Description of *ExiMiR*

Sylvain Gubian, Alain Sewer, PMP SA

October 30, 2018

Contents

1 Introduction

The *ExiMiR* package provides tools for normalizing miRNA expression data obtained primarily from Exiqon miRCURY LNA™ arrays (though other formats are supported as well). It gives the possibility of applying a novel miRNA-specific normalization method using spike-in probes and is based on controlled assumptions [1]. These features allow to take into account the differences between miRNA and gene (mRNA) expression data, as discussed in a recent study [2].

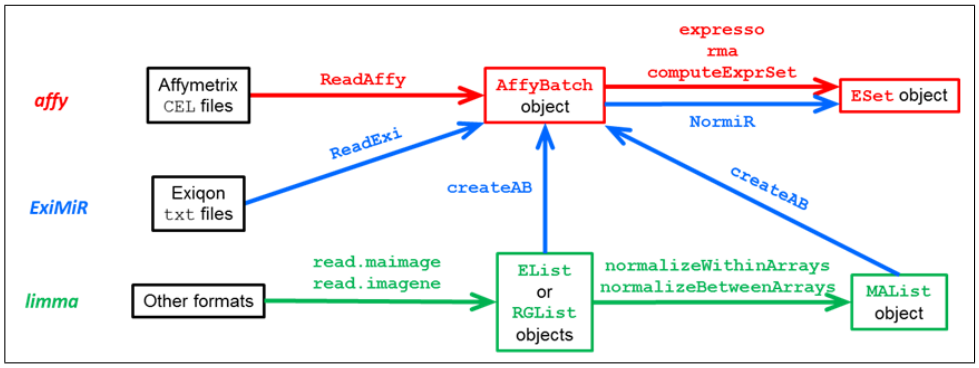

Figure 1: Overview of the functions of the ExiMiR package (in blue).

ExiMiR is particularly suited for two-color microarray experiments using a common reference. In such cases the spike-in probe-based normalization method allows to treat the raw data as if they were coming from single-channel arrays, like Affymetrix® Genechip®. This is why the objects and functions in *ExiMiR* have been chosen to closely resemble those of the "single-color" *affy* package, while remaining compatible with those of the "two-color" *limma* package, as shown on Figure 1.

The latest versions of *ExiMiR* explicitly integrate the raw data objects from the *limma* package. Thus the spike-in probe-based normalization method can be applied to raw data formats other than Exiqon or Affymetrix, as long as suitable "spike-in probes" are provided.

Further features of *ExiMiR* include:

- reading raw data in the ImaGene® TXT format used for the Exiqon miRCURY LNA™ arrays;
- allowing to update the array probe annotations to the latest miRBase releases incorporated in the Exiqon GAL files;
- enabling a background correction adapted to the raw data format by wrapping the *limma* function `backgroundCorrect`;
- entering a user-defined set of appropriate probe IDs that can be used by the normalization method instead of the automatically selected spike-in probes;
- enabling to generate the same objects for summarized miRNA expression data after having applied the alternative normalization methods contained in both *affy* and *limma* packages.

This vignette shows how to process raw expression data obtained from the Affymetrix® Genechip® miRNA array (CEL and CDF files) in order to illustrate the similarities between *ExiMiR* and *affy*. An example with other raw data formats supported by *limma* is also discussed.

2 Raw and annotation data

This section describes how to find raw and annotation data on which *ExiMiR* can be applied. [a] It covers both Affymetrix (CEL and CDF) and Exiqon/ImaGene (TXT and GAL) cases. If you have your own expression data in CEL or TXT formats, then you just need to complete them with the annotation files in CDF or GAL formats, respectively, as described below. Do no forget the appropriate `samplesinfo.txt` file for the Exiqon case. In case your raw data are neither in Affymetrix CEL nor in Exiqon/ImaGene TXT formats, then first read Subsection 3.3 to see whether *ExiMiR* can be applied.

2.1 Affymetrix

First create a directory `Affymetrix` in your file system. The GEO repository (http://www.ncbi.nlm.nih.gov/geo/) contains several datasets using the Affymetrix miRNA-1_0 array. We choose the series GSE19183 for which the raw data file `GSE19183_RAW.tar` can be downloaded from this URL. Extract the CEL files into the `Affymetrix` directory. As long as `R` can connect to the Bioconductor server (http://www.bioconductor.org/), the annotation will be generated automatically by the *affy* package when reading the CEL files (see Subsection 3.1). If this is not the case, you need to get and load the package *mirna10cdf*, which will create the appropriate annotation environment for the dataset GSE19183 in your `R` session.

2.2 Exiqon

First create a directory `Exiqon` in your file system. The GEO series GSE20122 contains a suitable raw data file `GSE20122_RAW.tar` at the following URL. After downloading it, extract and expand (`gunzip`) the enclosed raw data file in ImaGene TXT format into the `Exiqon` directory. Then download the corresponding annotation GAL file from the following URL into the `Exiqon` directory as well. Finally, copy and paste the content of Appendix A of this vignette into a TAB-separated text file called `samplesinfo.txt` and also located the `Exiqon` directory. This file is required by *ExiMiR* to match the raw data results from the Hy3 and Hy5 channels. [b] It contains the names of all the TXT files in the experiment, organized in a table with one row for each array and two columns corresponding to the two channels Hy3 and Hy5. [c] It is very similar to the file `Targets.txt` required by the *limma* package and is usually provided by Exiqon together with the raw data TXT files.

