including the components:

`positves.train`
: positive samples used to train prediction model.

`negatives.train`
: negative samples used to train prediction model.

`positives.test`
: positive samples used to test prediction model.

`negatives.test`
: negative samples used to test prediction model.

`ml`
: machine learning method.

`classifier`
: prediction model constructed with the best parameters obtained from training dataset.

`positives.train.score`
: scores of postive samples in training dataset predicted by classifier.

`positives.train.score`
: scores of postive samples in training dataset predicted by classifier.

`positives.test.score`
: scores of postive samples in testing dataset predicted by classifier.

`negatives.test.score`
: scores of negative samples in testing dataset predicted by classifier.

`train.AUC`
: AUC value of the ML-based classifer on training dataset.

`test.AUC`
: AUC value of the ML-based classifer on testing dataset.

Author(s)

Jingjing Zhai, Chuang Ma, Jie Song

Examples

```
## Not run:

 ##Load samples and feature matrix
   data(sampleData)

   positiveSamples <- sampleData$positives
   negativeSamples <- sampleData$negatives
   featureMat <- sampleData$featureMat

   ##Perform five-fold cross-validation using random forest algorithm
   cvRes <- cross_validation(method = "randomForest", featureMat = featureMat,
                             positives = positiveSamples, negatives = negativeSamples,
                             cross = 5, cpus = 1)

## End(Not run)
```

`extractCov` *Extract the reads coverage from BAM file*

Description

Extracting reads coverage at each nucleotide using bedtools or Rsamtools.

Usage

```
extractCov(BAM, refGenome, method = c("bedtools", "Rsamtools"))
```

Arguments

`BAM`	A character vector representing file name in bam format.
`refGenome`	A character vector representing file name of reference genome in FASTA format.
`method`	A string, which specifies the method to be used. For 'bedtools', users have to install the bedtools in the terminal by typing "sudo apt-get install bedtools". if not specified, "bedtools" will be used as default.

Value

A string containing file name of coverage.

Author(s)

Jingjing Zhai, Chuang Ma, Jie Song

Examples

```
## Not run:
   ##Load data
   BAM <- system.file("extdata/chr1_input_test.bam", package = "PEA")
   refGenome <- system.file("extdata/chromosome1.fa", package = "PEA")

   results <- extractCov(BAM = BAM, refGenome = refGenome, method = "bedtools")

## End(Not run)
```

`extractSeqs` *Extract sequences according to samples*

Description

Extracting the sequences of fixed length according to the samples.

Usage

```
extractSeqs(RNAseq, samples, seqLen = 101)
```

Arguments

`RNAseq`	A character string specified the file name of cDNA sequences.
`samples`	A character vetor specified the samples.
`seqLen`	The length of extracted sequence.

Value

A list containing the extracted sequences.

Author(s)

Jingjing Zhai, Chuang Ma, Jie Song

Examples

```
## Not run:

  ##Load data
  data(sampleData)
  positiveSamples <- sampleData$positives
  cDNA <- system.file("extdata/chr1_cdna.fa", package = "PEA")
  posSeq <- extractSeqs(RNAseq = cDNA, samples = positiveSamples, seqLen = 101)

## End(Not run)
```

`featureEncoding` *Feature encoding*

Description

This function contains three feature encoding scheme, binary, k-mer and PseDNC. For binary encoding scheme, a vector of 404 (4*101) features is generated through assigning 'A', 'C', 'G', 'U' and 'N' with (1,0,0,0), (0,1,0,0), (0,0,1,0), (0,0,0,1) and (0,0,0,0), respectively. Here 'N' is a gap used to ensure the fixed features of each sample, if an m6A/non- m6A site occurs near the initiation or termination of the transcript. For K-mer encoding, the composition of short sequence with different lengths was considered to encoding samples. For PseDNC (pseudo dinucleotide composition) encoding, the local and global sequence-order information along the RNA sequence was used for scoring the each sample.

