Package 'cleanUpdTSeq'

April 9, 2015

Type Package

Title This package classifies putative polyadenylation sites as true or false/internally oligodT primed.

Version 1.4.0

Date 2013-11-22

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

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

Depends R (>= 2.15), BiocGenerics (>= 0.1.0), BSgenome, BSgenome.Drerio.UCSC.danRer7, GenomicRanges, seqinr, e1071

Description This package uses the Naive Bayes classifier (from e1071) to assign probability values to putative polyadenylation sites (pA sites) based on training data from zebrafish. This will allow the user to separate true, biologically relevant pA sites from false, oligodT primed pA sites.

License GPL-2

biocViews Sequencing, SequenceMatching, Genetics, GeneRegulation

R topics documented:

cleanUpdTSeq-package *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.

Details

Package:	cleanUpdTSeq
Type:	Package
Version:	1.0
Date:	2013-07-22
License:	GPL-2

Author(s)

Sarah Sheppard, Jianhong Ou, Nathan Lawson, Lihua Julie Zhu Maintainer: Sarah Sheppard <Sarah.Sheppard@umassmed.ed
Jianhong Ou <Jianhong.Ou@umassmed.edu>, Lihua Julie Zhu <Julie.Zhu@umassmed.edu>

References

1. Meyer, D., et al., e1071: Misc Functions of the Department of Statistics (e1071), TU Wien. 2012.

2. Pages, H., BSgenome: Infrastructure for Biostrings-based genome data packages.

3. Sarah Sheppard, Nathan D. Lawson, and Lihua Julie Zhu. 2013. Accurate identification of polyadenylation sites from 3' end deep sequencing using a na\"ive Bayes classifier. Bioinformatics. Under revision

Examples

```
#read in a test set
#### first install the package using the following command
#### biocLite("cleanUpdTSeq")
if (interactive())
{
library(cleanUpdTSeq)
testFile = system.file("extdata", "test.bed", package="cleanUpdTSeq")
```

```
testSet = read.table(testFile, sep = "\t", header = TRUE)

#convert the test set to GRanges with upstream and downstream sequence information
peaks = BED2GRangesSeq(testSet,upstream.seq.ind = 7, downstream.seq.ind = 8, withSeq=TRUE)
#build the feature vector for the test set with sequence information
testSet.NaiveBayes = buildFeatureVector(peaks,BSgenomeName = Drerio, upstream = 40,
 downstream = 30, wordSize = 6, alphabet=c("ACGT"),
 sampleType = "unknown",replaceNAdistance = 30,
method = "NaiveBayes", ZeroBasedIndex = 1, fetchSeq = FALSE)

#convert the test set to GRanges without upstream and downstream sequence information
        peaks = BED2GRangesSeq(testSet,withSeq=FALSE)

#build the feature vector for the test set without sequence information
testSet.NaiveBayes = buildFeatureVector(peaks,BSgenomeName = Drerio, upstream = 40,
         downstream = 30, wordSize = 6, alphabet=c("ACGT"),
         sampleType = "unknown",replaceNAdistance = 30,
        method = "NaiveBayes", ZeroBasedIndex = 1, fetchSeq = TRUE)

#predict the test set
data(data.NaiveBayes)
predictTestSet(data.NaiveBayes$Negative, data.NaiveBayes$Positive, testSet.NaiveBayes,
outputFile = "test-predNaiveBayes.tsv", assignmentCutoff = 0.5)
}
```

BED2GRangesSeq *BED2GRangesSeq*

Description

This function converts an object of data.frame from a bed file with sequence information to GRanges with sequence information.

Usage

```
BED2GRangesSeq(data.BED, header = FALSE, upstream.seq.ind = 7, downstream.seq.ind = 8, withSeq)
```

Arguments

data.BED	An object of data.frame from a bed file. The data.frame should at least contains 3 required fields: chrome, chromStart, chromend. The fourth field for "name" is suggested for keeping track of the putative polyadenylation site from the input to the output. The sixth field for "strand" is suggested, as this will affect the classification. For this function, the bed data.frame may also contain two additional fields for the sequence upstream and downstream of the putative pA site. If these are not supplied, the sequence may be obtained when the feature vector is built. Please see http://genome.ucsc.edu/FAQ/FAQformat.html#format1 for more information about the bed file format.

header	header = Boolean TRUE if the first row is the header FALSE if the first row is data
upstream.seq.ind	upstream.seq.ind = int, to delineate the column location containing the sequence upstream of the putative pA site
downstream.seq.ind	downstream.seq.ind = int, to delineate the column location containing the sequence downstream of the putative pA site
withSeq	TRUE indicates that there are upstream and downstream sequences in the file, FALSE indicates that there is no upstream or downstream sequences in the file