3 Raw data normalization

This section describes how to apply *ExiMiR* to normalize raw miRNA expression data obtained from the Affymetrix® miRNA Genechip® or from the Exiqon miRCURY LNA™ arrays. Under

[a] N.B.: `R` objects corresponding to the raw and annotation data described in this section are provided by the *ExiMiR* package, see Section 5

[b] In the Exiqon common reference experimental design, Hy3 and Hy5 are usually the samples and the reference channels, respectively.

[c] For practical purposes only 12 of the 54 Hy3-Hy5 raw data filename pairs from the GEO series GSE20122 are listed in Appendix A. Feel free to complete `samplesinfo.txt` with the 42 remaining ones if you want to be exhaustive!

specific conditions, this is also possible for other raw data formats (see Subsection 3.3). Notice that although these descriptions are generic, some of the filenames given in the command-line examples might differ from case to case (e.g. GAL filenames). Begin by launching an `R` session at the same level as the `Affymetrix` and `Exiqon` directories created in Section 2.

3.1 Affymetrix: from CEL files to ExpressionSet objects

You don't need to run the following three commands if your `R` *installation can connect the Bioconductor server* (http://www.bioconductor.org/).

The array annotation environment is created by loading the *mirna10cdf* package:

```
R> library(mirna10cdf)
```

Using the `makecdfenv` package, the above corresponds to the following two commands

```
R> library(makecdfenv)
R> cdfenv <- make.cdf.env(cdf.path='Affymetrix', filename='miRNA-1_0.CDF')
```

provided that the CDF file `miRNA-1_0.CDF` (formerly downloadable from the Affymetrix website) is located in the `Affymetrix` folder.

Load the CEL file raw data into an AffyBatch object using the *affy* package: [d]

```
R> library(affy)
R> abatch <- ReadAffy(cdfname='cdfenv', celfile.path='Affymetrix')
```

Raw data normalization is directly applied on `abatch` to create an ExpressionSet object. For instance: [e]

```
R> eset.rma <- expresso(abatch, bgcorrect.method='rma',
                                normalize.method='quantiles',
                                pmcorrect.method='pmonly',
                                summary.method='medianpolish')
```

As an alternative to the above RMA quantile normalization validated for gene (mRNA) expression [3], the spike-in probe-based method implemented in the *ExiMiR* function `NormiR` might give better results for miRNA expression data [1, 2]:

```
R> library(ExiMiR)
R> eset.spike <- NormiR(abatch, bgcorrect.method='rma',
                                normalize.method='spikein',
                                pmcorrect.method='pmonly',
                                summary.method='medianpolish')
```

[d] Drop the argument `cdf.name='cdfenv'` if the environment `cdfenv` has not been defined previously with `make.cdf.env`.

[e] This is equivalent to `eset.rma <- rma(abatch)`.

For the GSE19183 dataset, the assumptions allowing the application of the `NormiR` spike-in probe-based normalization are not satisfied and a default median normalization is performed instead. Section 4 describes this safeguarding strategy and the options allowing to deal with problematic cases. If the `NormiR` assumptions are satisfied, a series of control figures are generated. Their description is given in Section 4.1 below.

3.2 Exiqon: from TXT files to ExpressionSet objects

First load the *ExiMiR* package:

```
R> library(ExiMiR)
```

Then create the array annotation environment using the GAL file and the `make.gal.env` function:

```
R>  make.gal.env(galname='galenv', gal.path='Exiqon')
```

Read the raw data TXT files into an AffyBatch object using the `ReadExi` function: [f]

```
R> ebatch <- ReadExi(galname='galenv', txtfile.path='Exiqon')
```

Similarly to the Affymetrix case in Subsection 3.1, the raw data normalization is applied on `ebatch` to create an ExpressionSet object. For instance the RMA quantile normalization from the *affy* package [3], using the option `bg.correct=FALSE`, as recommended by a recent study[4]: [g]

```
R> eset.rma <- NormiR(ebatch, bg.correct=FALSE,
                              normalize.method='quantile',
                              summary.method='medianpolish')
```

However, the assumptions for applying the quantile normalization are not guaranteed to be satisfied in the case of miRNA expression data [1, 2]. It might be better to use the spike-in probe-based method from *ExiMiR*:

```
R> eset.spike <- NormiR(ebatch, bg.correct=FALSE,
                                normalize.method='spikein',
                                summary.method='medianpolish')
```

If all the assumptions underlying the application of the `NormiR` spike-in probe-based normalization are satisfied, a series of control figures are generated, which will be explained in Subsection 4.1. If one or more assumptions are not met, then the median normalization is applied instead. However, *ExiMiR* offers several options to deal with such situations and still allow the application of the spike-in probe-based method, as explained in Subsections 4.2 and 4.3.

[f] `ReadExi` stores the dual-channel foreground and background intensity data as follows: the foreground data matrix of the Hy3 channel (second column of `samplesinfo.txt`) and the foreground data matrix of the Hy5 channel (third column of `samplesinfo.txt`) are concatenated and stored in the `exprs` slot of the resulting AffyBatch object. The same goes for the background data in the `se.exprs` slot.