Usage

```
featureEncoding(RNAseq, lambda = 6, w = 0.9)
```

Arguments

`RNAseq`	A list containing the FASTA format sequences.
`lambda`	The lambda parameter for the PseDNC-related features, default is 6.
`w`	The weighting parameter for PseDNC-related features, default is 0.9.

Value

A matrix with features.

Author(s)

Jingjing Zhai, Chuang Ma, Jie Song

Examples

```
## Not run:

  ##Load data
  data(sampleData)
  positiveSamples <- sampleData$positives
  cDNA <- system.file("extdata/chr1_cdna.fa", package = "PEA")

  ##Extracting flanking sequences (101-nt) of positive samples
  posSeq <- extractSeqs(RNAseq = cDNA, samples = positiveSamples, seqLen = 101)

  ##Performing feature encoding using Binary, k-mer and PseDNC encoding schmes
  posFeatureMat <-  featureEncoding(RNAseq = posSeq)


## End(Not run)
```

`findConfidentPosSamples` *Find confident positive samples*

Description

The confident positive samples are identified if only one CMR signal existed in the corresponding peak.

Usage

```
findConfidentPosSamples(peaks, RNAseq = NULL, motifPos = NULL, ...)
```

Arguments

`peaks`	A BED format matrix containing the peak regions.
`RNAseq`	A character string specified the file name of cDNA sequences.
`motifPos`	A list containing the position of the consensus motif for each transcript.
`...`	Further paramters used for find confident positive samples.

Value

A list containing two elements: the positive samples and its transcript IDs.

Author(s)

Jingjing Zhai, Chuang Ma, Jie Song

Examples

```
## Not run:

  ##Load data
  input.bam <- system.file("extdata/chr1_input_test.bam", package = "PEA")
  RIP.bam <- system.file("extdata/chr1_RIP_test.bam", package = "PEA")
  refGenome <- system.file("extdata/chromosome1.fa", package = "PEA")
  cDNA <- system.file("extdata/chr1_cdna.fa", package = "PEA")
  GTF <- system.file("extdata/chromosome1.gtf", package = "PEA")

  ##m6A peak calling using sliding window-based method
  cmrMat <- CMRCalling(CMR = "m6A", IPBAM = RIP.bam, inputBAM = input.bam,
                       method = "SlidingWindow", mappedInput = 17472,
                       mappedRIP = 20072, refGenome = refGenome)


  #Convert genomic position to cDNA position
  peaks <- G2T(bedPos = cmrMat, GTF = GTF)
```

```
  ##Search consensus motif in cDNA sequence
  motifPos <- searchMotifPos(sequence = cDNA)

  ##Find confident positive samples
  posSamples <- findConfidentPosSamples(peaks = peaks, motifPos = motifPos)


## End(Not run)
```

`findUnlabelSamples` *Find unlabeleded samples*

Description

The unlabeled samples are extracted from the same set of transcripts with consensus motif (e.g. RRACH) of confident positive samples, but exclude the confident positive samples.

Usage

```
findUnlabelSamples(cDNAID, motifPos, posSamples, ...)
```

Arguments

`cDNAID`	A character vector representing the transcripts IDs.
`motifPos`	A list containing the position of the consensus motif for each transcript.
`posSamples`	A character vetor specified the positive samples, which can be generated by function "findConfidentPosSamples".
`...`	Further paramters used for find confident positive samples.

Value

A list containing two elements: the positive samples and its transcript IDs.