Value

Returns object of GRanges

Author(s)

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

Examples

```
testFile <- system.file("extdata", "test.bed", package="cleanUpdTSeq")
testSet <- read.table(testFile, sep = "\t", header = TRUE)
peaks <- BED2GRangesSeq(testSet,withSeq=TRUE)
```

---

buildFeatureVector *buildFeatureVector*

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, BSgenomeName = Drerio, upstream = 50,
downstream = 40, wordSize = 6, alphabet = c("ACGT"),
 sampleType = c("TP", "TN", "unknown"), replaceNAdistance = 40,
 method = c("NaiveBayes", "SVM"), ZeroBasedIndex = 1, fetchSeq = FALSE)
```

Arguments

peaks	An object of GRanges that may contain the upstream and downstream sequence information. This item is created by the function BED2GRangesSeq.
BSgenomeName	Name of the genome to use to get sequence. To find out a list of available genomes, please type available.genomes() in R.
upstream	This is the length of upstream sequence to use in the analysis.

downstream	This is the length of downstream sequence to use in the analysis.
wordSize	This is the size of the word to use as a feature for the upstream sequence. wordSize = 6 should always be used.
alphabet	These are the bases to use, for example DNA bases ACTG.
sampleType	This is the type of sample. For example TP (true positive) or TN (true negative) for training data or unknown for test data.
replaceNAdistance	If there is no A in the downstream sequence, then use this for the average distance of As to the putative polyadenylation site.
method	This is which machine learning method to specify. For this release, method should always be set to NaiveBayes.
ZeroBasedIndex	If the coordinates are set using Zero Based indexing, set this = 1.
fetchSeq	Boolean, for getting upstream and downstream sequence at this step or not.

Value

An object of data frame. Fields include peak name, upstream sequence, downstream sequence, and features to be used in classifying the putative polyadenylation site

Author(s)

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

Examples

```
testFile = system.file("extdata", "test.bed", package="cleanUpdTSeq")
testSet = read.table(testFile, sep = "\t", header = TRUE)

#convert the test set to GRanges with upstream and downstream sequence information
peaks = BED2GRangesSeq(testSet[1:10, ], upstream.seq.ind = 7, downstream.seq.ind = 8, withSeq=TRUE)
#build the feature vector for the test set with sequence information
testSet.NaiveBayes = buildFeatureVector(peaks,BSgenomeName = Drerio, upstream = 40,
   downstream = 30, wordSize = 6, alphabet=c("ACGT"),
   sampleType = "unknown",replaceNAdistance = 30,
   method = "NaiveBayes", ZeroBasedIndex = 1, fetchSeq = FALSE)

#convert the test set to GRanges without upstream and downstream sequence information
peaks = BED2GRangesSeq(testSet[1:10, ],withSeq=FALSE)

#build the feature vector for the test set without sequence information
testSet.NaiveBayes = buildFeatureVector(peaks,BSgenomeName = Drerio, upstream = 40,
        downstream = 30, wordSize = 6, alphabet=c("ACGT"),
        sampleType = "unknown",replaceNAdistance = 30,
        method = "NaiveBayes", ZeroBasedIndex = 1, fetchSeq = TRUE)
```

`data.NaiveBayes` *Training Data*

Description

This is the negative and positive training data.

Usage

```
data(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:

`Negative` 'data.frame': 9219 obs. of 4120 variables:

`Positive` 'data.frame': 22770 obs. of 4120 variables:

Both of them have same structure.

`y` a numeric vector

`n.A.Downstream` a numeric vector

`n.C.Downstream` a numeric vector

`n.T.Downstream` a numeric vector

`n.G.Downstream` a numeric vector

`avg.distanceA2PeakEnd` a numeric vector

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

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

`upstream.seq` a vector of sequence string

`downstream.seq` a vector of sequence string

Examples

```
data(data.NaiveBayes)
head(str(data.NaiveBayes$Negative))
head(str(data.NaiveBayes$Positive))
```

getDownstreamSequence *getDownstreamSequence*

Description

This function gets the sequence upstream of a putative pA site (including the site)

Usage

```
getDownstreamSequence(peaks, downstream = 20, genome)
```

Arguments

peaks	GRanges containing putative pA sites
downstream	downstream = int. This is the length of the sequence to get.
genome	BSgenomeName

Value

Returns an object of GRanges with downtream sequences.

