arrayMagic: two-colour DNA array quality control and preprocessing

Andreas Buneß

June 3, 2004

Abstract

This document explains an efficient and automated way to load data from two colour microarray image quantification result files into R and Bioconductor. The data are normalised and several quality diagnostic plots are generated on the fly.

Workflow Overview

The workflow can be summarized as follows:

1. create and load description of the experiment
2. read image quantification result files
3. normalisation of the data
4. quality diagnostics and visualisation
5. export of the data
6. follow-up analysis

Preliminaries

To go through this document, you need to have installed R 1.9.0, the Bioconductor libraries *Biobase, limma* >=1.2.0, *vsn* >=1.4.9 and *arrayMagic* >= 1.3.7, which contains part of the lymphoma data set which is used as example (cf. http://llmpp.nih.gov/lymphoma/).

```
> library(Biobase)
> library(vsn)
> library(limma)
> library(arrayMagic)
```

Workflow

Create and load description of the experiment

The starting point is a description file which comprises the experiment. The description file must contain the file names of all image analysis quantification results and should contain all other analysis relevant information. For the example data set such a tab-deliminated file is provided as shown below:

```
        fileName  sampleid tumortype  sex slideNumber
1 lc7b047rex.DAT    CLL-13       CLL    m           1
2 lc7b048rex.DAT    CLL-13       CLL    m           2
3 lc7b069rex.DAT    CLL-52       CLL    f           3
4 lc7b070rex.DAT    CLL-39       CLL    f           4
5 lc7b019rex.DAT DLCL-0032      DLCL    f           5
6 lc7b056rex.DAT DLCL-0024      DLCL    m           6
7 lc7b057rex.DAT DLCL-0029      DLCL    m           7
8 lc7b058rex.DAT DLCL-0023      DLCL <NA>           8
```

Relevant information may be as various as probe origin, hybridisation day, slide charge, tumour type etc. Each line of the description file has to correspond to either a single hybridisation or a single channel of one hybridisation depending on the requirements of the experimental design (cf. the argument method `channelColumn` of the method `readIntensities`).

The example description file as well as the image quantification result files are accessible on your system. The access route is determined as follows.

```
> packageDirectory <- system.file(package = "arrayMagic")
> loadPath <- file.path(packageDirectory, "extdata")
> fileName <- "phenoDataLymphoma.txt"
```

The description is read and stored in a *data.frame* named `description`:

```
> description <- readpDataSlides(fileName, loadPath = loadPath)
```

Reading image quantification result files

The `fileNameColumn` of your experiment description contains the file names of the image quantification result files. All raw data are read in and stored in an instance of the class *arrayData* named `aD`.

```
> aD <- readIntensities(description, fileNameColumn = "fileName",
+     slideNameColumn = "slideNumber", loadPath = loadPath, type = "ScanAlyze")
```

There is a generic way to deal with different types of image quantification result apart from predefined types like 'ScanAlyze' and 'GenePix' (cf. the arguments `generic` and `genericOneFilePerChannel` of `readIntensities`). For 'ScanAlyze' type files the following lines do the same job.

```
> dataColumns <- c("CH1I", "CH1B", "CH2I", "CH2B")
> names(dataColumns) <- c("greenForeground", "greenBackground",
+     "redForeground", "redBackground")
> spotAnnoColumns <- c("HEADER", "SPOT", "GRID", "ROW", "COL")
> skip <- 0
> aD <- readIntensities(description, fileNameColumn = "fileName",
+     slideNameColumn = "slideNumber", loadPath = loadPath, type = "generic",
+     spotAnnoColumns = spotAnnoColumns, dataColumns = dataColumns,
+     skip = skip)
```

Normalisation of the data

In general dye, slide and hybridisation effects besides others make a calibration or normalisation step of the data inevitable. Moreover, for quality assessment not only the raw data, but also the normalised ones have to be considered. Difficulties in normalisation may indicate low quality hybridisations. The method `normalise` offers a flexible way to normalise your data with 'loess', 'vsn' or 'quantile' type normalisations. Channels, groups of hybridisations and spots can be normalised separately.

```
> eSRG <- normalise(aD, subtractBackground = TRUE, method = "vsn",
+     spotIdentifier = "SPOT")
```

