arrayMagic: two-colour DNA array quality control and preprocessing

Andreas Buness

November 17, 2007

Contents

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.

Preliminaries

To go through this document, you need to have installed R, the Bioconductor libraries *Biobase, limma*, *vsn* and *arrayMagic*, 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(genefilter)
> 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
> removePatterns <- c("^REMARK")
> aD <- readIntensities(description, fileNameColumn = "fileName",
+     slideNameColumn = "slideNumber", loadPath = loadPath,
+     type = "generic", spotAnnoColumns = spotAnnoColumns,
+     dataColumns = dataColumns, removePatterns = removePatterns,
+     skip = skip)
```

Some image quantification result files require to change the `quote` argument to succeed in reading of the files; as first alternative choice you may set `quote="\""`.

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 subsets of spots, e.g. by grid or print-tip, can be normalised separately, Options should be used carefully and generally do require justification. Accordingly, background subtraction may not be applied in the first instance.

```
> 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. The quality scores given per microarray can be saved in a tab-deliminated file if the argument `resultFileName` is supplied. Details on all quality measures can be found on the help page or in the manual.

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

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")
```

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.

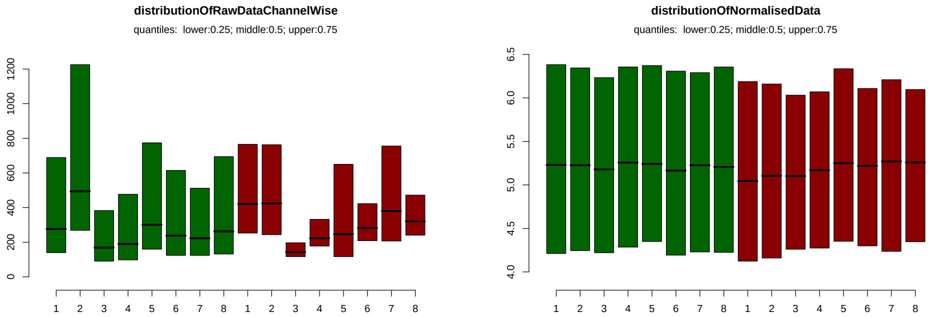

Figure 1: The plot on the left hand side characterises the distribution of the raw intensity values and on the right hand side of the normalised intensity values for each channel green and red of all hybridisations. Each box characterizes the median and range of 50% of the data, i.e. the 25, 50 and 75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

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

Additional graphical checks can be implemented by means of the R and Bioconductor environment.

A simple scatterplot is easily generated as shown below and on the left in Figure 5. The spots of the first eight subgrids on the microarray are coloured in red, all others in black. The average intensity against the log ratio is shown on the right of Figure 5.

```
> green <- exprs(getExprSetGreen(eSRG))
> red <- exprs(getExprSetRed(eSRG))
> grid <- getSpotAttr(aD)[["GRID"]]
> colour <- c("black", "red")[as.integer(grid %in% 1:8) +
+     1]
> par(mfrow = c(1, 2))
> plot(green[, 5], red[, 5], pch = ".", col = colour)
> plot(x = rowMeans(green[, 5, drop = FALSE], red[, 5,
+     drop = FALSE]), y = green[, 5] - red[, 5], pch = ".",
+     xlab = "A", ylab = "M", col = colour)
```

The esitmated density of the log ratios for each of the sixteen subgrids of the first hybridisation is shown on the left in Figure 6. Similarly, the estimated density of the first

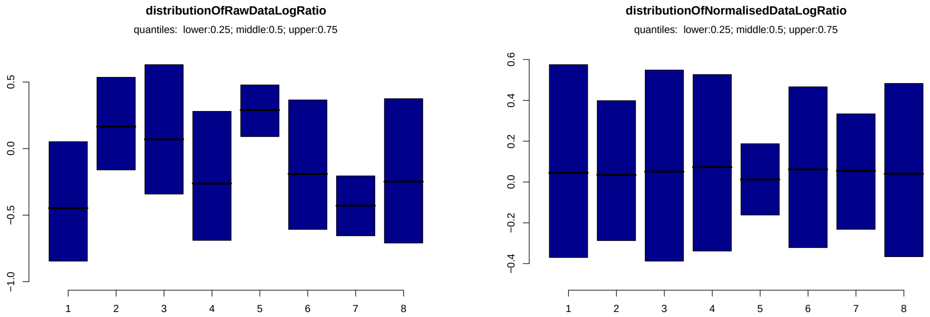

Figure 2: The plot on the left hand side characterises the distribution of the log-ratios of the non-normalised and on the right hand side of the normalised data. Each box characterizes the median and range of 50% of the data, i.e. the 25, 50 and 75 percent quantiles of the data.

subgrid for each of the eight hybridisations is shown on the right. The within and between hybridisation variability appears to be comparable.