Author(s)

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

Examples

```
testFile <- system.file("extdata", "test.bed", package="cleanUpdTSeq")
testSet <- read.table(testFile, sep="\t", header=TRUE)
peaks <- BED2GRangesSeq(testSet[1:10, ], withSeq=FALSE)
seq = getDownstreamSequence(peaks, downstream=30, genome=Drerio)
```

getUpstreamSequence *Get upstream sequences of the putative pA site*

Description

This function gets the sequence upstream of a putative pA site (including the site)

Usage

```
getUpstreamSequence(peaks, upstream = 40, genome)
```

Arguments

peaks	GRanges containing putative pA sites
upstream	upstream = int. This is the length of the sequence to get.
genome	BSgenomeName

Value

Returns an object of GRanges with uptream sequences.

Author(s)

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

Examples

```
testFile <- system.file("extdata", "test.bed", package="cleanUpdTSeq")
testSet <- read.table(testFile, sep="\t", header=TRUE)
peaks <- BED2GRangesSeq(testSet[1:10, ], withSeq=FALSE)
seq = getUpstreamSequence(peaks, upstream=40, genome=Drerio)
```

---

predictTestSet *predictTestSet*

Description

This function can be used to predict the probabilities for a set of putative pA sites.

Usage

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

Arguments

Ndata.NaiveBayes	This is the negative training data, described further in data.NaiveBayes.
Pdata.NaiveBayes	This is the positive training data, described further in data.NaiveBayes.
testSet.NaiveBayes	This is the test data, a feature vector that has been built for Naive Bayes analysis using the function "buildFeatureVector".
outputFile	This is the name of the file the output will be written to.
assignmentCutoff	This is the cutoff used to assign whether a putative pA is true or false. This can be any floating point number between 0 and 1. For example, assignmentCutoff = 0.5 will assign an putative pA site with prob.1 > 0.5 to the True class (1), and any putative pA site with prob.1 <= 0.5 as False (0).

Value

The output is written to a tab separated file containing fields for peak name, the probability of the putative pA site being false (prob.0), the probability of the putative pA site being true (prob.1), the predicted class (0/False or 1/True) depending on the assignment cutoff, and the upstream and downstream sequence used in assessing the putative pA site.

`PeakName`	This is the name of the putative pA site (originally from the 4th field in the bed file).
`prob False/oligodT internally primed`	This is the probability that the putative pA site is false. Values range from 0-1, with 1 meaning the site is False/oligodT internally primed.
`prob True`	This is the probability that the putative pA site is true. Values range from 0-1, with 1 meaning the site is True.
`pred.class`	This is the predicted class of the putative pA site, based on the assignment cutoff. 0= Falsee/oligodT internally primed, 1 = True
`UpstreamSeq`	This is the upstream sequence of the putative pA site used in the analysis.
`DownstreamSeq`	This is the downstream sequence of the putative pA site used in the analysis.

Author(s)

Sarah Sheppard, Nathan Lawson, Lihua Julie Zhu

References

Sarah Sheppard, Nathan D. Lawson, and Lihua Julie Zhu. 2013. Accurate identification of polyadenylation sites from 3' end deep sequencing using a na\"ive Bayes classifier. Bioinformatics. Under revision

Examples

```
    testFile = system.file("extdata", "test.bed", package="cleanUpdTSeq")
    testSet = read.table(testFile, sep = "\t", header = TRUE)

#convert the test set to GRanges without upstream and downstream sequence information
        peaks = BED2GRangesSeq(testSet,withSeq=FALSE)

#build the feature vector for the test set without sequence information
testSet.NaiveBayes = buildFeatureVector(peaks,BSgenomeName = Drerio, upstream = 40,
         downstream = 30, wordSize = 6, alphabet=c("ACGT"),
         sampleType = "unknown",replaceNAdistance = 30,
        method = "NaiveBayes", ZeroBasedIndex = 1, fetchSeq = TRUE)

    data(data.NaiveBayes)

    ## sample the test data for code testing, DO NOT do this for real data
    ## START SAMPLING
    samp <- c(1:22, sample(23:4119, 50), 4119, 4120)
    Ndata.NaiveBayes <- data.NaiveBayes$Negative[,samp]
    Pdata.NaiveBayes <- data.NaiveBayes$Positive[,samp]
```

```
    testSet.NaiveBayes <- testSet.NaiveBayes[, samp-1]
    ## END SAMPLING

predictTestSet(Ndata.NaiveBayes,
                   Pdata.NaiveBayes,
                   testSet.NaiveBayes,
               outputFile="test-predNaiveBayes.xls",
                   assignmentCutoff = 0.5)
```

Index