The method `normalise` returns an object of class *exprSetRG* which contains the normalised data of both channels. Details on the class are given below. The experiment description information has been passed on to the object. Unique identifiers which have been supplied with the raw data image quantification result files in the column 'SPOT' have been added as 'gene'-names to the object. Here, the given unique identifiers correspond to a consecutive numbering. The annotation information can be obtained as follows.

```
> pDataSlide(eSRG)
```

```
         fileName  sampleid tumortype  sex slideNumber
1 lc7b047rex.DAT    CLL-13       CLL    m           1
2 lc7b048rex.DAT    CLL-13       CLL    m           2
3 lc7b069rex.DAT    CLL-52       CLL    f           3
4 lc7b070rex.DAT    CLL-39       CLL    f           4
5 lc7b019rex.DAT DLCL-0032      DLCL    f           5
6 lc7b056rex.DAT DLCL-0024      DLCL    m           6
7 lc7b057rex.DAT DLCL-0029      DLCL    m           7
8 lc7b058rex.DAT DLCL-0023      DLCL <NA>           8
```

```
> spotGeneNames <- geneNames(eSRG)
```

Quality diagnostics and visualisation

Several quality and hybridisation characteristics are calculated with the help of the function `qualityParameters`. All results are stored in a list object. Details are given on the help page.

```
> qPL <- qualityParameters(arrayDataObject = aD, exprSetRGObject = eSRG,
+     spotIdentifier = "SPOT", slideNameColumn = "slideNumber")
```

These quality characteristics are utilised by the function `qualityDiagnostics` which automatically generates several diagnostic plots. Examples of these plots are shown in the Figures 1, 2 and 3.

```
> qualityDiagnostics(arrayDataObject = aD, exprSetRGObject = eSRG,
+     qualityParameters = qPL, plotOutput = "pdf")
```

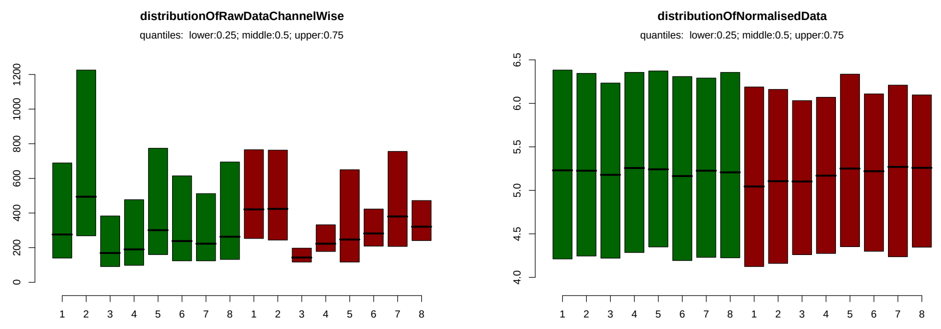

Figure 1: The plot on the left hand side characterises the distribution of the raw intensity values and on the right hand side of the normalised intensity values for each channel green and red of all hybridisations. Each box characterizes the median and range of 50% of the data, i.e. the 25, 50 and 75 percent quantiles of the data.

Spatial inhomogeneities and scratches can be detected by visual inspection of the hybridisations. The function `visualiseHybridisations` re-generates a two dimensional representation of the raw or normalised data. The output of the following code is shown in Figure 4.

```
> mappingColumns <- list(Block = "GRID", Column = "COL", Row = "ROW")
> visualiseHybridisations(arrayDataObject = aD[, c(2, 6)], slideNameColumn = "slideN
+     mappingColumns = mappingColumns)
> visualiseHybridisations(arrayDataObject = aD[, 6], exprSetRGObject = slideSubset(e
+     j = 6), type = "normalised", slideNameColumn = "slideNumber",
+     mappingColumns = mappingColumns)
```

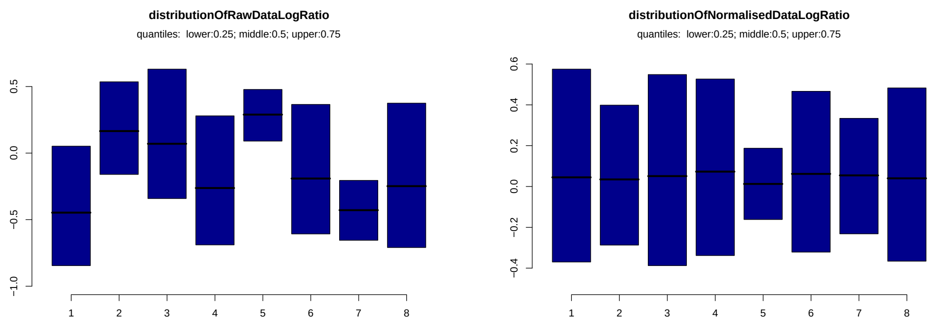

Figure 2: The plot on the left hand side characterises the distribution of the log-ratios of the non-normalised and on the right hand side of the normalised data. Each box characterizes the median and range of 50% of the data, i.e. the 25, 50 and 75 percent quantiles of the data.