Author(s)

Jingjing Zhai, Chuang Ma, Jie Song

Examples

```
## Not run:

  ##Load data
  input.bam <- system.file("extdata/chr1_input_test.bam", package = "PEA")
  RIP.bam <- system.file("extdata/chr1_RIP_test.bam", package = "PEA")
  refGenome <- system.file("extdata/chromosome1.fa", package = "PEA")
  cDNA <- system.file("extdata/chr1_cdna.fa", package = "PEA")
  GTF <- system.file("extdata/chromosome1.gtf", package = "PEA")
```

```
##m6A peak calling using sliding window-based method
cmrMat <- CMRCalling(CMR = "m6A", IPBAM = RIP.bam, inputBAM = input.bam,
                     method = "SlidingWindow", mappedInput = 17472,
                     mappedRIP = 20072, refGenome = refGenome)



#Convert genomic position to cDNA position
peaks <- G2T(bedPos = cmrMat, GTF = GTF)

##Search consensus motif in cDNA sequence
motifPos <- searchMotifPos(sequence = cDNA)

##Find confident positive samples
posSamples <- findConfidentPosSamples(peaks = peaks, motifPos = motifPos)


##Find unlabel samples
unlabelSamples <- findUnlabelSamples(cDNAID = posSamples$cDNAID,
                                     motifPos = motifPos,
                                     posSamples = posSamples$positives)



## End(Not run)
```

G2T *Convert the genomic position to transcriptomic position*

Description

According to the reference genome GTF annotation file, transform the genomic position to transcriptomic position, if one position can be mapped multiple transcripts, all of them will be retained.

Usage

```
G2T(bedPos, GTF)
```

Arguments

bedPos	A matrix in BED format containing genomic position.
GTF	A string vector of file name, which specifies the reference genome annotation in GTF format.

Value

A list recording transcriptomic position.

Author(s)

Jingjing Zhai, Chuang Ma, Jie Song

Examples

```
## Not run:

 #######Load data
 input.bam <- system.file("extdata/chr1_input_test.bam", package = "PEA")
 RIP.bam <- system.file("extdata/chr1_RIP_test.bam", package = "PEA")
 refGenome <- system.file("extdata/chromosome1.fa", package = "PEA")
 GTF <- system.file("extdata/chromosome1.gtf", package = "PEA")


 cmrMat <- CMRCalling(CMR = "m6A", IPBAM = RIP.bam, inputBAM = input.bam,
                      method = "SlidingWindow", mappedInput = 17472,
                      mappedRIP = 20072, refGenome = refGenome)

 ###Convert genomic position to cDNA position
 peaks <- G2T(bedPos = cmrMat, GTF = GTF)


## End(Not run)
```

---

getUTR *Get the genomic features for each transcript*

Description

According to the reference genome GTF annotation file, a matrix containing the five prime UTR, CDS and three prime UTR positional information will be generated.

Usage

```
getUTR(GTF, cpus = 1)
```

Arguments

GTF	A string vector of file name, which specifies the reference genome annotation in GTF format.
cpus	an integer number specifying the number of cpus to be used for parallel computing.

Value

A matrix with six columns containing:

five_UTR_Start	The start position for five prime UTR.

five_UTR_End	The end position for five prime UTR.
CDS_Start	The start position for CDS.
CDS_End	The end position for CDS.
three_UTR_Start	The start position for three prime UTR.
three_UTR_End	The end position for three prime UTR.

Author(s)

Jingjing Zhai, Chuang Ma, Jie Song

Examples

```
## Not run:

  GTF <- system.file("extdata/chromosome1.gtf", package = "PEA")
  UTRMat <- getUTR(GTF = GTF)

## End(Not run)
```

---

motifDetect *De novo motif discovery in the peak regions*

Description

The de novo motif discovery is performed through genetic algorithm in 'rGADEM' package.

Usage

```
motifDetect(sequence, plot = T, ...)
```

Arguments

sequence	A character vector representing the file name of the sequence in FASTA format.
plot	Logical, where TRUE indicates that perform visualization for the results.
...	Further paramters used for motif detection, see GADEM.

Value

A list containing consensus motifs in PWM (position weight matrix) format.

