Quality assessment with arrayQualityMetrics

Audrey Kauffmann, Wolfgang Huber

November 7, 2008

Contents

Introduction

The goal of this vignette is to show how to obtain a quality assessment report by the use of `arrayQualityMetrics`. The first step of the analysis of microarray is to assess the quality of the experiment. You should also assess the quality after preprocessing your data. Each type of microarray data (one colour, two colour, Affymetrix, Illumina...) is represented by a class of object in Bioconductor. However, the `arrayQualityMetrics` function is used the same way on any kind of data. `arrayQualityMetrics` produces a *HTML* report as an output. The more information is in the input object, the more complete is the report produced.

1 Usage

The call of the function is the same for the various classes of objects but it depends on the specificities that you want in the report. The function `arrayQualityMetrics` is called with the following arguments:

- *expressionset*: is an object of class *ExpressionSet*, *AffyBatch*, *NChannelSet*, *RGList* or *BeadLevelList*.
- *outdir*: is the directory in which the result files are created. The default is the current directory.
- *force*: if TRUE, if outdir already exists, it will be overwritten. The default is FALSE.
- *do.logtransform*: if TRUE, the data are log transformed before the analysis. The default is FALSE.
- *split.plots*: if the number of studied arrays is more than 50 it is advised to define a number of experiments to represent on the density plots. The default is FALSE, meaning that all the density curves are represented on the same plot.
- *intgroup*: is the name of the column in the *phenoData* that contains the information about a covariate of interest to be shown as a side bar on the heatmap. The default name of this column is "Covariate".
- *grouprep*: define if yes or no (TRUE or FALSE) you want the boxplots and density plots to be coloured according to the groups set in 'intgroup'. The default is FALSE.

2 Output

A report named `QMreport.html` is produced in the subdirectory given as 'outdir'. It contains text illustrated by pictures. Each of the picture is linked to corresponding `.pdf` files in order to provide high quality images for publication. In the case of *AffyBatch* input, some Affymetrix specific plots are added to the standard report. However the way of calling the function remains the same. As explained in the Section 5, other plots can be added to the standard report if some specific columns are present in the input object.

3 Minimal use

3.1 Example on Affymetrix data

If you are working with Affymetrix chips, an *AffyBatch* object is the most appropriate to import your raw data in Bioconductor. To learn how to produce an *AffyBatch* from your data, please read the documentation from the *affy* package. We can use the *MLL.A AffyBatch* provided in the data package *ALLMLL* as an example to create a quality report. *MLL.A* has been obtained from the CEL files, using the `ReadAffy` function from the *affy* package.

```
> library("ALLMLL")
> data("MLL.A")
```

Now that the data are loaded, we can call `arrayQualityMetrics`.

```
> library("arrayQualityMetrics")
> arrayQualityMetrics(expressionset = MLL.A,
+                     outdir = "MLL",
+                     force = TRUE,
+                     do.logtransform = TRUE)
```

Here is the simplest way of calling the function. We give a name to the directory ('outdir') and we overwrite the possibly existing files of this directory ('force'). Finally, we set 'do.logtransform' to log transform the intensities.

3.2 Example with one colour arrays

If you are working on one colour non Affymetrix chips, you can load your data in Bioconductor as an *ExpressionSet* object. Please see the documentation of the *Biobase* package to do so. The *ExpressionSet* is made to contain one colour data set. Here, as an example, we can simulate this by using a preprocessed object from *MLL.A* which contains one value for each gene.

```
> rMLL = rma(MLL.A)
```

```
Background correcting
Normalizing
Calculating Expression
```

We can then call the `arrayQualityMetrics` function the standard way:

```
> arrayQualityMetrics(expressionset = rMLL,
+                     outdir = "rMLL",
+                     force = TRUE)
```

We do not need to set 'do.logtransform' as the data are already log transformed.