Export of the data

The `writeToFile` offers a convenient way to export processed data to a well-formated tab-deliminated text file for any kind of subsequent analysis in other tools. The `channels` allows to select between the two channels and the log-ratios, as well as among raw and processed data.

```
> writeToFile(arrayDataObject = aD, exprSetRGObject = eSRG, rowSelection = 1:100,
+     slideNameColumn = "slideNumber", channels = c("logRatio"),
+     fileName = "normalisedLogRatios.txt", )
```

Follow-up analysis

A simple group comparison is presented as example for a follow-up analysis. Genes are selected which differ between the two tumour groups and which belong to the top 4000 highest intensities in the green channel. The 'difference' is calculated by means of an univariate t-test with no correction for multiple testing.

```
> myTTest <- function(x) {
+     xs <- split(x, tumortype)
+     result <- t.test(x = xs[[1]], y = xs[[2]], var.equal = TRUE)$p.value
+     return(result)
+ }
> logRatios <- getExprSetLogRatio(eSRG)
> pValues <- esApply(logRatios, 1, myTTest)
> factorTumor <- factor(pDataSlide(eSRG)$tumortype)
> pValuesFast <- rowttests(x = exprs(logRatios), fac = factorTumor)$pvalue
```

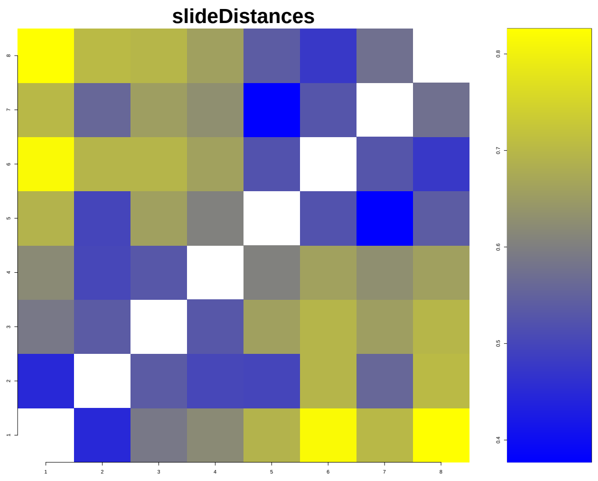

Figure 3: (Dis)similarities between all pairs of hybridisations. This may unveil potential technical artifacts like batch effects. Such global view on the data does not allow to separate technical and biological variation. Here, the blue and yellow 'blocks' characterize the two different tumor types 'CLL' and 'DLCL'. An ANOVA approach explicitly addressing known technical and biological variation, followed by a subsequent correlation analysis of the residuals may provide further insight into the data.

```
> stopifnot(all.equal.numeric(pValuesFast, pValues))
> sampleIntensities <- exprs(getExprSetGreen(eSRG))
> meanIntensity <- rowMeans(sampleIntensities)
> highestMeans <- rank(-meanIntensity) <= 4000
> selectedIndexes <- which(pValues < 1e-05 & highestMeans)
> selectedGenes <- geneNames(logRatios)[selectedIndexes]
```

The selected genes and their p-values are written to a html-file named 'result.html'.

```
> df <- data.frame(genes = I(selectedGenes), pValues = pValues[selectedIndexes])
> write.htmltable(df, "result", sortby = "pValues", decreasing = FALSE)
```

Details on the classes

All data are stored in two R classes. The class *exprSetRG* offers a convienent way to deal with normalised two colour microarray data. An instance of the class stores the normalised data for each channel. This offers the possibility to retrieve log-ratios and single channel intensities from the same instance as you might need it for filtering and analysis. It is an extension of the class *exprSet* which is part of the library *Biobase*. The class *arrayData* is a simple class for all raw data information like foreground and background intensities, spot weights, hybridisation and spot annotation information.