[g] This command is not equivalent to `eset.rma <- rma(ebatch,background=FALSE)` because `NormiR` applies the quantile normalization separately on the data from the Hy3 and Hy5 channels (which have been concatenated in `ebatch` by `ReadExi`) while `rma` mixes the data from two channels.

3.3 Other formats: reading with *limma* and normalizing with *ExiMiR*

The approach consists in reading the raw data using the *limma* package and then transforming the result into an AffyBatch object using the function `creatAB`, as illustrated on Figure 1. Therefore the conditions for applying *ExiMiR* on other raw data forms are (1) the ability to load the raw data into *limma* objects (EList or RGList), and (2) the availability of probe sets that can be used as "spike-in probes" during normalization. The parameter `probeset.list` of the function `NormiR` allows to select the IDs of such "spike-in probes" for running the normalization method, as long as they satisfy its underlying assumptions (see Subsections 4.1 and 4.3).

As a toy example, this approach is applied on the Exiqon raw data from Subsection 3.2, which satisfy the two conditions discussed above. You will need to set the parameters of the function `read.maimage` according to your particular case (see Chapter 4 of the *limma* User's Guide for more details).

First create the `Targets.txt` file in the `Exiqon` directory. In our case this consists simply in replacing the header line of the file `samplesinfo.txt` by "`FileName Cy3 Cy5`" (TAB-separated) and saving the result as `Targets.txt`. Then proceed by reading the raw data and annotating the probe sets using the available GAL file:

```
R> library(limma)
R> targets <- readTargets(path='Exiqon')
R> RGList <- read.maimages(targets[,c('Cy3','Cy5')], source='imagene', path='Exiqon')
R> RGList$genes <- readGAL(path='Exiqon')
R> RGList$printer <- getLayout(RGList$genes)        # optional
```

The last line enables plotting the spatial distributions of the intensities of the arrays using the `image` method of the AffyBatch object created below. Running `getLayout` is not always possible (see Section 4.7 of of the *limma* User's Guide) and not needed for the spike-in probe-based normalization. The AffyBatch object is created using the function `createAB` of the *ExiMiR* package: [h]

```
R> library(ExiMiR)
R> obatch <- createAB(RGList)
```

Next the "spike-in probes" used for normalization must be specified. In our toy example they will consist in a subset of the available annotated spike-in probe sets, excluding `spike_control_f`. Other choices are possible, as long as `NormiR` can be run (see Subsections 4.1 and 4.3). The raw data normalization is then performed similarly as above (see Subsection 3.2):

```
R> spikein.all <- grep('^spike', featureNames(obatch), value=TRUE)
R> spikein.subset <- setdiff(spikein.all, 'spike_control_f')
R> eset.spike <- NormiR(obatch, bg.correct=FALSE,
                            normalize.method='spikein',
                            normalize.param=list(probeset.list=spikein.subset),
                            summary.method='medianpolish')
```

[h] `createAB` stores dual-channel foreground and background intensity data in the same way as `ReadExi`, see footenote f in Subsection 3.2.

The comments made after the execution of NormiR at the end of the previous subsection remain valid in the present case.

3.4 More information about NormiR

This paragraph provides additional information about the NormiR function and can be skipped on first reading.

The function NormiR has been designed to closely resemble the expresso function of the *affy* package. Therefore their usages are very similar (see Section 3.3.1 of the *affy* vignette):

```
ESet <- NormiR(AffyBatch,
             # background correction
               bg.correct=TRUE,
               bgcorrect.method='auto',
               bgcorrect.param=list(),
             # normalization
               normalize=TRUE,
               normalize.method='spikein',
               normalize.param=list(),
             # PM correction (enabled only when MM-values are available)
               pmcorrect.method='pmonly',
               pmcorrect.param=list(),
             # summarization
               summary.method='medianpolish',
               summary.param=list(),
             # miscellaneous
               verbose=TRUE
               ...)
```

The methods proposed by NormiR for the first step of background correction are provided by the *affy* or *limma* packages, depending on whether the input AffyBatch object has been created with ReadAffy or ReadExi/createAB, respectively. The *ExiMiR* function NormiR.bgcorrect.methods lists the available methods, as for instance in the case of the AffyBatch objects defined in Subsections 3.1 and 3.2:

```
R> NormiR.bgcorrect.methods(abatch)
[1] "bg.correct"        "mas"        "none"        "rma"
R> NormiR.bgcorrect.methods(ebatch)
[1] "edwards"        "half"        "minimum"        "movingmin" "none" "normexp" ...
```

The *limma* background correction methods requiring background intensity values are provided the suitable input data, automatically extracted from the AffyBatch object previously created with ReadExi/creatAB. [i] Therefore, applying for instance the normexp background correction on the ebatch AffyBatch object is straightforward with NormiR:

[i] The way ReadExi and creatAB store dual-channel foreground and background intensity data in an AffyBatch object has been described above, see Footnote f.

```
R> eset.spike <- NormiR(ebatch, bgcorrect.method='normexp'
                                bgcorrect.param=list(offset=50),
                                normalize.method='spikein',
                                summary.method='medianpolish')
```

