MiRNAss user guide

Genome-wide pre-miRNA discovery from few labeled examples

Cristian A. Yones (cyones@sinc.unl.edu.ar)[1], Georgina Stegmayer[1], and Diego H. Milone[1]

[1] *Research Institute for Signals, Systems and Computational Intelligence, sinc(i), FICH-UNL, CONICET, Santa Fe, Argentina.*

January 2, 2018

Abstract

MiRNAss is a machine learning method specifically designed for pre-miRNA prediction. It takes advantage of unlabeled sequences to improve the prediction rates even when there are just a few positive examples, and when the negative examples are unreliable or are not good representatives of its class. Furthermore, the method can automatically search for negative examples if the user is unable to provide them. MiRNAss can find a good boundary to divide the pre-miRNAs from other groups of sequences; it automatically optimizes the threshold that defines the classes boundaries, and thus, it is robust to high class imbalance. Each step of the method is scalable and can handle large volumes of data. The last version of the package can be found at CRAN. Also, the development version of the package can be found at: `https://github.com/cyones/miRNAss`. Related projects can be found in `http://fich.unl.edu.ar/sinc/`

1 Input data

MiRNAss receive as input numerical features extracted from hairpin sequences. This means that a genome needs to be pre-proccesed to be able to make predictions with miRNAss. There are two steps: split the genome-wide data in shorter sequences and extract features from this sequences. The first step can be accomplished with *HExtractor* (`https://sourceforge.net/projects/sourcesinc/files/hextractor/`), which is a tool specifically designed for this task. For the feature extraction we have developed a comprehensive tool of feature extraction called *miRNAfe* `http://fich.unl.edu.ar/sinc/blog/web-demo/mirnafe-full/` that is able of calculate almost all the features used in the state-of-the-art prediction methods. For further details see Yones *et. al.*, 2015 [1].

[1] Yones, C. A., Stegmayer, G., Kamenetzky, L., & Milone, D. H. (2015). miRNAfe: A comprehensive tool for feature extraction in microRNA prediction. *Biosystems*, **138**, 1-5.

2 How to use miRNAss

After install the package, load miRNAss with the following command:

```
> library('miRNAss')
```

The following command is the simplest way to execute miRNAss:

```
> miRNAss(features, labels)
```

Where:

- **features**: is a data frame with the features extracted from hairpyn sequences, one sequence per row and one numeric feature per column.
- **labels**: is a numeric vector where the i-th element has a value of 1 if it is a well-known pre-miRNA, a -1 if it is not a pre-miRNA, and zero if it is an unknown sequence that has to be classified (predicted) by the method.

The data provided with the package can be used to test miRNAss. This small dataset is composed of a small set of features extracted from 1000 hairpins randomly extracted from C. elegans hairpins. To use miRNAss with this dataset, first construct the label vector with the CLASS column

```
> y = as.numeric(celegans$CLASS)*2 - 1
```

Remove some labels to make a test

```
> y[sample(which(y > 0),200)] = 0
> y[sample(which(y < 0),700)] = 0
```

Take all the features but remove the label column

```
> x = subset(celegans, select = -CLASS)
```

Call miRNAss with default parameters

```
> p = miRNAss(x,y)
```

To get the indexes of the sequences that were predicted as possible pre-miRNA

```
> is.miRNA = which(p > 0)
```

If the true labels are known, some performance measures can be calculated

```
> SE = mean(p[ celegans$CLASS & y == 0] > 0)
> SP = mean(p[!celegans$CLASS & y == 0] < 0)
> cat('Sensitivity: ', SE, '\nSpecificity: ', SP, '\n')
```

```
Sensitivity:  0.91
Specificity:  0.7471429
```

For more help about all the parameters execute:

```
> help(miRNAss)
```

3 Extra datasets and test scripts

A set of experiments and comparisons with other methods can be done. The scripts and the data of these experiments are contained in the file miRNAss-experiments.zip that can be found in:

`https://sourceforge.net/projects/sourcesinc/files/mirnass/`

To run these tests, after unzip the file, set this directory as the working directory and simply run each script with the function 'source':

```
> setwd('miRNAss-experiments')
> source('2-delta_mirBase.R')
```

This will generate one csv file for each test in the 'results' folder. It is important to point that most of these experiments are computationally expensive and could take quite a while (about 40 minutes for the experiment 2 in an intel i7 PC). You can plot the results executing:

```
> source('plotResults.R')
```

The figures will be saved in the folder 'results'.

4 Software used

- R version 3.4.1 (2017-06-30), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=es_AR.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=es_AR.UTF-8`, `LC_COLLATE=C`, `LC_MONETARY=es_AR.UTF-8`, `LC_MESSAGES=es_AR.UTF-8`, `LC_PAPER=es_AR.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=es_AR.UTF-8`, `LC_IDENTIFICATION=C`
- Running under: `Gentoo/Linux`
- Matrix products: default
- BLAS: `/usr/lib64/blas/reference/libblas.so.0.0.0`
- LAPACK: `/usr/lib64/R/lib/libRlapack.so`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: miRNAss 1.4
- Loaded via a namespace (and not attached): CORElearn 1.51.2, Matrix 1.2-10, RSpectra 0.12-0, Rcpp 0.12.10, cluster 2.0.6, compiler 3.4.1, grid 3.4.1, lattice 0.20-35, nnet 7.3-12, rpart 4.1-11, tools 3.4.1