MiRNA-QC-and-Diagnosis package User Manual

v1.1.2

M. Castelluzzo
A. Perinelli
S. Detassis
M. A. Denti
L. Ricci

May 2021

Contents

1. Overview & License

MiRNA-QC-and-Diagnosis (Micro RNA Quality Control and Diagnosis) is an R package that provides a set of functions to carry out training and classification analyses on datasets containing multiplets of values of micro RNA expressions. The package consists of a set of R functions, each contained in a dedicated source file located in `/R/`.

DISCLAIMER

The software in this package is for general research purposes only and is thus provided WITHOUT ANY WARRANTY. It is NOT intended to form the basis of clinical decisions. Please refer to the **GNU General Public License (GPL) v3** for further information.

Reference

The analysis algorithm implemented in this package was originally published in

L. Ricci, V. Del Vescovo, C. Cantaloni, M. Grasso, M. Barbareschi and M. A. Denti, *Statistical analysis of a Bayesian classifier based on the expression of miRNAs*, BMC Bioinformatics **16**:287 (2015).
DOI: 10.1186/s12859-015-0715-9

The paper is henceforth referred to as **"main reference"**.
The present software package is described in

M. Castelluzzo, A. Perinelli, S. Detassis, M. A. Denti and L. Ricci, *MiRNA-QC-and-Diagnosis: An R package for diagnosis based on MiRNA expression*, SoftwareX **12**:100569 (2020).
DOI: 10.1016/j.softx.2020.100569

Please cite both these references in works that use the present package.
Bibliography entries can be displayed from within R by typing

```
        citation("MiRNAQCD")
        # or, to get BibTeX items,
        toBibtex(citation("MiRNAQCD"))
```

Licensing

This package, all the included source code, documentation and examples are released under the **GNU General Public License (GPL) v3**. A copy of the license is provided in the package root folder.
The R logo is licensed under the CC-BY-SA 4.0 license.

Package authors

Michele Castelluzzo
Department of Physics, University of Trento, 38123 Trento, Italy
michele.castelluzzo@unitn.it

Alessio Perinelli
CIMeC, Center for Mind/Brain Sciences, University of Trento, 38068, Rovereto, Italy
alessio.perinelli@unitn.it

Simone Detassis
Department of Cellular, Computational and Integrative Biology (CIBIO), University of Trento, 38123 Trento, Italy
simone.detassis@unitn.it

Michela Alessandra Denti
Department of Cellular, Computational and Integrative Biology (CIBIO), University of Trento, 38123 Trento, Italy
michela.denti@unitn.it

Leonardo Ricci
Department of Physics, University of Trento, 38123 Trento, Italy
CIMeC, Center for Mind/Brain Sciences, University of Trento, 38068, Rovereto, Italy
leonardo.ricci@unitn.it
Nonlinear Systems & Electronics Lab website: nse.physics.unitn.it

2. Download & setup

The package requires R version 3.2 or later. The package depends on the packages `stats`, `utils`, `tools`, `pROC`, and `ggplot2` (the latter for plotting purposes). Once installed, the package is available within R with the name `MiRNAQCD` and is therefore imported by typing

```
library(MiRNAQCD)
```

Download

The package is available on CRAN at the address https://CRAN.R-project.org/package=MiRNAQCD. The development version of the package (which is usually a few steps ahead of the CRAN release) can be downloaded at https://github.com/LeonardoRicci/MiRNA-QC-and-Diagnosis. Setup instructions can be found within the `/setup/` directory.

Manual setup

After downloading the package, setup can be carried out as follows. The straightforward way to install the package is to open a terminal in the folder where the archive file is stored and type

```
R CMD INSTALL MiRNAQCD_1.1.2.tar.gz
```

Alternatively, launch an R console and type

```
install.packages("MiRNAQCD_1.1.2.tar.gz", type="source")
```

Please note that permissions to write on the target library directory are required.

If the package cannot be installed from the `.tar.gz` archive, the latter can be rebuilt from source. This procedure requires the DEVTOOLS package to be installed in order to produce the function documentation. Open R from within the package root directory and type

```
library(devtools)
document()
build()
```

A `.tar.gz` file will be produced in the parent directory. Then, the package can be installed through the `install.packages()` function as described above.

For further information on how to install packages please refer to the official CRAN manual page.

3. Citing the package

The primary work concerning the method implemented in the package is

L. Ricci, V. Del Vescovo, C. Cantaloni, M. Grasso, M. Barbareschi and M. A. Denti, *Statistical analysis of a Bayesian classifier based on the expression of miRNAs*, BMC Bioinformatics **16**:287 (2015).
DOI: 10.1186/s12859-015-0715-9

The publication describing the software package is

M. Castelluzzo, A. Perinelli, S. Detassis, M. A. Denti and L. Ricci, *MiRNA-QC-and-Diagnosis: An R package for diagnosis based on MiRNA expression*, SoftwareX **12**:100569 (2020).
DOI: 10.1016/j.softx.2020.100569

Please cite both these references in works that use the present package.
BibTeX entries for these references are

```
@Article{RicciEtAl2015,
        title = {Statistical analysis of a Bayesian classifier based on the
            expression of miRNAs},
        author = {Leonardo Ricci and Valerio {Del Vescovo} and Chiara Cantaloni and
            Margherita Grasso and Mattia Barbareschi and Michela Alessandra Denti},
        journal = {BMC Bioinformatics},
        year = {2015},
        volume = {16},
        pages = {287},
        doi = {10.1186/s12859-015-0715-9},
}
```

```
@Article{CastelluzzoEtAl2020,
        title = {MiRNA-QC-and-Diagnosis: An R package for diagnosis based on MiRNA
            expression},
        author = {Michele Castelluzzo and Alessio Perinelli and Simone Detassis and
            Michela Alessandra Denti},
        journal = {SoftwareX},
        year = {2020},
        volume = {12},
        pages = {100569},
        doi = {10.1016/j.softx.2020.100569},
}
```

4. Package structure

4.1 Function list

The MiRNA-QC-and-Diagnosis package is entirely written in R . All source files (`*.R`) are located within `/R/`. Each source file defines a single function. A list of the functions is provided in the following table; clicking on a link leads to the corresponding documentation in Part 5, where arguments, options and returned values are reported. Functions devoted to the generation of graphical output (and internally called by some of the functions listed in the table) are not exported to the package namespace.