Author(s)

Jingjing Zhai, Chuang Ma, Jie Song

Examples

```
## Not run:
  testSeq <- system.file("extdata/test.fa", package = "PEA")
  results <- motifDetect(sequence = testSeq)

## End(Not run)
```

---

motifScan *Scanning the motifs in the peak regions*

Description

For a given motif, all peak regions will be scanned and then calculates a position weight matrix(PWM) for the analyzed motifs. Finally, a motif logo will be generated to show the positional differences in the frequency of four nucleotides.

Usage

```
motifScan(sequence, motif = "[ag][ag]ac[act]")
```

Arguments

sequence	A character vector representing the file name of the sequence in FASTA format.
motif	A string, which specifies the motif to be searched.

Value

A PWM matrix and motif logo.

Author(s)

Jingjing Zhai, Chuang Ma, Jie Song

Examples

```
## Not run:
  testSeq <- system.file("extdata/test.fa", package = "PEA")
  results <- motifScan(sequence = testSeq)

## End(Not run)
```

peakCalling *Peak Calling*

Description

Identify genomic regions in a genome that have been enriched with aligned reads as a consequence of performing MeRIP-seq or ChIP-seq experiment.

Usage

```
peakCalling(IPBAM, inputBAM, GTF, expName = NULL,
            method = c("SlidingWindow","exomePeak", "MetPeak",
                       "MACS2", "BayesPeak"), paired = F, ...)
```

Arguments

IPBAM	A string vector of file name, which specifies the IP samples in BAM format.
inputBAM	A string vector of file name, which specifies the input control samples in BAM format.
GTF	A string vector of file name, which specifies the reference genome annotation in GTF format.
expName	A string, which specifies the name generated in the output directory which contains all the results.
paired	Logical, if TRUE, the paired-end parameters will be used for peak calling, otherwise, the single-end parameters would be used.
method	A string, which specifies the method used for peak calling, the default is "SlidingWindow".
...	Further parameters used peak calling.

Value

A list containing two elements:

peaks	A enriched peak region matrix in BED format.
method	A string represent which method was used for peak calling.

Author(s)

Jingjing Zhai, Chuang Ma, Jie Song

Examples

```
## Not run:

  #Preparing sample data for peak calling
  input.bam <- system.file("extdata/chr1_input_test.bam", package = "PEA")
  RIP.bam <- system.file("extdata/chr1_RIP_test.bam", package = "PEA")
  refGenome <- system.file("extdata/chromosome1.fa", package = "PEA")

  cmrMat <- peakCalling(IPBAM = RIP.bam, inputBAM = input.bam,
                        method = "SlidingWindow", mappedInput = 17472,
                        mappedRIP = 20072, refGenome = refGenome)


## End(Not run)
```

`plotROC` *Plotting receiver operating characteristic(ROC) Curves*

Description

This function plots ROC curves for estimating the performance of machine learning-based classification model in cross validation experiments.

Usage

```
plotROC(cvRes)
```

Arguments

`cvRes`	results from the "cross_validation" function.

Value

A ROC plot

Author(s)

Chuang Ma, Jingjing Zhai

Examples

```
## Not run:

 ##Load samples and feature matrix
   data(sampleData)

   positiveSamples <- sampleData$positives
   negativeSamples <- sampleData$negatives
   featureMat <- sampleData$featureMat
```

```
  ##Perform five-fold cross-validation using random forest algorithm
  cvRes <- cross_validation(method = "randomForest", featureMat = featureMat,
                            positives = positiveSamples, negatives = negativeSamples,
                            cross = 5, cpus = 1)
plotROC(cvRes = cvRes)


## End(Not run)
```

---

predCMR *Predicting the CMR status using machine learning-based models*

Description

This function is used to perform CMR prediction using machine learning-based model.

Usage

```
predCMR(predSeqs, seqLen = 101, motif = "[ag][ag]ac[act]", model, ...)
```

Arguments

predSeqs	a character string specifying the directory of FASTA format sequences.
seqLen	The length of extracted sequence.
motif	A string, which specifies the motif to be searched.
model	A class object representing the model.
...	Further parameters passed to predict CMRs.

