Microarray quality assessment with arrayQualityMetrics

Audrey Kauffmann, Wolfgang Huber

September 30, 2010

Contents

Introduction

The *arrayQualityMetrics* package produces, through a single function call, a comprehensive HTML report of *quality metrics* about the data set. The quality metrics are mainly on the *per-array* level, i. e. they can be used to assess the relative quality of different arrays within a data set. Some of the metrics can also be used to diagnose batch effects, and thus the quality of the overall data set.

The report can be extended to contain further diagnostics through additional arguments, and we will see examples below.

The aim of the *arrayQualityMetrics* package is to produce information that is relevant for your decision making - not, to make the decision. It will often be applied to two, somewhat distinct, use cases: (i) assessing quality of a "raw" data set, in order to get feedback on the experimental procedures that produced the data; (ii) assessing quality of a normalised data set, in order to decide whether and how to use the data set (or subsets of arrays in it) for subsequent data analysis.

Each type of microarray data (one colour, two colour, Affymetrix, Illumina...) is represented by a class of object in Bioconductor. However, the interface to the function `arrayQualityMetrics` is the same for all of them. Please consult its manual page for its argments.

When the function `arrayQualityMetrics` is finished, a report named `QMreport.html` is produced in the subdirectory specified by the function's `outdir` argument. The report contains a series of plots explained by text. For the web presentation, the plots are produced as bitmaps (`.png`). They are also linked to `.pdf` files in order to provide high resolution images e. g. for publication.

In the case of *AffyBatch* input, some Affymetrix specific plots are added to the standard report. As explained in the Section 3, other plots can be added to the standard report if some specific columns are present in the input object.

The function `arrayQualityMetrics` also produces an R object (essentially, a very big list) with all the information contained in the report, and this object can be used by downstream tools for programmatic analysis of the report.

1 Minimal use

1.1 Example on Affymetrix data

If you are working with Affymetrix chips, an *AffyBatch* object is the most appropriate to import your raw data into Bioconductor. To learn how to produce an *AffyBatch* from your data, please read the documentation of the *affy* package. Here, we use the data set *MLL.A*, an object of class *AffyBatch* which is provided in the data package *ALLMLL* for demonstration. *MLL.A* was created from CEL files using the function `ReadAffy` 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)
```

This 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 logarithm transform the intensities.

1.2 Example with one colour arrays

If you are working on one colour non Affymetrix chips, you can load your data into Bioconductor as an *ExpressionSet* object. Please see the documentation of the *Biobase* package on how to do so. Here, for demonstration, we use the *ExpressionSet* produced from *MLL.A* by calling the RMA algorithm. It contains one value (expression estimate) for each gene for each array.

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

We can then call `arrayQualityMetrics`.

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

We do not need to set `do.logtransform` as after `rma` the data are already logarithm transformed.

1.3 Example with two colour arrays

If you are working on two colour chips, you can create an *NChannelSet* to contain your data. The documentation of the *Biobase* package gives instructions on how to build a *NChannelSet*. Here, for demonstration, we use the data set `CCl4` from the data package *CCl4* and normalize it using the variance stabilization method available in the package *vsn*.

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

1.4 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 objects of the classes *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*.

2 Playing with the arguments

2.1 Factor of interest

A useful feature of `arrayQualityMetrics` is the possibility to show the results in the context of an experimental factor of interest, i. e. a categorical variable associated with the arrays such as *hybridisation date*, *treatment level* or *replicate number*. Specifying a factor does not change anything how the quality metrics are computed. By setting the argument `intgroup` to contain the names of one or multiple columns of the data object's *phenoData* slot[1], a bar on the side of the heatmap with colours representing the respective factors is added. Similarly, the colours of the boxplots and density plots reflect the levels of the first of the factors named by `intgroup`.

We use the *rMLL* example again, and create artificial columns

```
> pData(rMLL)$fact1 = rep(letters[1:4], times = 5)
> pData(rMLL)$fact2 = rep(letters[1:4], each = 5)
```

```
> arrayQualityMetrics(expressionset = rMLL,
+                     outdir = "rMLL-facts",
+                     force = TRUE,
+                     intgroup = c("fact1","fact2"))
```

3 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.

3.1 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` (please check its manual page for details) 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
```

[1]This is where Bioconductor objects store array annotation

3.2 Mapping of the reporters

The report can also include an assessment of the effect of the target mapping of the reporters. Thus a *featureData* column named `hasTarget` should include a logical `TRUE` if the reporter matches for a coding mRNA and `FALSE` if not. In the *CCl4* example, many of the feature names are RefSeq identifiers. Thus, we test the feature names whether they with ”NM”, and assume that these corresponding to coding mRNA. ”hasTarget” can be derived from this, in 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.

3.3 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.

3.4 RNA quality - RIN number

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 are 4 RIN values: 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
```

```
> arrayQualityMetrics(expressionset = nCCl4,
+                     outdir = "CCl4-RIN",
+                     force = TRUE,
+                     intgroup = "RNAintegrity")
```

A report named `QMreport.html` is produced in the subdirectory `CCl4-RIN`. Boxplots 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 2.13.0 Under development (unstable) (2010-09-29 r53067), `x86_64-apple-darwin10.4.0`
- Locale: `C`
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: ALLMLL 1.2.8, Biobase 2.9.2, CCl4 1.0.9, affy 1.27.2, affyPLM 1.25.1, arrayQualityMetrics 3.0.14, fortunes 1.4-0, gcrma 2.21.1, hgu133acdf 2.6.0, limma 3.5.21, preprocessCore 1.11.0, vsn 3.17.2
- Loaded via a namespace (and not attached): AnnotationDbi 1.11.8, Biostrings 2.17.47, DBI 0.2-5, IRanges 1.7.37, KernSmooth 2.23-4, RColorBrewer 1.0-2, RSQLite 0.9-2, SVGAnnotation 0.6-0, XML 3.1-1, affyio 1.17.4, annotate 1.27.1, beadarray 1.17.1, genefilter 1.31.2, grid 2.13.0, hwriter 1.2, lattice 0.19-11, latticeExtra 0.6-14, marray 1.27.0, simpleaffy 2.25.0, splines 2.13.0, stats4 2.13.0, survival 2.35-8, tools 2.13.0, xtable 1.5-6