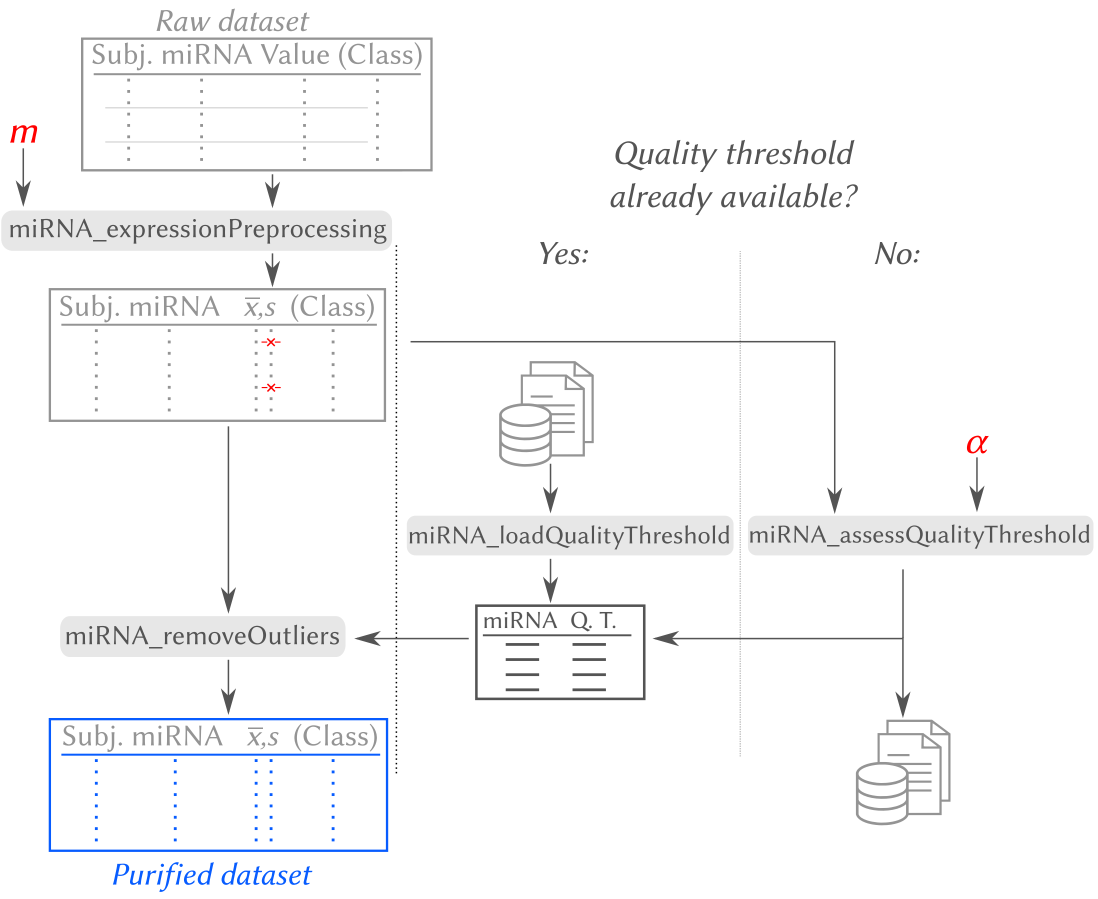
`miRNA_expressionPreprocessing` ↗	Preprocesses a dataset
`miRNA_assessQualityThreshold` ↗	Assesses, for the purpose of outlier identification, quality threshold values
`miRNA_removeOutliers` ↗	Removes outliers from a dataset according to a set of quality threshold values
`miRNA_loadQualityThreshold` ↗	Loads from file the outcomes of `miRNA_assessQualityThreshold`
`miRNA_classifierSetup` ↗	Trains the Bayesian classifier on a training dataset; alternatively, carries out a statistical analysis of miRNA (means and variances, normality, cross-correlation)
`miRNA_diagnosis` ↗	Classifies a datasets by assigning a diagnosis according to a trained classifier
`miRNA_loadDiagnosticThreshold` ↗	Loads from file the outcomes of `miRNA_classifierSetup`

4.2 Quality control

The functions `miRNA_expressionPreprocessing`, `miRNA_assessQualityThreshold` and `miRNA_removeOutliers`, as well as the load utility `miRNA_loadQualityThreshold`, provide the necessary tools to preprocess and clean datasets. The workflow concerning this first analysis stage is schematically shown in Fig. 4.1 and is summarized as follows.

1. Input datasets have to be preprocessed before any use. The `miRNA_expressionPreprocessing` function transforms a “raw” dataset into a format that is suited to be used by any other function of the package. Besides properly checking and formatting the data, the function computes the mean and standard deviation of every multiplet in the dataset.

2. In order to properly remove outliers, quality threshold values – i.e. critical standard deviation values – have to be assessed. The `miRNA_assessQualityThreshold` function carries out this task. Alternatively, a set of quality threshold values that were previously assessed on another dataset can be loaded from a file through the `miRNA_loadQualityThreshold` function.



Raw dataset
Subj. miRNA Value (Class)
m
miRNA_expressionPreprocessing
Subj. miRNA $\bar{x}$,s (Class)
Quality threshold already available?
Yes:
No:
α
miRNA_loadQualityThreshold
miRNA_assessQualityThreshold
miRNA Q. T.
miRNA_removeOutliers
Subj. miRNA $\bar{x}$,s (Class)
Purified dataset

Figure 4.1: Workflow of the quality control stage, consisting of preprocessing and outlier removal functions.

3. Given a preprocessed dataset and a set of quality threshold values, the `miRNA_removeOutliers` function removes the outliers from the dataset and produces a purified dataset to be used in the subsequent analysis stages.

4.3 Classifier setup and diagnosis

The functions `miRNA_classifierSetup` and `miRNA_diagnosis`, as well as the load utility `miRNA_loadDiagnosticThreshold`, provide the necessary tools to train a Bayesian classifier and use it for dataset classification, i.e. diagnosis. In addition, the `miRNA_classifierSetup` function also allows to assess statistical properties of miRNAs. The workflow concerning this analysis stage is schematically shown in Fig. 4.2 and is summarized as follows. Notice that both training and diagnosis require a preprocessed and purified dataset as described in Sec. 4.2.