3.3 Example with two colours arrays

If you are working on two colour chips, you can create a *NChannelSet* to contain your data. The documentation of the *Biobase* package gives instruction to build a *NChannelSet*. As an instance of a *NChannelSet*, we can use the *CCl4* data package and normalize it using the variance stabilization method available in the package *vsn*.

```
> library("CCl4")
> data("CCl4")
> nCCl4 = justvsn(CCl4, subsample=2000)
> arrayQualityMetrics(expressionset = nCCl4,
+                     outdir = "CCl4norm",
+                     force = TRUE)
```

3.4 Other type of objects

Through the previous examples, you have seen that `arrayQualityMetrics` is used the same way on different types of object. If you have Illumina bead arrays, you can build a *BeadLevelList*, which is explained in the *beadarray* package, and run `arrayQualityMetrics`. It is also possible to use `arrayQualityMetrics` on *RGList*, *MAList*, *marrayRaw* and *marrayNorm*. More information about the *RGList* and *MAList* classes is given in the package *limma*. The objects *marrayRaw* and *marrayNorm* are explained in the vignette of the package *marray*.

4 Playing with the arguments

4.1 Factor of interest

A useful feature of `arrayQualityMetrics` is the possibility to show the results by taking into account a factor of interest, thanks to 'intgroup' and 'grouprep'. It does not change the way of computing the quality metrics. By setting 'intgroup' a bar on the side of the heatmap with colours representing factors is added. With 'grouprep' equals to TRUE it changes the colours of the boxplots and density plots and adds a bar on the side of the heatmap with colours representing factors. We can use the *rMLL* example again.

```
> pData(rMLL)$fakefac = rep(letters[1:4],5)
```

Here we have created fake phenoData (sample annotation) because we do not know anything about the samples, but this can be your factor of interest (time, mutant, concentration, clinical state etc...). Then we can set the 'intgroup' argument as the name of the column containing the information we want to show on the report.

```
> arrayQualityMetrics(expressionset = rMLL,
+                     outdir = "rMLL1fac",
+                     force = TRUE,
+                     intgroup = "fakefac",
+                     grouprep = TRUE)
```

With 'grouprep' being TRUE, we have the colour side bar on the heatmap and the boxplots and density plots that are coloured according to the factor of interest (here, a,b,c,d).
If we have several factors of interest, we can set 'intgroup' as a vector of column names.

```
> pData(rMLL)$fakefac2 = c(rep(LETTERS[1],10), rep(LETTERS[2],10))
> arrayQualityMetrics(expressionset = rMLL,
+                     outdir = "rMLL2fac",
+                     force = TRUE,
+                     intgroup = c("fakefac","fakefac2"),
+                     grouprep = TRUE)
```

In that case, there will be two colour side bars on the heatmap. The argument 'grouprep' being TRUE, the colours used on the boxplot and density plots are the ones corresponding to the first factor of interest given in the vector 'intgroup'.

4.2 Splitting the plots

If we have a large number of arrays, the density plots are not readable. We thus can set the argument 'split.plots' to a number of density curves to be shown on one plot.

```
> arrayQualityMetrics(expressionset = rMLL,
+                     outdir = "rMLLsp",
+                     force = TRUE,
+                     split.plots = 10)
```

This argument can be set together with 'intgroup' and 'grouprep'.

5 Extended use

Some of the quality metrics provided by the package are performed using specific information about the features of the arrays. To have more quality metrics in the report, you can add the needed information in your input object. We can use the *nCCl4* example again.

5.1 Spatial layout of the array

To plot the spatial distributions of the intensities of the arrays `arrayQualityMetrics` needs the coordinates of the spots on the chip. In the case of *AffyBatch* or *BeadLevelList*, this is automatically done without further information needed. For the other types of objects, two columns corresponding to the row and column numbers of the features are required in the *featureData*. These columns should be named "X" for rows and "Y" for columns. If the arrays are split into blocks, then the function `addXYfromGAL` should be executed prior to `arrayQualityMetrics` to convert the rows and columns of the blocks in absolute "X" and "Y" on the array. In the example of the data set *CCl4*, the coordinates of the spots are in the columns named "Row" and "Column" of the featureData (the slot of the object containing the annotation of the probes). We thus need to copy this information in columns named "X" and "Y" respectively, so that these coordinates are taken into account and the spatial distribution of the intensities can be drawn.

```
> featureData(nCCl4)$X = featureData(nCCl4)$Row
> featureData(nCCl4)$Y = featureData(nCCl4)$Column
```