Value

A matrix with two columns representing the transcript IDs and corresponding scores.

Author(s)

Chuang Ma, Jingjing Zhai

Examples

```
## Not run:

  data(sampleData)

  positiveSamples <- sampleData$positives
  negativeSamples <- sampleData$negatives
  featureMat <- sampleData$featureMat
```

```
   #Saving psol results in working directory
   dir.create("psol", showWarnings = F)
   psolResDic <- paste0(getwd(), "/psol/")
   psolRes <- PSOL(featureMatrix = featureMat, positives = positiveSamples,
                   unlabels = negativeSamples, PSOLResDic = psolResDic,
                   cpus = 1)


   predSeq <- system.file("extdata/test_pred.fa", package = "PEA")
   predMat <- predCMR(predSeq, model = psolRes$model)

## End(Not run)
```

`pseudoURatio` *Pseudouridine ratio*

Description

Calculate the pseudouridine ratio from the pseudo-seq data.

Usage

```
pseudoURatio(refGenome, inputBAM, cpus = 1)
```

Arguments

`refGenome`	A character vector representing file name of reference genome in FASTA format.
`inputBAM`	A string vector of file name, which specifies the input control samples in BAM format.
`cpus`	an integer number specifying the number of cpus to be used for parallel computing.

Value

A list containing the position and ratio for each pseudouridine.

Author(s)

Jingjing Zhai, Chuang Ma, Jie Song

References

Carlile, T.M., et al. Pseudouridine profiling reveals regulated mRNA pseudouridylation in yeast and human cells. Nature 2014;515(7525):143-146.

Examples

```
## Not run:

  inputBAM <- system.file("extdata/pseudoU_test.bam", package = "PEA")
  referenceGenome <- system.file("extdata/chr1_cdna.fa", package = "PEA")
  results <- pseudoURatio(refGenome = referenceGenome,
                          inputBAM = inputBAM, cpus = 4)

## End(Not run)
```

PSOL *Machine learning-based PSOL(positive samples only learning algorithm) algorithm*

Description

Firstly, this function selects an initial negative set with the machine learning(ML)-based positive-only sample learning (PSOL) algorithm. The PSOL algorithm has been previously applied to predict stress-response genes and genomic loci encoding functional noncoding RNAs (see References).

Usage

```
PSOL(featureMatrix = NULL, positives, balanced = FALSE, ratio = 10,
     unlabels, cpus = 1, PSOLResDic = NULL, iterator = 10, TPR = 0.995,
     cross = 5, method = c("randomForest", "svm"), ... )
```

Arguments

`featureMatrix`	A numeric matrix recording the features for all sample.
`positives`	A character vector recording positive samples
`balanced`	A logical value, where TRUE represents balance the positive and negative samples according to the ratio.
`ratio`	A numuric value of the the ratio between negative and positive samples.
`unlabels`	A character vector recording unlabeled samples.
`cpus`	An integer number specifying the number of cpus will be used for parallel computing.
`PSOLResDic`	A character string, PSOL Result directory
`iterator`	An integer calue, iterator times.
`TPR`	A numeric value ranged from 0 to 1.0, used to select the prediction score cutoff.
`cross`	An integer value, cross-times cross validation.
`method`	A character string, machine learing method.
`...`	Further parameters used for PSOL.

Value

A list containing three components:

positives	a character vector including the input positive samples.
negatives	a character vector recording the selected negative samples.
model	a class object representing the model.

Author(s)

Jingjing Zhai, Chuang Ma, Jie Song.

References

[1] Chunlin Wang, Chris Ding, Richard F. Meraz and Stephen R. Holbrook. PSoL: a positive sample only learning algorithm for finding non-coding RNA genes. Bioinformatics, 2006, 22(21): 2590-2596.