1. A Bayesian classifier relies on a *sample score*, i.e. a linear combination of miRNA expression values, and a threshold that discriminates sample scores into two possible classification outcomes, or diagnosis, henceforth referred to as "*Target*" and "*Versus*". The optimal threshold value has to be assessed by training the Bayesian classifier on a dataset whose entries have been already classified by using an alternative method. Training can be performed on a dataset by means of the `miRNA_classifierSetup` function, which returns diagnostic threshold values to be used for classification. Methodological details on the Bayesian classifier can be found in the *main reference* . Alternatively, a set of threshold values that were previously determined on another dataset can be loaded from a file through the `miRNA_loadDiagnosticThreshold` function.

2. In addition to training capability, the `miRNA_classifierSetup` function allows to analyze statistical

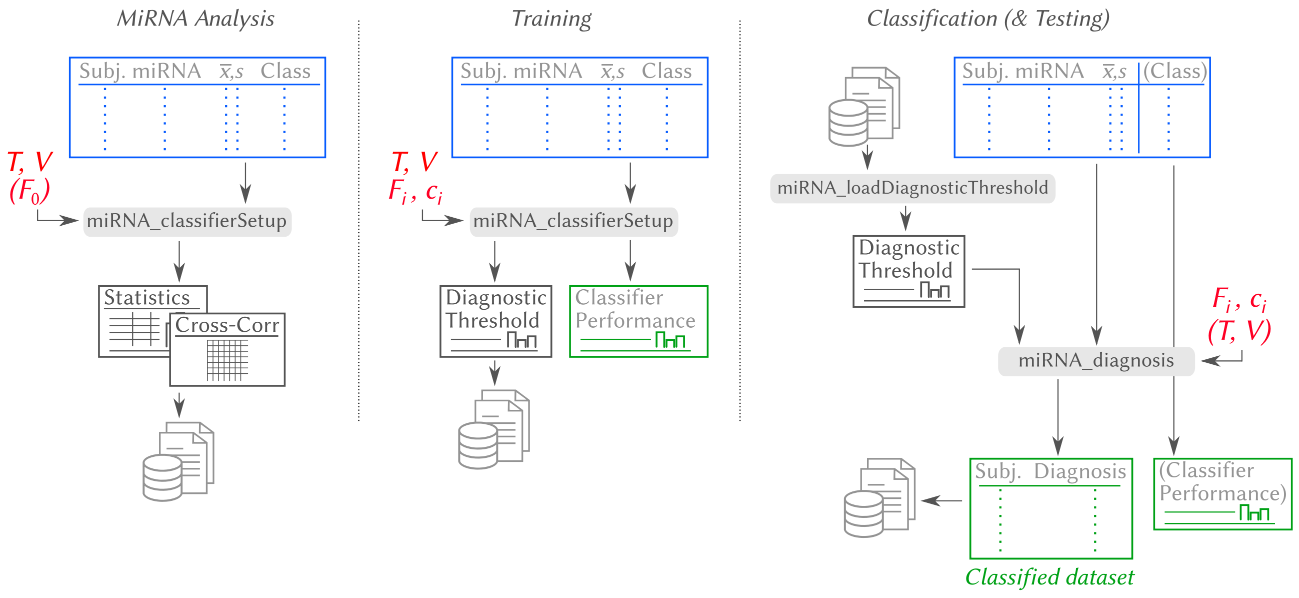


Figure 4.2: Workflow of the analysis, classifier setup and diagnosis stages.

properties of miRNA such as the separation between *Target* and *Versus* distributions, the outcomes of normality test, and the cross correlation between each pair of miRNAs.

3. Given a preprocessed and purified dataset and given a set of diagnostic threshold values corresponding to a trained classifier, the `miRNA_diagnosis` carries out the classification of the dataset entries (diagnosis) by assigning each of them either to the *Target* or the *Versus* set. Beside classifying the input dataset, this function can also assess the performance of a classifier on datasets for which an *a priori* (independent) classification is already available.

5. Detailed functions documentation

A brief documentation about each function can be evoked from within the R environment by typing

```
        help(<function name>)
        # e.g.
        help(miRNA_expressionPreprocessing)
```

In the following, the detailed function documentation is reported. Functions are grouped as in Part 4. The mathematical arguments underlying the package are discussed in detail in the *main reference* .

5.1 Quality control functions: preprocessing and dataset cleaning

Preprocess – *Preprocesses a dataset containing multiplets of miRNA expression.*

```
preprocDataset <- miRNA_expressionPreprocessing(inputDataset, multipletSize)
```

Arguments:

`inputDataset`	**Data frame** containing the input dataset. Three columns, labelled with '`Subject`', '`miRNA`', '`Value`', must be present: if any of these is missing, the function exits. For each miRNA and subject, an n-tuplet of values can be present. The column labelled with '`Class`', if present, is also echoed in the output frame. Any other column is discarded.
`multipletSize`	**Integer** number corresponding to the size of multiplets to be used. If missing or invalid (e.g. $\leqslant 0$), the function exits. Multiplets having different size are discarded.

Returned value:

`preprocDataset`	**Data frame** containing the preprocessed dataset. The data frame contains the columns '`Subject`', '`miRNA`', '`Mean`', '`StdDev`', '`SampleSize`' and, if present in the input data frame, '`Class`'.

Assess quality threshold – *Assesses, for the purpose of outlier identification, quality threshold values.*

```
qualityThresholdValues <- miRNA_assessQualityThreshold(inputDataset,
                                                       significanceLevel=0.05,
                                                       saveOutputFile=FALSE,
                                                       outputFileName="",
                                                       sep="\t")
```

Arguments:

`inputDataset`	**Data frame** containing the preprocessed dataset upon which the critical sigma values have to be assessed. Five columns, labelled with '`Subject`', '`miRNA`', '`Mean`', '`StdDev`', '`SampleSize`', must be present: if any of these is missing, the function exits. Any other column, inclusively, if any, '`Class`', is ignored.
`significanceLevel`	**Floating-point** number corresponding to the significance level to be used. Default is 0.05, i.e. 5%.
`saveOutputFile`	**Boolean** value specifying whether results have to be stored into a file (`TRUE`) or not (`FALSE`). Default is `FALSE`.
`outputFileName`	**String** specifying the name (and path) of the file where quality threshold values have to be saved. If not assigned, a suitable filename is automatically generated.
`sep`	**Character** specifying the column separator for the output file. The default is tabulator (`\t`).

