Package 'cleanUpdTSeq'

December 9, 2024

Type Package

Title cleanUpdTSeq cleans up artifacts from polyadenylation sites from oligo(dT)-mediated 3' end RNA sequending data

Description This package implements a Naive Bayes classifier for accurately differentiating true polyadenylation sites (pA sites) from oligo(dT)-mediated 3' end sequencing such as PAS-Seq, PolyA-Seq and RNA-Seq by filtering out false polyadenylation sites, mainly due to oligo(dT)-mediated internal priming during reverse transcription. The classifer is highly accurate and outperforms other heuristic methods.

Version 1.45.0

Date 2021-04-27

Author Sarah Sheppard, Haibo Liu, Jianhong Ou, Nathan Lawson, Lihua Julie Zhu

Maintainer Jianhong Ou <Jianhong.Ou@duke.edu>; Lihua Julie Zhu <Julie.Zhu@umassmed.edu>

Depends R (>= 3.5.0), BSgenome.Drerio.UCSC.danRer7, methods

Imports BSgenome, GenomicRanges, seqinr, e1071, Biostrings, GenomeInfoDb, IRanges, utils, stringr, stats, S4Vectors

Suggests BiocStyle, rmarkdown, knitr, RUnit, BiocGenerics (>= 0.1.0)

License GPL-2

biocViews Sequencing, 3' end sequencing, polyadenylation site, internal priming

VignetteBuilder knitr

RoxygenNote 7.1.2

LazyData true

Encoding UTF-8

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

git_branch devel

git_last_commit 0239c1b

git_last_commit_date 2024-10-29

Repository Bioconductor 3.21

Date/Publication 2024-12-09

Contents

`BED6WithSeq2GRangesSeq` *Covert (extended) BED6 file to a GRanges object*

Description

Convert to a GRanges object from a (extended) BED6 file with at least six columns: chrom, chromStart, strEnd, name, score and strand, and optional upstream sequences (including pA sites) and downstream sequences of pA sites

Usage

```
BED6WithSeq2GRangesSeq(
  file,
  skip = 1L,
  withSeq = TRUE,
  upstream.seq.ind = 7L,
  downstream.seq.ind = 8L
)
```

Arguments

`file`	A character(1) vector, representing a path to a extended BED file containing at least six columns in the order of chrom, chromStart, strEnd, name, score and strand. The strand information must be designated as "+", or "-". Optional fields–upstream sequences (including pA sites) and downstream sequences of pA sites–are allowed. For more details about the BED format, see https://genome.ucsc.edu/FAQ/FAQform
`skip`	A integer(1) vector, indicating how many rows (header lines) to skip when the BED file is read into R.
`withSeq`	A logical(1) vector, indicating that upstream and downstream sequences flanking pA sites are included in the file

upstream.seq.ind
: An integer(1),vector delineating the column location of upstream sequences of the putative pA site

downstream.seq.ind
: An integer(1),vector delineating the column location of downstream sequences of the putative pA site

Value

An object of GRanges

Author(s)

Haibo Liu, Lihua J. Zhu

Examples

```
testFile <- system.file("extdata", "test.bed",
                        package = "cleanUpdTSeq")
peaks <- BED6WithSeq2GRangesSeq(file = testFile,
                                skip = 1L, withSeq = TRUE)
```

---

buildClassifier *Build a Naive Bayes Classifier*

Description

Computes the conditional a-posterior probabilities of a categorical class variable given independent predictor variables using the Bayes rule.

Usage

```
buildClassifier(
  Ndata.NaiveBayes,
  Pdata.NaiveBayes,
  upstream = 40L,
  downstream = 30L,
  wordSize = 6L,
  alphabet = c("ACGT")
)
```

Arguments

Ndata.NaiveBayes
: A data.frame, containing features for the negative training data, described further in data.NaiveBayes.

Pdata.NaiveBayes
: A data.frame, containing features for the positive training data, described further in data.NaiveBayes.

upstream
: An integer(1) vector, length of upstream sequence to retrieve.

downstream
: An integer(1) vector, length of downstream sequence to retrieve.

wordSize
: An integer(1) vector, size of the kmer feature for the upstream sequence. wordSize = 6 should always be used.

alphabet
: A character(1) vector, a string containing DNA bases. By default, "ACTG".