For the next step of normalization, the methods are independent of the AffyBatch objects and explicitly enumerated by the *ExiMiR* function `NormiR.normalize.methods`:

```
R> NormiR.normalize.methods()
[1] "mean"        "median"        "none"        "quantile"        "spikein"
```

As explained above (see Footnote g), the quantile normalization `quantile` treats separately the two channels in case of dual-channel raw data processed by `ReadExi` or `creatAB`. When the spike-in probe-based normalization `spikein` is chosen, `NormiR` automatically detects the spike-in probes, based on the Affymetrix or Exiqon annotations. If this detection is not successful or if you want to use your own selection of "spike-in probes", the parameter `probeset.list` in `NormiR` allows to enter the appropriate list of probe set IDs (see also Subsection 4.3). More generally, all the `NormiR` parameters enabling fine-tuning of the spike-in probe-based normalization `spikein` are provided as a list with their default values by the function `NormiR.spikein.params`. You can modify this list and submit it to `NormiR` using the parameter `normalize.param`. [j] This constitutes the starting point for controlling the execution of `NormiR` (see Section 4 for details). For instance, using the subset of spike-in probes `spikein.subset` defined in Subsection 3.3:

```
R> NormiR.spikein.params()
R> spikein.params <- list(probeset.list=spikein.subset,
                          loess.span=0.6,
                          force.zero=TRUE)
R> eset.spike <- NormiR(ebatch, bgcorrect.method='normexp'
                                bgcorrect.param=list(offset=50),
                                normalize.method='spikein',
                                normalize.param=spikein.params,
                                summary.method='medianpolish')
```

The next step of PM correction is enabled only when the CDF environment provides values for the MM probes in the AffyBatch object, which is not the case of the Exiqon data considered here.
The last step is the probe summarization, for which `NormiR` proposes the methods of the *affy* package:

```
R> NormiR.summary.methods()
[1] "avgdiff"        "liwong"        "mas"        "medianpolish"        "playerout"
```

Actually, `NormiR` is a wrapper for three lower-level functions, which all accept the corresponding `NormiR` parameters. Therefore the previous `NormiR` command is equivalent to:

[j] Due to compatibility constraints with older versions of `NormiR`, the normalization parameters provided by `NormiR.spikein.params` are also adjustable as explicit `NormiR` arguments, for instance `NormiR(ebatch, normalize.method='spikein', normalize.param=list(figures.show=FALSE))` is equivalent to `NormiR(ebatch, method='spikein', figures.show=FALSE)`.

```
R> ebatch.tmp <- bg.correct.miR(ebatch, bgcorrect.method='normexp',
                                                bgcorrect.param=list(offset=50))
R> ebatch.tmp <- norm.miR(ebatch.tmp, normalize.method='spikein',
                                            normalize.param=spikein.params)
R> eset.spike <- summarize.miR(ebatch.tmp, summary.method='medianpolish')
```

Separating the normalization steps allows to more efficiently search the most appropriate parameters for the spike-in probe-based normalization without running the time-consumming summarization step at every attempt (see Section 4 for details).

The low-level summary function `summarize.miR` can also be run in combination with `createAB` in order to summarize a MAList object obtained with a normalization method of the *limma* package (see Figure 1). For instance, using the RGList object defined in Subsection 3.3:

```
R> MAList.tmp <- normalizeWithinArrays(RGList, method='loess')
R> MAList <- normalizeBetweenArrays(MAList.tmp, method='Rquantile')
R> obatch <- createAB(MAList)
R> oset.rrma <- summarize.miR(obatch, summary.method='medianpolish')
```

4 Troubleshooting and fine-tuning options

ExiMiR provides several functionalities to ensure a safe application of the spike-in probe-based normalization. This section first describes the control figures generated during the execution of the `NormiR` function (Subsection 4.1). They allow to identify the problems that may arise when applying *ExiMiR*. Subsection 4.2 will help understanding their origin and deciding whether to still use *ExiMiR* (with different parameters) or to switch to another method like median normalization. The fine-tuning options offered by *ExiMiR* are explained in Subsection 4.3.

4.1 Control figures

In order to follow the execution of the spike-in probe-based normalization implemented in `NormiR`, a series of three control figures are generated for each channel of the input data. They allow to confirm the successful application of the normalization method but also to detect possible anomalies, that can be then treated with the options described in Subsection 4.3. This feature runs by default and can be deactivated by setting `figures.show=FALSE` in `NormiR`.

The three control figures generated for the Hy3 channel of the Exiqon example from Subsections 2.2 and 3.2 are briefly described hereafter. For more details see [1].