Returned value:

`qualityThresholdValues`	**Data frame** containing two columns: the first one, labelled with '`miRNA`', for the miRNA names; the second one, labelled with '`QualityThreshold`', for the related quality threshold values.

If `saveOutputFile` is set to `TRUE`, the function produces the following data file (plain text):

- A file containing the same data as the data frame returned by the function.

Load quality threshold values – *Loads from file the outcomes of* `miRNA_assessQualityThreshold`.

```
qualityThresholdValues <- miRNA_loadQualityThreshold(inputFileName, sep="")
```

Arguments:

`inputFileName`	**String** specifying the name (and path) of the file. Two columns, labelled with '`miRNA`' and '`QualityThreshold`', must be present in the file: if any of these is missing, the function exits. Any other column is ignored.
`sep`	**Character** specifying the column separator. If none or empty (`""`) is specified, any space or tabulator is assumed to be a separator.

Returned value:

`qualityThresholdValues`	**Data frame** containing two columns: the first one, labelled with '`miRNA`', for the miRNA names; the second one, labelled with '`QualityThreshold`', for the related quality threshold values.

Remove outliers – *Removes outliers from a dataset according to a set of quality threshold values.*

```
cleanedDataset <- miRNA_removeOutliers(inputDataset, qualityThresholdValues)
```

Arguments:

`inputDataset`	**Data frame** containing the preprocessed dataset that has to be purified from outliers. Five columns, labelled with ‘`Subject`’, ‘`miRNA`’, ‘`Mean`’, ‘`StdDev`’, ‘`SampleSize`’, must be present: if any of these is missing, the function exits. The column labelled with ‘`Class`’, if present, is also echoed in the output frame. Any other column is discarded.
`qualityThresholdValues`	**Data frame** containing the assessed quality threshold values to be used for outlier removal. Two columns, labelled with ‘`miRNA`’ and ‘`QualityThreshold`’, must be present: if any of these is missing, the function exits. Any other column is ignored.

Returned value:

`cleanedDataset`	**Data frame** containing the purified dataset. The data frame contains the columns ‘`Subject`’, ‘`miRNA`’, ‘`Mean`’, ‘`StdDev`’, ‘`SampleSize`’ and, if present in the input data frame, ‘`Class`’.

5.2 Classifier setup and diagnosis functions

Bayesian classifier training / miRNA analysis – *Trains the Bayesian classifier on a training dataset; alternatively, carries out a statistical analysis of miRNA expressions (means and variances, normality, cross-correlation).*

```
outputFrame <- miRNA_classifierSetup(inputDataset,
                                      inputTargetList, inputVersusList="",
                                      inputMiRNAList="", coeffList="",
                                      saveOutputFile=FALSE,
                                      outputFileBasename="",
                                      sep="\t", plotFormat="pdf",
                                      histogramParameters="")
```

This function carries out two different tasks and thus can run in two different modes, henceforth referred to as "*Analysis mode*" and "*Training mode*".

- *Analysis mode*: carries out a statistical analysis of miRNA expressions (means and variances, normality, cross-correlation), either directly or upon subtraction of the expression of a "normalizer".
- *Training mode*: trains a Bayesian classifier by assessing the corresponding optimal diagnostic threshold value and its uncertainty.

In order to select between *Analysis mode* and *Training mode*, the input parameters `inputMiRNAList` and `coeffList` have to comply with the requirements described below.

- *Analysis mode*: `coeffList` has zero length (default setting); `inputMiRNAList` can either have zero length (default setting) or unitary length; in the latter case, the single entry of `inputMiRNAList` is assumed to be the normalizer.
- *Training mode*: `inputMiRNAList` and `coeffList` have the same nonzero length.

In the case of any other setting of `inputMiRNAList` and `coeffList`, the function exits.

Arguments:

`inputDataset`	**Data frame** containing the preprocessed and purified dataset to analyze or to train on. Six columns, labelled with '`Subject`', '`miRNA`', '`Mean`', '`StdDev`', '`SampleSize`' and '`Class`', must be present: if any of these is missing, the function exits. Any other column is ignored.
`inputTargetList`	**List** of classes, specified by means of, for example, `c("A", "B", ...)`, to be used as *Target* classification set.
`inputVersusList`	**List** of classes, specified by means of, for example, `c("C", "D", ...)`, to be used as *Versus* classification set. If no list is entered (default setting), the *Versus* classification set is taken as the complement of the *Target* classification set within `inputDataset`.

`inputMiRNAList`	**List** of miRNAs, specified by means of, for example, `c("F1", "F2", ...)`, to be used by the classifier in *Training mode*. Subjects for which any of the chosen miRNA is absent are discarded. In *Analysis mode* two possibilities are given: if no normalizer is used, the `inputMiRNAList` argument is omitted; alternatively, the `inputMiRNAList` argument has to correspond to the label of the miRNA used as a normalizer.
`coeffList`	**List** of coefficients, specified by means of, for example, `c(1.0, 0.5, ...)`, to be used by the classifier in *Training mode*. The number and the order of the coefficients have to be the same as the number and order of miRNAs. In *Analysis mode*, the `coeffList` argument is omitted.
`saveOutputFile`	**Boolean** value specifying whether results have to be stored into files (`TRUE`) or not (`FALSE`). Default is `FALSE`.
`outputFileBasename`	**String** specifying the name and path, without extension, of the files where the results are stored. If not assigned, a filename is automatically generated.
`sep`	**Character** specifying the column separator for the output file. The default is tabulator (`\t`).
`plotFormat`	**String** specifying the format of graphic files. Possible choices are "`pdf`" (default) or "`png`".
`scorePlotAscending`	**Boolean** value setting the direction in which samples are ordered: `TRUE` corresponds to samples ordered by ascending Score, `FALSE` corresponds to samples ordered by descending Score. Default is `TRUE`. This argument is meaningful only if `saveOutputFile` is set to `TRUE`.
`scoreplotParameters`	**String** specifying the parameters used to configure the y axis of the scores diagram (see below). If empty, the axis is configured by assessing suitable parameters from the data. This argument is meaningful only if `saveOutputFile` is set to `TRUE` and if the function is running in *Training mode*. The string has to comply with the format `"yl_yu_yt"`, where: `yl` is the lower y limit; `yu` is the upper y limit; `yt` is the y tics interval. An example is `"-4.0_4.0_0.5"` for y tics covering the $[-4, 4]$ interval in steps of 0.5.
`histogramParameters`	**String** specifying the parameters used to build histograms (see below). If empty, histograms are built by assessing suitable parameters from the data. This argument is meaningful only if `saveOutputFile` is set to `TRUE`. The string has to comply with the format `"xl_xu_bw"`, where: `xl` is the lower boundary of the leftmost bin; `xu` is the upper boundary of the rightmost bin; `bw` is the bin width. An example is `"0.0_10.0_1.0"` for bins covering the $[0, 10]$ interval and having width 1.0.
`colorComplementFlag`	**Boolean** value to switch between the default palette (`FALSE`) and its inverted version (`TRUE`). Default is `FALSE`, corresponding to Target samples being reported in blue and Versus samples being reported in red. This argument is meaningful only if `saveOutputFile` is set to `TRUE`.