[2] Chuang Ma, Xiangfeng Wang. Machine learning-based differential network analysis: a case study of stress-responsive transcriptomes in Arabidopsis thaliana. 2013(Submitted).

Examples

```
## Not run:

   data(sampleData)

   positiveSamples <- sampleData$positives
   negativeSamples <- sampleData$negatives
   featureMat <- sampleData$featureMat

   #Saving psol results in working directory
   dir.create("psol", showWarnings = F)
   psolResDic <- paste0(getwd(), "/psol/")
   psolRes <- PSOL(featureMatrix = featureMat, positives = positiveSamples,
                   unlabels = negativeSamples, PSOLResDic = psolResDic,
                   cpus = 1)


## End(Not run)
```

QCBAM *Quality control for BAM format files*

Description

Performing quality control for mapping results in BAM format.

Usage

```
QCBAM(BAM, quality = c(35, 30), paired = F)
```

Arguments

BAM	A character vector representing file name in bam format.
quality	A vector which specifies the base-call quality score to be filtered for single-end and paired-end, respectively.
paired	Logical, where TRUE indicates the BAM files are generated with paired-end reads.

Value

A list containing filtered files which passed the quality control.

Author(s)

Jingjing Zhai, Chuang Ma, Jie Song

Examples

```
## Not run:
   BAM <- system.file("extdata/chr1_input_test.bam", package = "PEA")
   results <- QCBAM(BAM = BAM)

## End(Not run)
```

---

readMapping *Reads mapping*

Description

Mapping FASTQ format reads to the reference genome with user adjustable method.

Usage

```
readMapping(alignment = c("bowtie", "bowtie2", "tophat",
                          "tophat2", "hisat", "hisat2"),
                        fq, index = NULL, refGenome = NULL,
                        paired = FALSE, softwareDir = NULL, ...)
```

Arguments

alignment	A string, which specifies the method to be used for reads mapping.
fq	A character vector representing file name in FASTQ format.
index	A character vector representing file name of index used for alignment. If not specified, it will generate index files automaticly.
refGenome	A character vector representing file name of reference genome in FASTA format.
paired	Logical, if TRUE, the paired-end parameters will be used for reads mapping, otherwise, the single-end parameters would be used.
softwareDir	Optional, if the corresponding alignment tools are not in the environment variables, please specify the directory of corresponding tools.
...	Further parameters used for reads mapping.

Value

A list recording the results including:

command	The command used for reads mapping.
resDir	A character vector containg the directory of alignment files.

Author(s)

Jingjing Zhai, Chuang Ma, Jie Song

Examples

```
## Not run:

  #Loading sample data for reads mapping
  fq <-  system.file("extdata/test.fq", package = "PEA")
  referenceGenome <- system.file("extdata/chromosome1.fa", package = "PEA")

  #reads mapping using tophat with default parameter
  test.bam <- readMapping(alignment = 'tophat', fq = fq,
                          refGenome = referenceGenome, paired = F)

  #reads mapping using tophat with 2 threads
  test.bam <- readMapping(alignment = 'tophat', fq = fq,
                          refGenome = referenceGenome, paired = F,
                          ... = "-p 2")


## End(Not run)
```

runTopGO *Go enrichment*

Description

GO (Gene Ontology) enrichment analysis through "topGO" package.

Usage

```
runTopGO(geneID, statistic = "fisher", algorithm = "elim",
         topNodes = 20,
         dataset = "athaliana_eg_gene", plot = TRUE)
```

Arguments

geneID	A character vector representing the gene set to be used for GO enrichment.
statistic	A character string specifing which test to use.
algorithm	A character string specifing which algorithm to use.
topNodes	The number of top GO terms to be included in the results.
dataset	A character string specifing which dataset to use, the default is: athaliana_eg_gene; For other plant species, please use "listDatasets(mart = useMart(biomart = "plants_mart", host = 'plants.ensembl.org'))" to see all available datasets.
plot	Logical, if TRUE, a dot plot will be generated to represent the significance of each GO term.