Correction of the spike-in probe set intensities The four panels in Figure 2 show the successive steps in removing the array-dependent biases from the spike-in probe set intensities. A meaningful application of `NormiR` indeed requires that the spike-in probe set intensities display coherent deviations across the arrays of the experiment. Such a behavior manifests itself by roughly parallel curves on the upper-left panel and by collapsing ones on the upper-right panel. The normalization correction consists first in subtracting

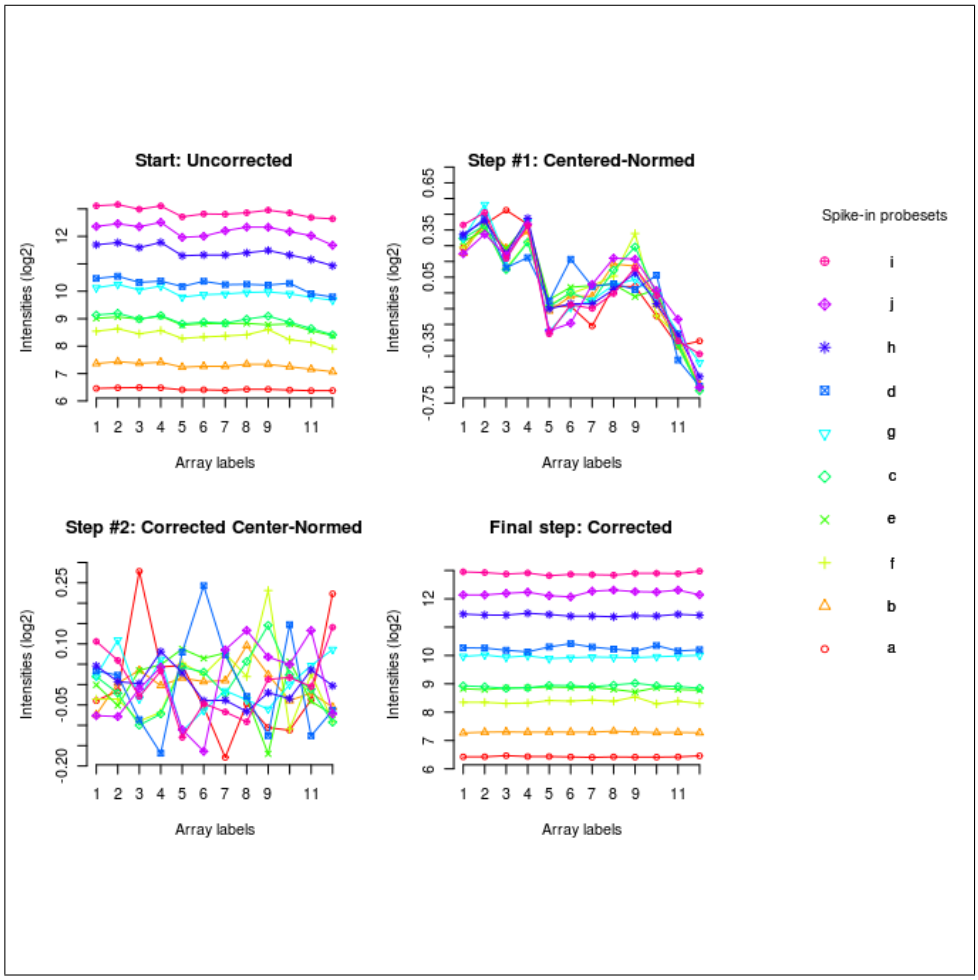

Figure 2: Correction of the spike-in probe set intensities (Hy3 channel)

this common variance (lower-left panel) and second in transforming back to the original intensity range (lower-right panel). Correcting the curves is proved efficient when the final ones appear 'straighter' than the initial ones.

Performance of the spike-in probe set intensity correction Figure 3 contains two measures for quantitatively assessing the performance of the spike-in probe set intensity correction used by `NormiR`. The upper panel shows a heatmap of the Pearson correlations between the array-dependent raw intensities of the spike-in probe sets, i.e. between the curves displayed on the upper-left panel of Figure 2. If the values are globally larger than 0.5, then the array-dependent biases are sufficiently coherent and applying `NormiR` is justified. The lower panel displays the variance ratio of the spike-in probe sets intensities before and after the correction. They correspond to the curves in the upper-left and lower-right panels of Figure 2. If these ratios are sufficiently low, then the `NormiR` approach was effective.