Returned value:

In *Training mode*, if `saveOutputFile` is set to `TRUE`, the function produces the following files:

- **Subjects' sample score and classification** file (`<outputFileBasename>.dat`) containing, for each subject, the sample score and the *a priori* diagnosis (i.e. the `'Class'`).

- **Output parameters** file (`<outputFileBasename>.txt`) containing the same data as the `outputFrame`. In addition, the confusion matrix is saved to a secondary file, `<outputFileBasename>_confusion_matrix.txt`.

- **Histograms** (`<outputFileBasename>_histogram.<plotFormat>`) of the classifier scores both in

`outputFrame`	In *Training mode*: **Data frame** containing the following columns: `‘Threshold’`, with the diagnostic threshold value assessed by the training procedure; `‘DeltaThreshold’`, with the corresponding uncertainty; `‘ChiUp’` and `‘DChiUp’`, with the threshold and the corresponding uncertainty, respectively, for the odds 90:10; `‘ChiDown’`, `‘DChiDown’`, with the threshold and the corresponding uncertainty, respectively, for the odds 10:90; `’Accuracy’`, i.e. rate of correct responses; `’Specificity’`; `’Sensitivity’`; `’F1-score’`; `‘DPrime’`, with the normalized difference between the *Target* and *Versus* sample means, namely $(\bar{x}_T - \bar{x}_V)/\sqrt{(s_T^2 + s_V^2)/2}$; area under curve (AUC) of the ROC plot (see below), and related uncertainty.
`outputFrame`	In *Analysis mode*: **Data frame** containing the statistics assessed by analyzing each miRNA. The data frame contains the columns `‘miRNA’`, `‘Classification’`, `‘NumberOfSubjects’`, `‘Mean’`, `‘StdDev’`, `‘NormalityTest’` (p-value of Shapiro-Wilk test of normality), `‘t-test’` (p-value of Student’s t-test to check the null hypothesis that the *Target* and *Versus* sets have the same population mean).

the case of the *Target* and the *Versus* set.

- **Score diagram** (`<outputFileBasename>_score.<plotFormat>`) of the sample scores highlighting the *Target* and *Versus* diagnosis.

- **ROC curve** (Receiver Operating Characteristic) plot (`<outputFileBasename>_ROC.<plotFormat>`). The area under the curve is reported in the **Output parameters** file.

In *Analysis mode*, if `saveOutputFile` is set to `TRUE`, the function produces the following files:

- **Output parameters** file (`<outputFileBasename>.txt`) containing the same data as the `outputFrame` and, in addition, the correlation matrix between miRNA expressions and the related p value matrix computed on three different sets of subjects: subjects whose classification belongs to the *Target* set, subjects whose classification belongs to the *Versus* set, subjects whose classification belongs to the union of the *Target* and the *Versus* set. In this last case, a matrix of coefficients “epsilon” that, according to the standard deviation and correlation analysis, maximize the classifier’s performance (see Eq. (9) of the *main reference*) is also reported. The epsilon matrix should be interpreted in the following way. Each row corresponds to a single miRNA: given a unitary coefficient to that miRNA, the row provides the coefficients to be used for the other miRNAs in order to maximize performance. Please note that the matrix is not symmetric.

- **Histograms** (`<outputFileBasename>_[<feature>_]histogram.<plotFormat>`), for each analyzed miRNA, of the feature values both in the case of the *Target* and the *Versus* set.

Load diagnostic threshold values – *Loads from file the outcomes of* `miRNA_classifierSetup`.

```
thresholdFrame <- miRNA_loadDiagnosticThreshold(inputFileName, sep="")
```

Arguments:

`inputFileName`	**String** specifying the name (and path) of the file. Six columns, labelled with '`Threshold`', '`DeltaThreshold`', '`ChiUp`', '`DChiUp`', '`ChiDown`', '`DChiDown`', must be present in the file: if any of these is missing, the function exits.
`sep`	**Character** specifying the column separator. If none or empty (`""`) is specified, any space or tabulator is assumed to be a separator.

Returned value:

`thresholdFrame`	**Data frame** containing six columns: '`Threshold`', '`DeltaThreshold`', '`ChiUp`', '`DChiUp`', '`ChiDown`', '`DChiDown`'.

Classify dataset – *Classifies a datasets – i.e. assigns a diagnosis to each subject – according to a trained classifier.*

```
classifiedDataset <- miRNA_diagnosis(inputDataset, inputMiRNAList,
                                     coeffList, inputThreshold,
                                     inputTargetList="", inputVersusList="",
                                     saveOutputFile=FALSE, outputFileBasename="",
                                     sep="\t", plotFormat="pdf",
                                     histogramParameters="")
```

Beside classifying the input dataset, this function can also run a *"Performance analysis mode"* through which the performance of a classifier are assessed. Clearly, this assessment requires a dataset for which an *a priori* (independent) classification is already available. To enable the *Performance analysis mode*, input parameters have to comply with the following requirements.

- The data frame `inputDataset` has to contain a '`Class`' column.
- The argument `inputTargetList` has to be nonempty, and has to provide the list of classes to be considered *a priori* as belonging to the *Target* set. The list corresponding to the *a priori Versus* set can be provided in `inputVersusList`. If this argument is left empty, the *a priori Versus* set is taken as the complement of the *a priori Target* set within `inputDataset`.