Important methods offered by the *exprSetRG-class* are `cbind, slideSubset, pDataSlide, getExprSetLogRatios` and by the *arrayData-class* the subset operator `[]` and `cbind`.

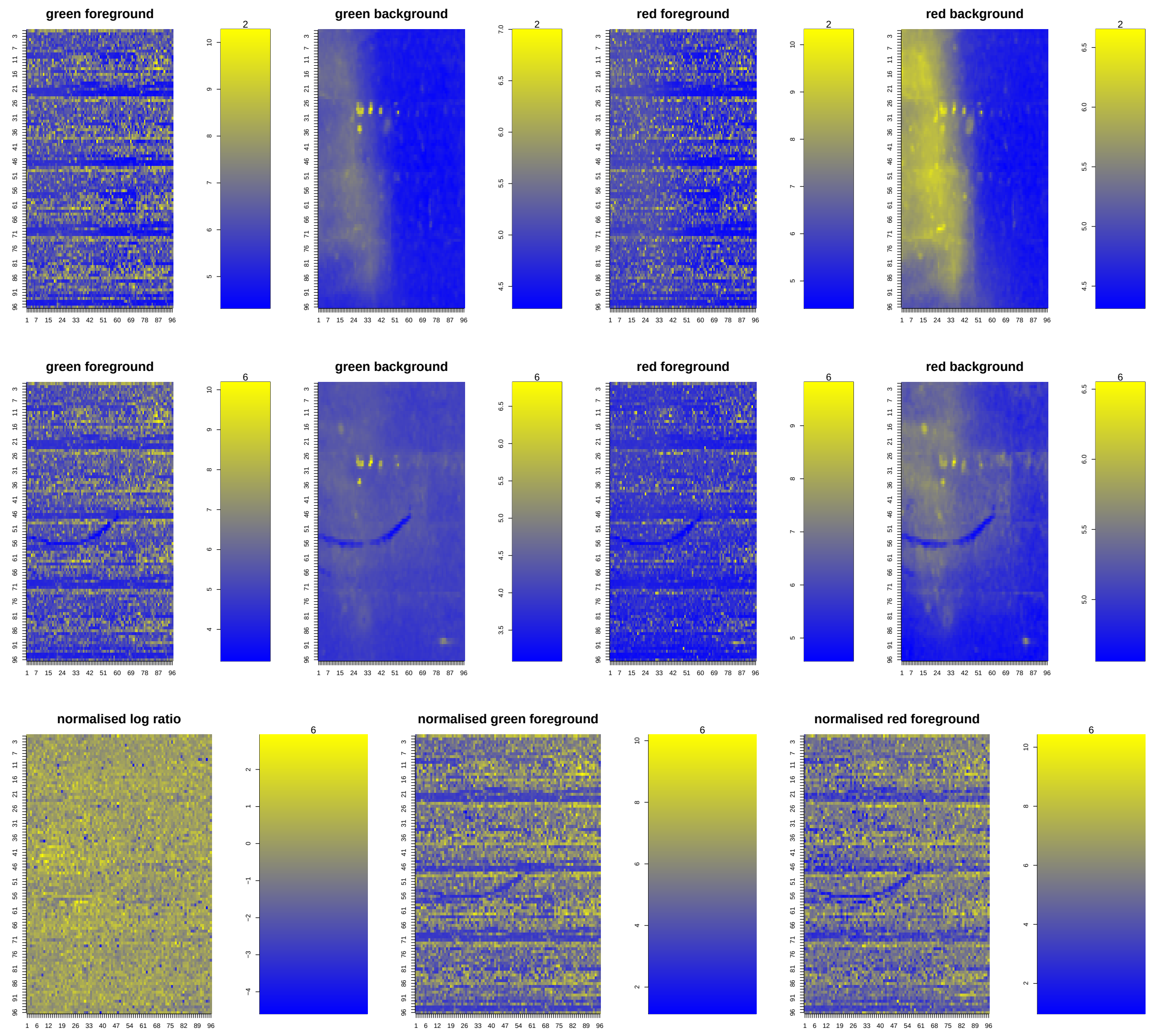

Figure 4: *Top:* visualisation of the raw data of the second hybridisation. The foreground and background of the red and green channel is shown. *Middle:* the raw data and *bottom:* normalised data of the sixth hybridisation.