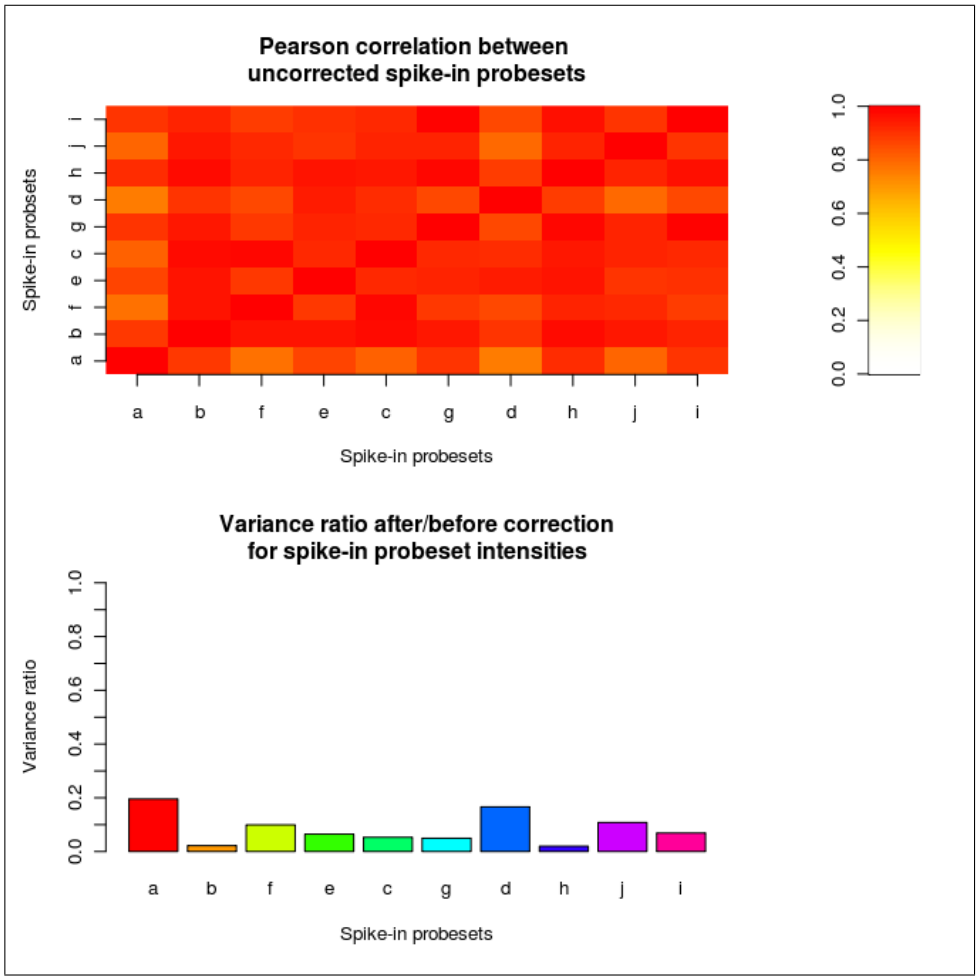

Figure 3: Performance of the spike-in probe set intensity correction (Hy3 channel)

Intensity-dependent correction functions for all probes Figure 4 displays the intensity and array dependent correction functions that `NormiR` applies to all miRNA probes to perform the normalization. It is constructed based on the spike-in probe corrections, already shown on Figures 2 and 3. Several requirements are necessary to ensure a stable coverage of the whole range of probe intensities measured on the array. *ExiMiR* automatically performs checks to prevent critical situations where its meaningful application is not guaranteed. Sometimes the constructed correction functions do not look good, even if `NormiR` ran smoothly. Dealing with such situations is also described in Section 4.3 below.

4.2 Possible problems

The application of *ExiMiR* fundamentally assumes that the spike-in probes capture the greatest part of the between-array technical variability in the miRNA expression data. This is normally

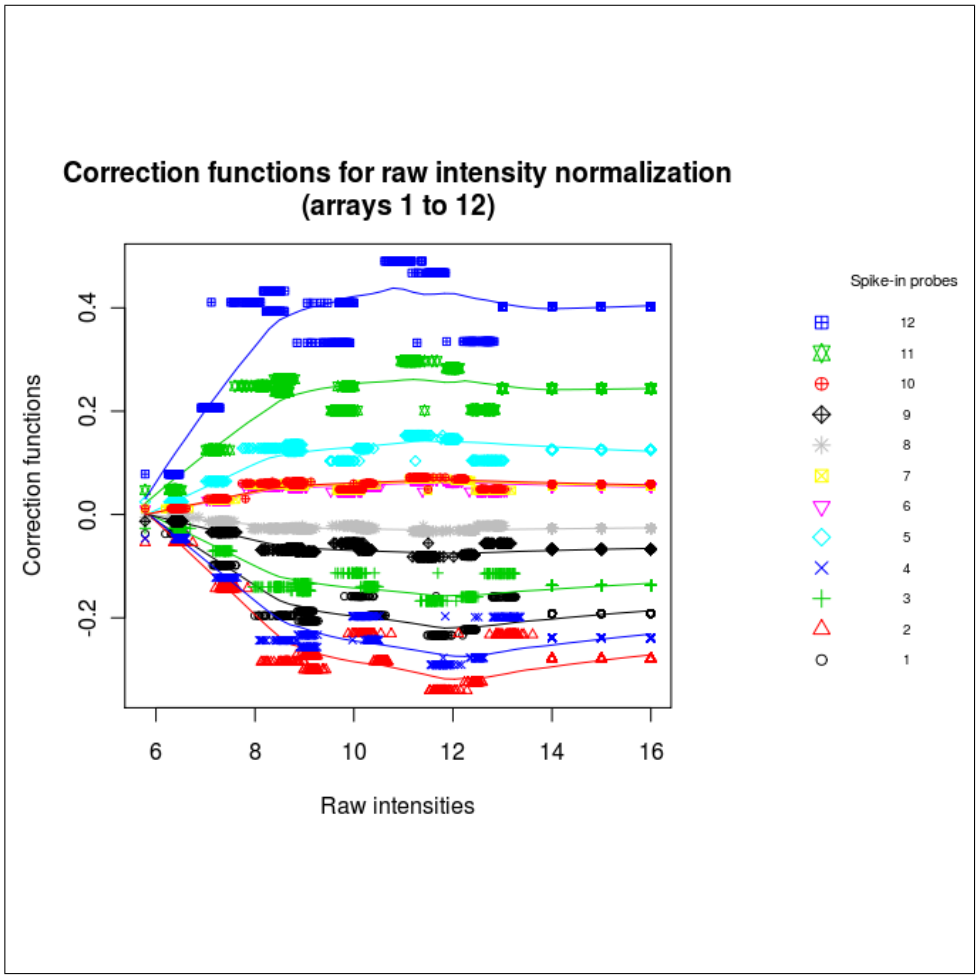

Figure 4: Intensity-dependent correction functions (Hy3 channel)