Arguments:

`inputDataset`	**Data frame** containing the preprocessed and purified dataset to be classified. Five columns, labelled with '`Subject`', '`miRNA`', '`Mean`', '`StdDev`', '`SampleSize`', must be present: if any of these is missing, the function exits. Any other column is ignored. If the *Performance analysis mode* is selected (see `inputTargetList`), the dataset has to contain the '`Class`' column as well.
`inputMiRNAList`	**List** of miRNAs, specified by means of, for example, `c("F1", "F2", ...)`, to be used by the classifier. Subjects for which any of the chosen miRNAs is absent are discarded.
`coeffList`	**List** of coefficients, specified by means of, for example, `c(1.0, 0.5, ...)`, to be used by the classifier. The number and the order of the coefficients have to be the same as the number and order of miRNAs.
`inputThreshold`	**Data frame** containing the results of a classifier training, e.g. the `outputFrame` returned by `miRNA_classifierSetup` in *Training mode* or loaded by the function `miRNA_loadDiagnosticThreshold`. The column '`Threshold`' has to be present.
`inputTargetList`	**List** of classes, specified by means of, for example, `c("A", "B", ...)`, to be used as *a priori Target* set. Providing this argument corresponds to request the *Performance analysis mode*. Consequently, `inputDataset` is expected to contain the '`Class`' column as well.
`inputVersusList`	(Used only if *Performance analysis mode* is requested). **List** of classes, specified by means of, for example, `c("C", "D", ...)`, to be used as *a priori Versus* set. If no list is entered (default setting), the *a priori Versus* set is taken as the complement of the *Target* classification set within `inputDataset`.
`saveOutputFile`	**Boolean** value specifying whether results have to be stored into files (`TRUE`) or not (`FALSE`). Default is `FALSE`.

`outputFileBasename`	**String** specifying the name and path, without extension, of the files where the results are stored. If not assigned, a filename is automatically generated.
`sep`	**Character** specifying the column separator for the output file. The default is tabulator (`\t`).
`fileFormat`	**String** specifying the format of graphic files. Possible choices are “`pdf`” (default) or “`png`”.
`scorePlotAscending`	**Boolean** value setting the direction in which samples are ordered: `TRUE` corresponds to samples ordered by ascending Score, `FALSE` corresponds to samples ordered by descending Score. Default is `TRUE`. This argument is meaningful only if `saveOutputFile` is set to `TRUE`.
`scorePlotParameters`	**String** specifying the parameters used to configure the y axis of the scores diagram (see below). If empty, the axis is configured by assessing suitable parameters from the data. This argument is meaningful only if `saveOutputFile` is set to `TRUE`. The string has to comply with the format `"yl_yu_yt"`, where: `yl` is the lower y limit; `yu` is the upper y limit; `yt` is the y tics interval. An example is `"-4.0_4.0_0.5"` for y tics covering the $[-4, 4]$ interval in steps of 0.5.
`histogramParameters`	(Used only if *Performance analysis mode* is requested). **String** specifying the parameters used to build the histogram (see below). If empty, the histogram is built by assessing suitable parameters from the data. This argument is meaningful only if `saveOutputFile` is set to `TRUE`. The string has to comply with the format `"xl_xu_bw"`, where: `xl` is the lower boundary of the leftmost bin; `xu` is the upper boundary of the rightmost bin; `bw` is the bin width. An example is `"0.0_10.0_1.0"` for bins covering the $[0, 10]$ interval and having width 1.0.
`colorComplementFlag`	**Boolean** value to switch between the default palette (`FALSE`) and its inverted version (`TRUE`). Default is `FALSE`, corresponding to Target samples being reported in blue and Versus samples being reported in red. This argument is meaningful only if `saveOutputFile` is set to `TRUE`.

Returned value:

`classifiedDataset`	**Data frame** containing the classified dataset. The data frame contains the columns ‘`Subject`’, ‘`Diagnosis`’, ‘`Score`’.

If `saveOutputFile` is set to `TRUE`, the function produces the following files:

- **Subjects’ sample score and classification** file (`<outputFileBasename>.dat`) containing, for each subject, the sample score and the (*a posteriori*) diagnosis.
- **Score diagram** (`<outputFileBasename>_score.<plotFormat>`) of the sample scores highlighting the *Target* and *Versus* diagnosis.
- (If *Performance analysis mode* is active) **Performance metrics** file (`<outputFileBasename>_performance.txt`) containing: confusion matrix; accuracy; specificity; sensitivity; F1-score; separation d', normalized by the standard deviation, between score distributions of the Target and the Versus set; area under curve (AUC) of the ROC plot (see below), and related uncertainty.
- (If *Performance analysis mode* is active) **Histograms** (`<outputFileBasename>_histogram.<plotFormat>`) of the classifier scores both in the case of the *Target* and the *Versus* set.
- (If *Performance analysis mode* is active) **ROC curve** (Receiver Operating Characteristic) plot (`<outputFileBasename>_ROC.<plotFormat>`). The area under the curve is reported in the **Performance metrics** file.

6. Examples

The following examples show how to preprocess and analyze datasets by means of the MiRNA-QC-and-Diagnosis functions. Scripts containing all instructions reported here can be found in /examples/ within the package directory tree, or within /path-to-library/MiRNAQCD/extdata/ after the package is installed[1].

Example on synthetic datasets

This example relies on two synthetic datasets, corresponding to the files synthetic_dataset_alpha.dat and synthetic_dataset_beta.dat. The analysis pipeline concerning this dataset makes up the R script example_synthetic_dataset.R, which contains all instructions reported here.

Data format, loading and preprocessing

The raw dataset synthetic_dataset_alpha.dat is analyzed first. Opening the file with a text editor reveals the following data lines:

```
Subject miRNA   Value   Class
1       FX      17.9494 A
1       FX      18.1404 A
1       FX      17.8583 A
1       FY      14.1306 A
...
```

The dataset contains 120 subjects (numbered 1-120) and three phony miRNAs (FX, FY, FZ). In this case the Class column is present: the dataset is thus suited to be used as a training one.

The package does not provide any function to load raw datasets; a simple read.table does the job:

```
datasetAlpha <- read.table(file="dataset_alpha.dat", header=TRUE)
```

In datasetAlpha, the size of multiplets is 3. The miRNA_expressionPreprocessing function is called as

```
preprocessedDatasetAlpha <- miRNA_expressionPreprocessing(datasetAlpha, multipletSize
   =3)
```

The output data frame processedDatasetAlpha looks like this:

```
Subject miRNA   Mean    StdDev  SampleSize      Class
1       FX      17.9827 0.1440  3               A
1       FY      14.0644 0.0993  3               A
1       FZ      11.8195 0.1243  3               A
10      FX      7.6036  0.3606  3               B
...
```

The preprocessed data frame processedDatasetAlpha does not contain an entry for subject 120, which was discarded due to its miRNA expressions being reported in doublets, instead of triplets.