5.2 Mapping of the reporters

The report can also include a study of the effect of the target mapping of the reporters. Thus a *featureData* column named "hasTarget" should include logical TRUE if the reporter matches for a coding mRNA and FALSE if not. In the *CCl4* case, probe names can be RefSeq identifiers. Thus the ones starting with "NM" are the one corresponding to coding mRNA. "hasTarget" can be derived from this, the following way:

```
> featureData(nCCl4)$hasTarget = (regexpr("^NM", featureData(nCCl4)$Name) > 0)
```

This command line produces TRUE if the probe name starts with "NM" and FALSE if it does not.

5.3 GC content of the reporters

If the GC content of the reporters is known, then it is possible to include it as percentages in the *featureData* of the *NChannelSet* under the column name "GC". Then a study of the GC content effect on intensities of the arrays can be performed. In the *CCl4* example we do not have this information. However, the process would be very similar to what is done with the "hasTarget" column.

5.4 Factor of interest

As seen before, we can make use of a factor of interest. In the case of the *CCl4* dataset, the RNA hybridized to the arrays can be of good, medium or poor quality accordingly to its RIN number (see *CCl4* vignette). We can read the sample information of *CCl4*, there is 4 RIN values possible 2.5, 5, 9 and 9.7, with 9 always corresponding to the reference. As the RIN number is given per dye, we need to create a variable which will be "Good" when one of the two dyes RIN is 9.7, "Medium" when it is 5 and "Poor" when it is 2.5. In this example, we will copy this variable in a column "RNAintegrity" of the *phenoData*. We will set the argument 'intgroup' with this name when calling the function `arrayQualityMetrics`.

```
> datapath = system.file("extdata", package="CCl4")
> p = read.AnnotatedDataFrame("samplesInfo.txt", path=datapath)
> cond = paste(p$RIN.Cy3,p$RIN.Cy5,sep="/")
> poor = grep(cond,pattern="2.5")
> medium = grep(cond,pattern="^5/|/5")
> good = grep(cond,pattern="9.7")
> cov = rep(0, length = nrow(p))
```

```
> cov[good] = "Good"
> cov[medium] = "Medium"
> cov[poor] = "Poor"
> phenoData(nCCl4)$RNAintegrity = cov
```

5.5 Report production

```
> arrayQualityMetrics(expressionset = nCCl4,
+                     outdir = "CCl4complete",
+                     force = TRUE,
+                     intgroup = "RNAintegrity",
+                     grouprep = TRUE)
```

A report named `QMreport.html` is produced in the subdirectory *CCl4*. As 'grouprep' is TRUE and 'intgroup' is "RNAintegrity", the boxplot and density plots are represented with colours depending on the "RNAintegrity" value and a side bar is drawn next to the heatmap and coloured according to this factor as well.

Session Info

```
> toLatex(sessionInfo())
```

- R version 2.8.0 (2008-10-20), `x86_64-unknown-linux-gnu`
- Locale: `LC_CTYPE=en_US.ISO-8859-1;LC_NUMERIC=C;LC_TIME=en_US.ISO-8859-1;LC_COLLATE=en_US.ISO-8`
- Base packages: base, datasets, graphics, grDevices, grid, methods, splines, stats, tools, utils
- Other packages: affy 1.17.3, affydata 1.11.3, affyio 1.5.11, affyPLM 1.13.6, ALLMLL 1.2.2, annotate 1.18.0, AnnotationDbi 1.3.12, arrayQualityMetrics 1.8.1, beadarray 1.9.0, Biobase 2.0.1, CCl4 1.0.7, DBI 0.2-4, gcrma 2.12.1, genefilter 1.20.0, geneplotter 1.18.0, hgu133acdf 1.17.0, lattice 0.17-15, latticeExtra 0.2-2, limma 2.14.5, marray 1.18.0, matchprobes 1.12.0, preprocessCore 0.99.12, RColorBrewer 1.0-1, RSQLite 0.6-8, simpleaffy 2.13.01, survival 2.34-1, vsn 3.6.0, xtable 1.5-2
- Loaded via a namespace (and not attached): KernSmooth 2.22-22