the case when the processing of the RNA samples prior to the addition of the spike-in RNAs is suitably standardized and controlled. If this condition is satisfied, then *ExiMiR* requires three features from the spike-in control probes to be meaningfully applied, see [1]. These features are automatically tested by the software. In case of failure, median normalization is used instead of spike-in probe-based method. However the threshold values used in these tests can be changed to force the application of the spike-in probe-based method. Its consequences can then be investigated on the control figures described in Subsection 4.1 to decide whether the application of *ExiMiR* was justified or not. Other problems like annotation conflicts are also supported by *ExiMiR*.

Here is the list of the problematic situations covered by *ExiMiR*, arranged by potential order of appearance.

Incompatibility between GAL and TXT files If the array annotation contained in the GAL file is not compatible with the one contained in the TXT files, or if there is no GAL

file available, then `ReadExi` directly generates a default `galenv` environment from the annotation contained in the TXT files.

Insufficient coherence between spike-in probe sets If the raw intensities of the spike-in probe sets are not sufficiently coherent across the arrays of the experiment, i.e. if the mean of the off-diagonal elements of the Pearson correlation matrix shown on the upper panel of Figure 3 is smaller than 0.5, then a median normalization is applied. The value can be changed by using the `min.corr` of `NormiR`.

Specificity of the spike-in probeset intensities If the spike-in probeset intensities are not specific, i.e. if the intensity ranges covered by the probes mapping to the same probe sets are too large, then computing the intensity-dependent correction functions from Figure 4 becomes problematic. The intensity-independent median normalization is preferred in this case. The `NormiR` parameter `max.log2span` can be changed to allow for probeset intensity ranges larger than the default value 1.

Insufficient coverage of the probe intensity range If the range [∼6,∼16] of all array probe intensities is not appropriately covered by the spike-in probe intensities, then computing the intensity dependence of the correction functions from Figure 4 becomes unstable. The `NormiR` parameter `cover.int` tests the size of the largest intensity interval between two consecutive spikes. Its default value is 1/3. The `NormiR` parameter `cover.ext` tests the minimal ratio between the intensity range covered by the spike-in probes and the one covered by all probes on the array. Its default value is 1/2. These two values can be changed but an eye must be kept on their consequences on the correction functions from Figure 4, since the latter are not explicitly tested by *ExiMiR*. The `NormiR` options for computing these correction functions are explained in Subsection 4.3 below.

4.3 *NormiR* options for computing the correction functions

The results for the spike-in probe-based correction functions displayed on Figure 4 are not tested automatically by *ExiMiR* and might not be entirely satisfactory. This might be due to multiple reasons, ranging from inhomogeneous affinities across the spike-in probe sets to an inappropriate coverage of the probe intensity range. *ExiMiR* offers the possibility of fine-tuning the parameters used by `NormiR` to improve the stability of the correction functions (see also Subsection 3.4 for command line examples).

Selection of the spike-in probes If the control figures show that only a subset of the spike-in probe sets displays the appropriate behavior, it can be manually selected for the calculation of the correction functions. Set the `NormiR` parameter `probeset.list` to the list of the appropriate spike-in probe sets IDs. [k]

Overall LOESS smoothing If the correction functions 'wiggle' too much, the `NormiR` parameter `loess.span` can be set to higher values to better smooth the resulting curves. By default, it takes the value 5/(number of spike-in probe sets), e.g. 5/10 in the Exiqon

[k] Actually `probeset.list` allows to select any set of probe set IDs and tentatively run `NormiR`. However, the safeguarding strategy implemented in `NormiR` prevents inappropriate choices of "spike-in probes".

case. In the extreme cases of values close to 1, the intensity dependence of the correction is lost and the results become very similar to a mean or a median normalization.

Low-intensity stabilization If one correction function change its sign in the low intensity range, then an inclusion into the LOESS smoothing of a zero value at the intensity minimum will prevent this feature. Set the `NormiR` parameter `force.zero` to `TRUE` to activate this functionality.

High-intensity extrapolation It often occurs that the largest spike-in probeset intensities are lower than the largest probe intensities on the array. In this case `NormiR` needs to include extrapolated values into the LOESS smoothing in order to compute the correction functions in the high-intensity range. Fortunately this step is quite stable thanks to the fact that high intensity values are less noisy. By default `NormiR` uses the mean of the correction values of two spike-in probe sets with the largest intensities. The parameter `extrap.points` allows to change the number of spike-in probe sets used in the extrapolation and `extrap.method` determines the extrapolation method.

5 Concrete example with provided data

ExiMiR provides datasets that allows one to test the functions described in this vignette. The test data are in the `R` objects obtained as described in Section 2. They can be used as follows, which reproduces the commands explained in Section 3.