Value

An object of class "naiveBayes".

Author(s)

Jianhong Ou

See Also

naiveBayes

Examples

```
if (interactive()){
    data(data.NaiveBayes)
    classifier <- buildClassifier(data.NaiveBayes$Negative,
                                  data.NaiveBayes$Positive)
}
```

---

buildFeatureVector *build Feature Vector_2*

Description

This function creates a data frame. Fields include peak name, upstream sequence, downstream sequence, and features to be used in classifying the putative polyadenylation site.

Usage

```
buildFeatureVector(
  peaks,
  genome = Drerio,
  upstream = 40L,
  downstream = 30L,
  wordSize = 6L,
  alphabet = "ACGT",
```

```
  sampleType = c("TP", "TN", "unknown"),
  replaceNAdistance = 30L,
  method = c("NaiveBayes", "SVM"),
  fetchSeq = FALSE,
  return_sequences = FALSE
)
```

Arguments

peaks	An object of GRanges that may contain the upstream and downstream sequence information. This item is created by the function BED6WithSeq2GRangesSeq.
genome	Name of the genome to get sequences from. To find out a list of available genomes, please type BSgenome::available.genomes() in R.
upstream	An integer(1) vector, length of upstream sequence to retrieve.
downstream	An integer(1) vector, length of downstream sequence to retrieve.
wordSize	An integer(1) vector, size of the kmer feature for the upstream sequence. wordSize = 6 should always be used.
alphabet	A character(1) vector, a string containing DNA bases. By default, "ACTG".
sampleType	A character(1) vector, indicating type of sequences for building feature vectors. Options are TP (true positive) and TN (true negative) for training data, or unknown for test data.
replaceNAdistance	An integer(1) vector, specifying an number for avg.distanceA2PeakEnd, the average distance of As to the putative pA site, when there is no A in the downstream sequence.
method	A character(1) vector, specifying a machine learning method to to use. Currently, only "NaiveBayes" is implemented.
fetchSeq	A logical (1), indicating whether upstream and downstream sequences should be retrieved from the BSgenome object at this step or not.
return_sequences	A logical(1) vector, indicating whether upstream and downstream sequences should be included in the output

Value

An object of "featureVector"

Author(s)

Sarah Sheppard, Haibo Liu, Jianhong Ou, Nathan Lawson, Lihua J. Zhu

Examples

```
library(BSgenome.Drerio.UCSC.danRer7)
testFile <- system.file("extdata", "test.bed",
                        package = "cleanUpdTSeq")
peaks <- BED6WithSeq2GRangesSeq(file = testFile,
```

```
                                skip = 1L, withSeq = TRUE)
## build the feature vector for the test set with sequence information
testSet.NaiveBayes = buildFeatureVector(peaks,
                                        genome = Drerio,
                                        upstream = 40L,
                                        downstream = 30L,
                                        wordSize = 6L,
                                        alphabet = "ACGT",
                                        sampleType = "unknown",
                                        replaceNAdistance = 30,
                                        method = "NaiveBayes",
                                        fetchSeq = FALSE,
                                        return_sequences = TRUE)

## convert the test set to GRanges without upstream and downstream
## sequence information
peaks <- BED6WithSeq2GRangesSeq(file = testFile,
                                skip = 1L, withSeq = FALSE)
#build the feature vector for the test set without sequence information
testSet.NaiveBayes = buildFeatureVector(peaks,
                                        genome = Drerio,
                                        upstream = 40L,
                                        downstream = 30L,
                                        wordSize = 6L,
                                        alphabet = "ACGT",
                                        sampleType = "unknown",
                                        replaceNAdistance = 30,
                                        method = "NaiveBayes",
                                        fetchSeq = TRUE,
                                        return_sequences = TRUE)
```

`classifier` *NaiveBayes classifier*

Description

An object of class "naiveBayes" generated from data.NaiveBayes

Usage

```
classifier
```

Format

An object of class "PASclassifier" including components:

Examples

```
data(classifier)
names(classifier)
```

`cleanUpdTSeq` *This package classifies putative polyadenylation sites.*

Description

3'ends of transcripts have generally been poorly annotated. With the advent of deep sequencing, many methods have been developed to identify 3'ends. The majority of these methods use an oligodT primer which can bind to internal adenine-rich sequences, and lead to artifactual identification of polyadenylation sites. Heuristic filtering methods rely on a certain number of As downstream of a putative polyadenylation site to classify the site as true or oligodT primed. This package provides a robust method to classify putative polyadenylation sites using a Naive Bayes classifier.

Author(s)

Sarah Sheppard, Haibo Liu, Jianhong Ou, Nathan Lawson, Lihua Julie Zhu

---

`data.NaiveBayes` *Training Data*

Description

A RData containing negative and positive training data

Usage

```
data.NaiveBayes
```

Format

A list with 2 data frame, "Negative" and "Positive". Negative has 9219 observations on the following 4120 variables. And Positive is a data frame with 22770 observations on the following 4120 variables. The format is:

list("Negative") 'data.frame': 9219 obs. of 4120 variables:

list("Positive") 'data.frame': 22770 obs. of 4120 variables:

Both of them have same structure.

list("y") a numeric vector

list("n.A.Downstream") a numeric vector

list("n.C.Downstream") a numeric vector

list("n.T.Downstream") a numeric vector

list("n.G.Downstream") a numeric vector

list("avg.distanceA2PeakEnd") a numeric vector

list("dimer") a numeric vector

: such as AA, AC, AG, AT, CA, ... etc. a numeric vector

list("heximer") a factor with levels `0` `1`

: such as AAAAAA, ACGTAC, ... etc. a factor with levels `0` `1`

list("upstream.seq") a vector of sequence string

list("downstream.seq") a vector of sequence string

Examples

```
library(BSgenome.Drerio.UCSC.danRer7)
data(data.NaiveBayes)
head(str(data.NaiveBayes$Negative))
head(str(data.NaiveBayes$Positive))
```

`featureVector-class` *Class* `"featureVector"`

Description

An object of class `"featureVector"` represents the output of `buildFeatureVector`

Objects from the Class

Objects can be created by calls of the form `new("featureVector", data, info)`.

`getContextSequences` *Retrieve upstream and downstream sequences*

Description

Retrieve upstream and downstream sequences of pA sites from a BSgenome object based on a GRanges object

Usage

```
getContextSequences(peaks, upstream = 40L, downstream = 30L, genome)
```

Arguments

peaks	An object of GRanges representing pA sites
upstream	An integer(1) vector, length of upstream sequence of pA sites, including pA site.
downstream	An integer(1) vector, length of downstream sequences of pA sites
genome	An object of BSgenome.

Value

A data.frame containing sequences upstream and downstream pA sites:

upstream.seq sequence upstream pA site, including pA site

downstream.seq sequence downstream pA site

Author(s)

Haibo Liu

Examples

```
library(BSgenome.Drerio.UCSC.danRer7)
testFile <- system.file("extdata", "test.bed",
                        package = "cleanUpdTSeq")
peaks <- BED6WithSeq2GRangesSeq(file = testFile,
                                skip = 1L, withSeq = FALSE)
peaks_seq <- getContextSequences(peaks,
                                 upstream = 40L,
                                 downstream = 30L,
                                 genome = Drerio)
```

---

modelInfo-class *Class* "modelInfo"

Description

An object of class "modelInfo" represents the information of sequence to use in the analysis

Objects from the Class

Objects can be created by calls of the form new("modelInfo", upstream, downstream, wordSize, alphabet).

naiveBayes-class *Class* "naiveBayes"

Description

An object of class "naiveBayes" represents the conditional a-posterior probabilities of a categorical class variable given independent predictor variables using the Bayes rule.

Objects from the Class

Objects can be created by calls of the form new("naiveBayes", apriori, tables, levels, call).

PASclassifier-class *Class* "PASclassifier"

Description

An object of class "PASclassifier" represents the output of buildClassifier

Objects from the Class

Objects can be created by calls of the form new("PASclassifier", classifier, info).