[1]"/path-to-library" can be shown from within R by means of the .libPaths() command. A possible output on a Linux system is /usr/lib/R/library.

Quality control: outlier removal

In order to remove outliers from a dataset, the quality threshold values that discriminate outliers have to be assessed. Upon setting, for example, the significance level to 5%, the quality threshold values are assessed by means of the following function call:

```
qualityThresholdAlpha <- miRNA_assessQualityThreshold(preprocessedDatasetAlpha,
    significanceLevel=0.05, saveOutputFile=TRUE, outputFileName="qualityThreshold_
    datasetAlpha.dat")
```

The resulting qualityThresholdAlpha data frame contains a list of miRNAs and the related quality threshold values:

```
miRNA   QualityThreshold
FX      0.51
FY      0.5
FZ      0.53
```

The miRNA_assessQualityThreshold function also saves a copy of qualityThresholdAlpha on the file qualityThreshold_datasetAlpha.dat, within the current working directory. The file will be used in later examples.

The quality threshold values stored in qualityThresholdAlpha can now be used to purify a dataset from outliers by calling

```
purifiedDatasetAlpha <- miRNA_removeOutliers(preprocessedDatasetAlpha,
    qualityThresholdAlpha)
```

As a consequence of the outliers that have been discarded, the purifiedDatasetAlpha data frame (345 entries) turns out to be smaller than preprocessedDatasetAlpha (357 entries).

Features analysis

The purifiedDatasetAlpha data frame is ready to be used to train a classifier. The Target class is "A", while the Versus classes are "B" and "C":

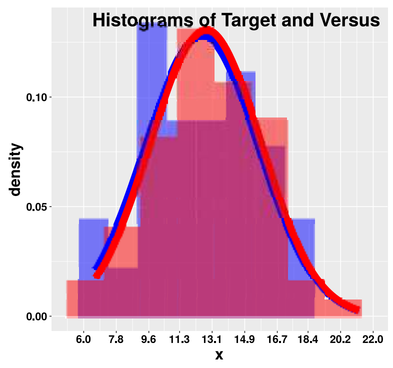
```
Target <- c("A")
Versus <- c("B", "C")
```

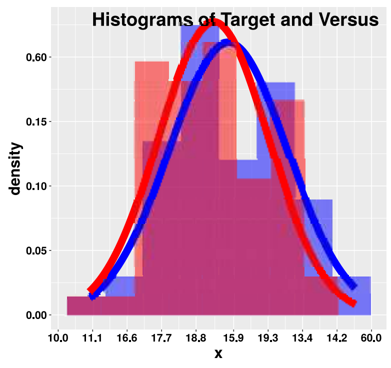
In this case the coefficients of the linear combination of miRNA expressions to be used as classifier are *a priori* unknown. In order to determine them, the miRNA_classifierSetup function is first used in *"Analysis mode"*. By analyzing each miRNA separately, the function provides insight on which miRNAs are better suited to discriminate between Target and Versus.

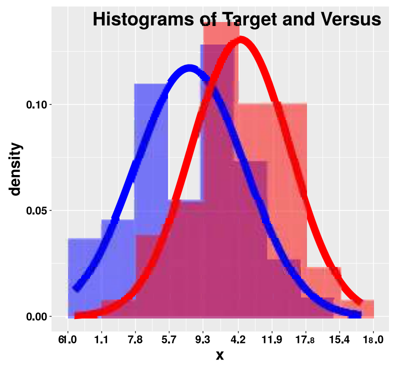
First, miRNA_classifierSetup is called by omitting miRNA and coefficient lists (see function documentation):

```
statisticsAlpha <- miRNA_classifierSetup(purifiedDatasetAlpha, inputTargetList=Target
    , inputVersusList=Versus, saveOutputFile=TRUE, outputFileBasename="mirnaAnalysis_
    datasetAlpha")
```

A list of the generated files is displayed by on-screen log messages that automatically follow the function call. In the present example, the list includes three graphic files, one for each miRNA (FX, FY, FZ). Each graphic file contains two histograms. The content of the three files is shown in the following figure: Target and Versus data are respectively shown in blue and red.


(FX)

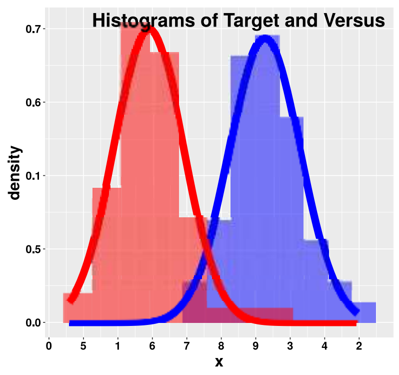

(FY)

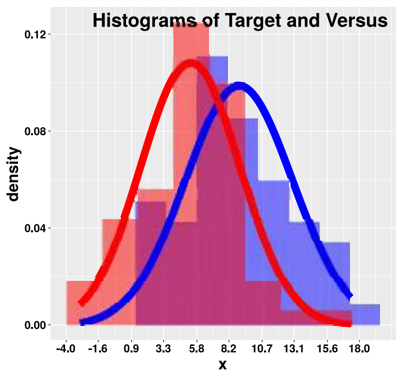

(FZ)

As shown above no clear separation between Target and Versus sets is provided by any of the three miRNAs. The function returns more detailed statistical information, including cross-correlations between miRNAS: a discussion on how to use this information to infer an improved linear combination of the miRNA expressions can be found in the *main reference* . However, for the sake of this example, let us repeat the last analysis by using a normalizer, i.e. by selecting one miRNA to be subtracted from the others. To carry out this operation, miRNA_classifierSetup is called by listing a single miRNA in its arguments, without any coefficient (see function documentation):

```
statisticsAlphaNorm <- miRNA_classifierSetup(purifiedDatasetAlpha, inputTargetList=
    Target, inputVersusList=Versus, inputMiRNAList=c("FZ"), saveOutputFile=TRUE,
    outputFileBasename="mirnaAnalysisNorm_datasetAlpha")
```

In this case only two graphic files are generated, one for ΔFX = FX−FZ and one for ΔFY = FY−FZ, as shown below:


(FX - FZ)


(FY - FZ)

The combination FX−FZ appears to discriminates the two sets. For the sake of this example, one could then use as classifier the linear combination $1 \cdot \mathrm{FX} + (-1) \cdot \mathrm{FZ}$.

