Package 'eMIRNA'

February 1, 2019

Type Package

Title eMIRNA: A comprehensive pipeline for discovery and annotation of microRNAs in animal species

Version 1.0

Date 2018-12-26

Author Emilio Marmol-Sanchez

Maintainer Emilio Marmol-Sanchez <emilio.marmol@cragenomica.es>

Description eMIRNA is a comprehensive and user-friendly R-based pipeline for predicting and annotating the presence of known and novel microRNAs in animal species.

License GPL-3

Depends R (>= 3.4.0)

Imports Biobase, LiblineaR, bimba, caret, scales, seqinr, stringr

URL https://github.com/emarmolsanchez/eMIRNA

R topics documented:

`eMIRNA` *eMIRNA: A comprehensive pipeline for discovery and annotation of microRNAs in animal species*

Description

eMIRNA is a comprehensive and user-friendly R-based pipeline for predicting and annotating the presence of known and novel microRNAs in animal species.

`eMIRNA.Features` *eMIRNA.Features*

Description

The third eMIRNA module aims to calculate a series of structural, statistical and sequence-derived features from each sequence that had passed previous filterings, in order to obtain an estimated representation of their structural characteristics.

Usage

```
eMIRNA.Features(file, prefix, rescale)
```

Arguments

`file`	PATH to eMIRNA.Filter.by.Structure Positive or Negative FASTA output files.
`prefix`	String with desired output prefix name.
`rescale`	Boolean TRUE/FALSE for implementing rescaling of calculated features.

Value

Returns a matrix with features representing each of the previously filtered sequences.

`eMIRNA.Filter.by.Size` *eMIRNA.Filter.by.Size*

Description

The first eMIRNA module takes FASTA files and filters sequences contained by a defined length.

Usage

```
eMIRNA.Filter.by.Size(file, prefix, a, b)
```

Arguments

`file`	PATH to Positive (miRNAs) or Negative (Other non-coding) FASTA file.
`prefix`	String with desired output prefix name.
`a`	Lower length filtering threshold.
`b`	Upper length filtering threshold.

Value

Returns a filtered FASTA file with sequences of length fitting the lower and upper defined thresholds.

`eMIRNA.Filter.by.Structure` *eMIRNA.Filter.by.Structure*

Description

The second eMIRNA module aims to estimate the secondary folding structure of selected filtered sequences both in Positive (miRNAs) and Negative (Other non-coding) datasets, thus filtering out all sequences that do not ressemble a pre-miRNA hairpin-like secondary structure.

Usage

```
eMIRNA.Filter.by.Structure(file, prefix)
```

Arguments

`file`	PATH to eMIRNA.Filter.by.Size Positive or Negative FASTA output files.
`prefix`	String with desired output prefix name.

Value

Returns a filtered FASTA file with sequences ressembling a pre-miRNA hairpin-like secondary structure folding.

`eMIRNA.Predict` *eMIRNA.Predict*

Description

The fifth eMIRNA module aims to perform microRNA classification by making use of the previously trained SVM algorithm and query candidate sequence features.

Usage

```
eMIRNA.Predict(model, features, prefix)
```

Arguments

`model`	SVM trained algorithm object.
`features`	Feature matrix representing candidate sequences to evaluate.
`prefix`	String with desired output prefix name.

Value

Returns a list of Sequence candidates names classified as putative pre-miRNAs by the SVM trained algorithm.

eMIRNA.Structural.Pvalues *eMIRNA.Structural.Pvalues*

Description

The last eMIRNA module implements a n-randomization of provided sequences while mantaining k-let counts that aims to analyse if the structural integrity of predicted pre-miRNAs can achieve a stable conformation at a statistically significant level.

Usage

```
eMIRNA.Structural.Pvalues(file, prefix, iterate = 100)
```

Arguments

file	PATH to FASTA file of putative novel miRNA candidates generated by eMIRNA.Refiner_denovo.
prefix	String with desired output prefix name.
iterate	Number of iterations to perform.

Value

Returns a csv file with MFE and EFE P-values for each query pre-miRNA predicted candidate sequence.

eMIRNA.Train *eMIRNA.Train*

Description

The fourth eMIRNA module aims to perform the training process of a Machine Learning based Support Vector Machine (SVM) algorithm, making use of Feature representation previously calculated, to construct a SVM model capable to distinguish between microRNAs and other non-coding sequences.

Usage

```
eMIRNA.Train(pos, neg, imbalance = 'none')
```

Arguments

pos	Positive Features calculated by eMIRNA.Features, saved in R object.
neg	Negative Features calculated by eMIRNA.Features, saved in R object.
imbalance	Imbalance correction algorithm.

Details

Available imbalance correction algorithms are:

- ADASYN: Adaptive Synthetic Sampling (imbalance = 'adasyn')
- BDLSMOTE: borderline-SMOTE1 and borderline-SMOTE2 (imbalance = 'bdlsmote1', imbalance = 'bdlsmote2')
- MWMOTE: Majority Weighted Minority Over-Sampling Technique (imbalance = 'mwmote')
- ROS: Random Over-Sampling (imbalance = 'ros')
- RWO: Random Walk Over-Sampling (imbalance = 'rwo')
- SLSMOTE: Safe-Level-SMOTE (imbalance = 'slsmote')
- SMOTE: Synthetic Minority Over-Sampling Technique (imbalance = 'smote')

Value

Returns a SVM classifier capable to differentiate between microRNAs and other structurally microRNA-like non-coding RNAs. Also returns training and testing tables employed for SVM classifier build.