```
> lR <- exprs(getExprSetLogRatio(eSRG))
> par(mfrow = c(1, 2))
> dHybOne <- lapply(split(lR[, 1], grid), function(x) density(x))
> dxHybOne <- cbind(sapply(dHybOne, function(x) x$x))
> dyHybOne <- cbind(sapply(dHybOne, function(x) x$y))
> matplot(dxHybOne, dyHybOne, pch = ".", ylim = c(0, 1))
> dxGridOne <- apply(lR[grid == 1, ], 2, function(x) density(x)$x)
> dyGridOne <- apply(lR[grid == 1, ], 2, function(x) density(x)$y)
> matplot(dxGridOne, dyGridOne, pch = ".", ylim = c(0,
+     1))
```

Following the same lines the distribution of the intensities before and after the normalisation can be compared separately for each channel.

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",
+     )
```

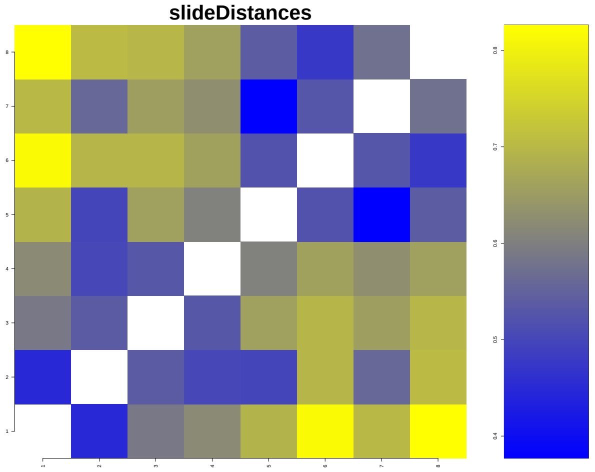

Figure 3: (Dis)similarities between all pairs of hybridisations. This may unveil potential technical artifacts like batch effects. Such global view on the data does not allow to separate technical and biological variation. Here, the blue and yellow 'blocks' characterize the two different tumour types '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.

Further Automation

The first three steps of the workflow like the reading of the image files and the normalisation are integrated in the function processArrayData. Empty spots may be specified and will excluded before or after the normalisation step. In our example the function could be called as follows:

```
> resultList <- processArrayData(slideDescriptionFile = fileName,
+     fileNameColumn = "fileName", slideNameColumn = "slideNumber",
+     loadPath = loadPath, savePath = tempdir(), type = "ScanAlyze",
+     spotIdentifier = "SPOT", spotsRemovedBeforeNormalisation = c("empty",
+         ""), subtractBackground = TRUE, normalisationMethod = "vsn",
+     plotOutput = "pdf")
```

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. The intensities of the green channel are ranked by their maximum intensity value to determine the 'highest' ones. A ranking based on the maximum intensity may for example exclude genes with low or no expression in all but one condition. The function `myTTest` is shown to demonstrate the usage of the generic function `esApply` of the package *Biobase*. The `rowttests` is part of the package *genefilter*.

```
> 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
> sampleIntensities <- exprs(getExprSetGreen(eSRG))
> maxIntensity <- apply(sampleIntensities, 1, max)
> highestOnes <- rank(-maxIntensity) <= 4000
> selectedIndexes <- which(pValues < 1e-05 & highestOnes)
> 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)
```

The corresponding log ratios for all samples are visualised in form of a 'signature' shown in Figure 7.

```
> dataColours <- c("#67001F", "#B2182B", "#D6604D", "#F4A582",
+     "#FDDBC7", "#F7F7F7", "#D1E5F0", "#92C5DE", "#4393C3",
+     "#2166AC", "#053061")
> sampleColours <- c("darkgreen", "orange")
> dataMatrix <- exprs(logRatios)[selectedIndexes, ]
> ordering <- heatmap(dataMatrix)
> heatmap(dataMatrix[ordering$rowInd, ordering$colInd],
+     col = dataColours, ColSideColors = sampleColours[as.integer(factorTumor[orderi
+     Rowv = NA, Colv = NA, scale = "none", main = "CLL vs. DLCL")
```