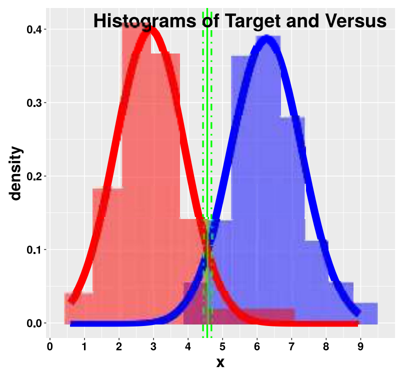
```
mirnaToUse <- c("FX", "FZ")
coefficientsToUse <- c(1.0, -1.0)
```

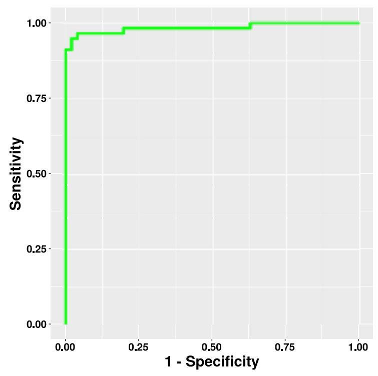
Training of a Bayesian classifier

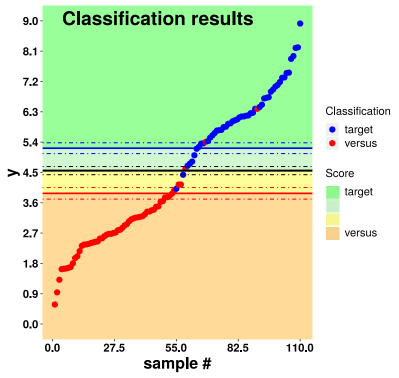
In order to train a Bayesian classifier, the miRNA_classifierSetup function has to be used in *“Training mode”*. The required parameters – namely the Target and Versus sets, and the lists of miRNAs and coefficients – have to be entered upon the analysis carried out in *“Analysis mode”*. The miRNA_classifierSetup function is thus called as follows:

```
thresholdValues <- miRNA_classifierSetup(purifiedDatasetAlpha, inputTargetList=Target
    , inputVersusList=Versus, inputMiRNAList=mirnaToUse, coeffList=coefficientsToUse,
     saveOutputFile=TRUE, outputFileBasename="threshold_datasetAlpha")
```

The function saves a file containing the diagnostic threshold values, as well as the three figures shown below and corresponding to the Target-Versus histograms (left), the ROC curve (center), and a plot of all scores (right).







Diagnosis of a test dataset

Once trained, the classifier can be used to classify, i.e. to assign a diagnosis, to subjects contained within other datasets. As an example, this section shows how to use the previously-trained classifier to classify the second synthetic dataset available, namely `synthetic_dataset_beta.dat`, which includes 200 subjects. In this case, no `Class` column is present. The quality control stage is identical to the case of the alpha dataset:

```
datasetBeta <- read.table(file="dataset_beta.dat", header=TRUE)
preprocessedDatasetBeta <- miRNA_expressionPreprocessing(datasetBeta, multipletSize
    =3)
```

Outliers are then removed from the dataset by using the quality threshold values estimated in the case of the alpha dataset and loaded from the related file:

```
qualityThresholdValues <- miRNA_loadQualityThreshold("qualityThreshold_datasetAlpha.
    dat")
purifiedDatasetBeta <- miRNA_removeOutliers(preprocessedDatasetBeta,
    qualityThresholdValues)
```

Similarly, the diagnostic threshold values previously obtained by training the classifier are loaded from the related file:

```
thresholdValues <- miRNA_loadDiagnosticThreshold("threshold_datasetAlpha.txt")
```

The dataset is classified by calling `miRNA_diagnosis`. The lists of miRNAs and coefficients are the same as the ones used in the training step:

```
mirnaToUse <- c("FX", "FZ")
coefficientsToUse <- c(1.0, -1.0)
diagnosedBeta <- miRNA_diagnosis(purifiedDatasetBeta, inputMiRNAList=mirnaToUse,
    coeffList=coefficientsToUse, inputThreshold=thresholdValues, saveOutputFile=TRUE,
     outputFileBasename="diagnosis_datasetBeta")
```

Finally, the data file `diagnosis_datasetBeta.dat` contains the results of the diagnosis:

```
Subject Diagnosis       Score
52      versus          0.4560
49      versus          0.8917
167     versus          1.0935
57      versus          1.1289
...
189     target          8.2310
192     target          8.2479
72      target          8.7586
```

Further examples

The example directory also contains two example scripts concerning the analysis of real, publicly available datasets. In both cases, a copy of the dataset is provided in the same directory.

- The example script `example_real_dataset_1.R` concerns data published in the following work:

  > L. Ricci, V. Del Vescovo, C. Cantaloni, M. Grasso, M. Barbareschi and M. A. Denti, *Statistical analysis of a Bayesian classifier based on the expression of miRNAs*, BMC Bioinformatics **16**:287 (2015).
  > DOI: 10.1186/s12859-015-0715-9

  and publicly available within the corresponding supplementary materials. Data are provided in the example directory as the plain text files `real_dataset_1_training.dat`, `real_dataset_1_testing.dat` and `real_dataset_1_additional.dat`.

- The example script `example_real_dataset_2.R` concerns data published in the following work:

  > S. Detassis, V. del Vescovo, M. Grasso, S. Masella, C. Cantaloni, L. Cima, A. Cavazza, P. Graziano, G. Rossi, M. Barbareschi, L. Ricci and M. A. Denti, *miR375-3p Distinguishes Low-Grade Neuroendocrine From Non-neuroendocrine Lung Tumors in FFPE Samples*, Frontiers in Molecular Biosciences **7**:86 (2020).
  > DOI: 10.3389/fmolb.2020.00086

  and submitted to GEO. Data are provided in the example directory as the plain text files `real_dataset_2_training.dat` and `real_dataset_2_testing.dat`.

In both cases, pipelines are similar to the one described above. Both pipelines reproduce the results that can be found in the corresponding reference. Please refer to the corresponding R scripts for details.