5.1 Affymetrix

Start by loading the *ExiMiR* package and the AffyBatch object, which includes the appropriate annotation environment and corresponds to the data described in Section 2.1:

```
> library(ExiMiR)
> data(GSE19183)
```

Apply the RMA quantile normalization on the AffyBatch object `GSE19183` to create the ExpressionSet object `eset.rma` containing the normalized data:

```
> library(affy)
> eset.rma <- expresso(GSE19183, bgcorrect.method='rma',
+                              normalize.method='quantiles',
+                              pmcorrect.method='pmonly',
+                              summary.method='medianpolish')

background correction: rma
normalization: quantiles
PM/MM correction : pmonly
expression values: medianpolish
background correcting...done.
normalizing...done.
7815 ids to be processed
```

```
|                    |
|####################|

>
```

The spike-in probe-based normalization implemented in *ExiMiR* can be applied similarly:

```
> eset.spike <- NormiR(GSE19183, bgcorrect.method='rma',
+                                normalize.method='spikein',
+                                normalize.param=list(figures.show=FALSE),
+                                pmcorrect.method='pmonly',
+                                summary.method='medianpolish')

7815 ids to be processed
|                    |
|####################|
```

As explained at the end of Section 3.1, the spike-in probe-based normalization method implemented in `NormiR` can not be applied with its default settings. Use the `figures.show=TRUE` option to diagnose graphically the problem (see Section 4.1). The safeguarding strategies are described in Section 4.2 and the `NormiR` options described in Section 4.3. Here `NormiR` cannot be satisfactorily applied to `GSE19183` because the spike-in probe intensities do not appropriately cover the intensity range of all the array probes.

5.2 Exiqon

Load the *ExiMiR* package, the GAL annotation environment and the AffyBatch objects corresponding to the data described in Section 2.2:

```
> library(ExiMiR)
> data(galenv)
> data(GSE20122)
```

Apply the RMA quantile normalization on the AffyBatch object `GSE20122`, using the `rma` option `background=FALSE` as recommended by a recent study[4]. This creates the ExpressionSet object `eset.rma` containing the normalized data:

```
> eset.rma <- NormiR(GSE20122, bg.correct=FALSE,
+                              normalize.method='quantile',
+                              summary.method='medianpolish')

1992 ids to be processed
|                    |
|####################|
```

The spike-in probe-based normalization implemented in *ExiMiR* is applied as follows:

```
> eset.spike <- NormiR(GSE20122, bg.correct=FALSE,
+                                normalize.method='spikein',
+                                normalize.param=list(figures.show=FALSE),
+                                summary.method='medianpolish')

1992 ids to be processed
|                    |
|####################|
```

To obtain the same control figures as the ones displayed in Section 4.1, use the `NormiR` option `figures.show=TRUE`.

References

[1] Sewer A et *al.*, to be published.

[2] Sarkar D et *al.*, Quality assessment and data analysis for miRNA expression arrays, Nucleic Acids Research 2009, **37**(2).

[3] Irizarry RA et *al.*, Biostatistics 2003, **4**(2).

[4] López-Romero P et *al.*, Procession of Agilent microRNA array data, BMC Research Notes 2010, **3**:18.

A Content of the file "samplesinfo.txt"

Names	Hy3	Hy5
1	GSM503402_Hy3_Exiqon_14114402_S01_Cropped.txt	GSM503402_Hy5_Exiqon_14114402_S01_Cropped.txt
2	GSM503403_Hy3_Exiqon_14114403_S01_Cropped.txt	GSM503403_Hy5_Exiqon_14114403_S01_Cropped.txt
3	GSM503404_Hy3_Exiqon_14114404_S01_Cropped.txt	GSM503404_Hy5_Exiqon_14114404_S01_Cropped.txt
4	GSM503405_Hy3_Exiqon_14114405_S01_Cropped.txt	GSM503405_Hy5_Exiqon_14114405_S01_Cropped.txt
5	GSM503406_Hy3_Exiqon_14114406_S01_Cropped.txt	GSM503406_Hy5_Exiqon_14114406_S01_Cropped.txt
6	GSM503407_Hy3_Exiqon_14114407_S01_Cropped.txt	GSM503407_Hy5_Exiqon_14114407_S01_Cropped.txt
7	GSM503408_Hy3_Exiqon_14114408_S01_Cropped.txt	GSM503408_Hy5_Exiqon_14114408_S01_Cropped.txt
8	GSM503409_Hy3_Exiqon_14114409_S01_Cropped.txt	GSM503409_Hy5_Exiqon_14114409_S01_Cropped.txt
9	GSM503410_Hy3_Exiqon_14114410_S01_Cropped.txt	GSM503410_Hy5_Exiqon_14114410_S01_Cropped.txt
10	GSM503411_Hy3_Exiqon_14114411_S01_Cropped.txt	GSM503411_Hy5_Exiqon_14114411_S01_Cropped.txt
11	GSM503412_Hy3_Exiqon_14114412_S01_Cropped.txt	GSM503412_Hy5_Exiqon_14114412_S01_Cropped.txt
12	GSM503413_Hy3_Exiqon_14114413_S01_Cropped.txt	GSM503413_Hy5_Exiqon_14114413_S01_Cropped.txt