Examples

```
data(classifier)
classifier$info$upstream
classifier$info$wordSize
classifier$info$alphabet
```

predictTestSet *predict authenticity of putative pA sites*

Description

classify putative pA sites into true and false bins.

Usage

```
predictTestSet(
  Ndata.NaiveBayes = NULL,
  Pdata.NaiveBayes = NULL,
  testSet.NaiveBayes,
  classifier = NULL,
  outputFile = "test-predNaiveBayes.tsv",
  assignmentCutoff = 0.5,
  return_sequences = FALSE
)
```

Arguments

Ndata.NaiveBayes	A data.frame, containing features for the negative training data, which is built using the function buildFeatureVector. It is described further indata.NaiveBayes.
Pdata.NaiveBayes	A data.frame, containing features for the positive training data, which is built using the function buildFeatureVector. It is described further indata.NaiveBayes.
testSet.NaiveBayes	An object of featureVector for test data built for Naive Bayes analysis using the function buildFeatureVector.
classifier	An object of class PASclassifier.
outputFile	A character(1) vector, file name for outputting prediction results. The prediction output is written to the file, tab separated.
assignmentCutoff	A numeric(1) vector, specifying the cutoff for classifying a putative pA site into a true or false pA class. It should be any number between 0 and 1. For example, assignmentCutoff = 0.5 will assign an putative pA site with prob_true_pA > 0.5 to the True class (1), and any putative pA site with prob_true_pA < = 0.5 as False (0).
return_sequences	A logical(1) vector, indicating whether upstream and downstream sequences should be included in the output

Value

A data.frame including all info as described below. The upstream and downstream sequence used in assessing the putative pA site might be included when return_sequences = TRUE.

peak_name	the name of the putative pA site (originally from the 4th field in the bed file).
prob_fake_pA	the probability that the putative pA site is false
prob_true_pA	the probability that the putative pA site is true
pred_class	the predicted class of the putative pA site, based on the assignment cutoff. 0 = Falsee/oligo(dT) internally primed, 1 = True
upstream_seq	the upstream sequence of the putative pA site used in the analysis
downstream_seq	the downstream sequence of the putative pA site used in the analysis.

Author(s)

Sarah Sheppard, Haibo Liu, Jianhong Ou, Nathan Lawson, Lihua J. Zhu

References

Sheppard S, Lawson ND, Zhu LJ. Accurate identification of polyadenylation sites from 3' end deep sequencing using a naive Bayes classifier. Bioinformatics. 2013;29(20):2564-2571.

Examples

```
library(BSgenome.Drerio.UCSC.danRer7)
testFile <- system.file("extdata", "test.bed",
                        package = "cleanUpdTSeq")
## convert the test set to GRanges without upstream and downstream sequence
## information
peaks <- BED6WithSeq2GRangesSeq(file = testFile,
                                skip = 1L, withSeq = TRUE)
## build the feature vector for the test set without sequence information
testSet.NaiveBayes = buildFeatureVector(peaks,
                                        genome = Drerio,
                                        upstream = 40L,
                                        downstream = 30L,
                                        wordSize = 6L,
                                        alphabet = c("ACGT"),
                                        sampleType = "unknown",
                                        replaceNAdistance = 30,
                                        method = "NaiveBayes",
                                        fetchSeq = TRUE,
                                        return_sequences = TRUE)
data(data.NaiveBayes)
## sample the test data for code testing, DO NOT do this for real data
samp <- c(1:22, sample(23:4118, 50), 4119, 4120)
Ndata.NaiveBayes <- data.NaiveBayes$Negative[, samp]
Pdata.NaiveBayes <- data.NaiveBayes$Positive[, samp]
testSet.NaiveBayes@data <- testSet.NaiveBayes@data[, samp[-1]-1]

test_out <- predictTestSet(Ndata.NaiveBayes,
                           Pdata.NaiveBayes,
                           testSet.NaiveBayes,
                           outputFile = tempfile(),
                           assignmentCutoff = 0.5)
```

Index