Value

A list recording enrichment results for three subcategories (MF, BP and CC).

BP	The GO enrichment results for BP(biological process).
MF	The GO enrichment results for MF(molecular function).
CC	The GO enrichment results for CC(cellular component).

Author(s)

Jingjing Zhai, Chuang Ma, Jie Song

Examples

```
## Not run:
  ##Please ensure packages "biomaRt" and "topGO" are pre-installed.
  library(biomaRt)
  library(topGO)
  mart <- biomaRt::useMart(biomart = "plants_mart", dataset = "athaliana_eg_gene",
                           host = 'plants.ensembl.org')
  GTOGO <- biomaRt::getBM(attributes = c( "ensembl_gene_id", "go_id"), mart = mart)
```

```
  GTOGO <- GTOGO[GTOGO$go_id != '', ]
  all.genes <- sort(unique(as.character(GTOGO$ensembl_gene_id)))
  geneID <- sample(all.genes, 100)
  results <- runTopGO(geneID = geneID)

## End(Not run)
```

---

sampleData *example data for PEA*

Description

positive and negative samples (m6A/non-m6A modifications) and sequence-based feature matrix.

Usage

```
data(sampleData)
```

Examples

```
## Not run:
   ##load datasets
   data(sampleData)

   ##load positive samples
   positives <- PEA$positives

   ##load negative samples
   negatives <- PEA$negatives

   ##Load sequence-based feature matrix
   featuresMat <- PEA$featureMat

## End(Not run)
```

---

searchMotifPos *Scanning the motif in the sequence*

Description

For a given motif, all sequence will be scanned return the central position for the motif.

Usage

```
searchMotifPos(sequence, motif = "[ag][ag]ac[act]", cenPos = 2)
```

Arguments

sequence	A character vector representing the file name of the sequence in FASTA format.
motif	A string, which specifies the motif to be searched.
cenPos	Integer represents the central position for the motif.

Value

A list containing the motif central position.

Author(s)

Jingjing Zhai, Chuang Ma, Jie Song

Examples

```
## Not run:

  cDNA <- system.file("extdata/chr1_cdna.fa", package = "PEA")
  results <- searchMotifPos(sequence = cDNA)

## End(Not run)
```

SlidingWindow *Fisher exact test based sliding window*

Description

Peak calling through Fisher exact test based sliding window method (see References).

Usage

```
SlidingWindow(input, RIP, mappedInput = NULL,
              mappedRIP = NULL, level = 0.05,
              ratio = -2, ...)
```

Arguments

input	A string vector of file name, which specifies the input control samples in BAM format.
RIP	A string vector of file name, which specifies the IP samples in BAM format.
mappedInput	Integer, which represents the number of mapped reads number in the input control samples.
mappedRIP	Integer, which represents the number of mapped reads number in the IP samples.
level	Significance level for the p-value.
ratio	The ratio between the normalized mapped reads number in IP and input samples, respectively.
...	Further parameters used for peak calling.

Value

A matrix in BED format containing the enriched peaks.

Author(s)

Jingjing Zhai, Chuang Ma, Jie Song

References

Meyer, K.D., ea al. Comprehensive analysis of mRNA methylation reveals enrichment in 3'UTRs and near stop codons. Cell 2012; 149(7):1635-1646

Examples

```
## Not run:
  ##Load data
  input.bam <- system.file("extdata/chr1_input_test.bam", package = "PEA")
  RIP.bam <- system.file("extdata/chr1_RIP_test.bam", package = "PEA")
  refGenome <- system.file("extdata/chromosome1.fa", package = "PEA")


  ########################peak calling#################################
  peaks <- SlidingWindow(RIP = RIP.bam, input = input.bam,
                         mappedInput = 17472,
                         mappedRIP = 20072, refGenome = refGenome)


## End(Not run)
```

Index