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*. *phenoData* or *pData* objects defined in the library *Biobase* can be obtained for each colour separately as well as for each slide, cf. `pDataSlide`. Important methods offered by the *exprSetRG-class* are `cbind, slideSubset, pDataSlide, getExprSetLogRatios`. For more details on the class and its methods call:

```
> help(exprSetRG)
```

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 *arrayData-class* are the subset operator `[]` and `cbind`.

Example application: different microarray layouts

Occasionally different types of microarrays have been used in the same experiment. For example an upgraded clone library or changing spotting layouts may result in different types of microarrays. If a unique identifier for all spots is supplied with the data files it is possible to extract a subset common to all types of microarrays. The following lines of code demonstrate how one might combine two types of microarrays. This mock example makes use of the same example data set as before. Two times the same data are combined. The approach naturally extends to more than two types. A separate description file is required for each set of microarrays belonging to the same type.

```
> SPOTIDENTIFIER <- "SPOT"
> rLOne <- processArrayData(slideDescriptionFile = fileName,
+     fileNameColumn = "fileName", slideNameColumn = "slideNumber",
+     loadPath = loadPath, savePath = tempdir(), spotIdentifier = SPOTIDENTIFIER,
+     type = "ScanAlyze", normalisationMethod = "none",
+     plotOutput = "pdf")
> rLTwo <- processArrayData(slideDescriptionFile = fileName,
+     fileNameColumn = "fileName", slideNameColumn = "slideNumber",
+     loadPath = loadPath, savePath = tempdir(), spotIdentifier = SPOTIDENTIFIER,
+     type = "ScanAlyze", normalisationMethod = "none",
+     plotOutput = "pdf")
> aDOne <- rLOne$arrayDataObject
> aDTwo <- rLTwo$arrayDataObject
> namesOne <- getSpotAttr(aDOne)[, SPOTIDENTIFIER]
> namesTwo <- getSpotAttr(aDTwo)[, SPOTIDENTIFIER]
> namesBoth <- intersect(namesOne, namesTwo)
> selectionOne <- namesOne %in% namesBoth
> selectionTwo <- namesTwo %in% namesBoth
> stopifnot(!any(duplicated(namesOne[selectionOne])))
> stopifnot(!any(duplicated(namesTwo[selectionTwo])))
> orderOne <- order(namesOne[selectionOne])
> orderTwo <- order(namesTwo[selectionTwo])
> aDOne <- aDOne[selectionOne, ][orderOne, ]
> aDTwo <- aDTwo[selectionTwo, ][orderTwo, ]
> aDBoth <- cbind(aDOne, aDTwo)
> oneNrExcluded <- sum(!selectionOne)
> if (oneNrExcluded > 0) {
+     cat(" excluded ", oneNrExcluded, " items from the first set\n")
+ }
> twoNrExcluded <- sum(!selectionTwo)
> if (twoNrExcluded > 0) {
+     cat(" excluded ", twoNrExcluded, " items from the second set\n")
+ }
> rL <- processArrayDataObject(arrayDataObject = aDBoth,
+     spotIdentifier = SPOTIDENTIFIER, normalisationMethod = "vsn",
+     subtractBackground = TRUE)
```

Example application: averaging microarrays

Technical replicates are very common in microarray analysis. One might explicitly address such situation by means of an ANOVA approach. Another approach could involve the averaging of the technical replicates. The following example explains how to average hybridisations in the framework of *arrayMagic*. The mock example above demonstrating the integration of different microarray types serves as basis for this example. Replication on different levels of the experimental hierarchy should be treated separately.

```
> eSRGTmp <- slideMerge(exprSetRGObject = rL$exprSetRGObject,
+     slideMergeColumn = "slideNumber")
> eSRGMerged <- slideMerge(exprSetRGObject = eSRGTmp, slideMergeColumn = "sampleid")
```

References

[1] Buness A., Huber W., Steiner K., Sueltmann H., Poustka A. arrayMagic: two-colour cDNA microarray quality control and preprocessing. *Bioinformatics* 2005 **21**, 554–556.

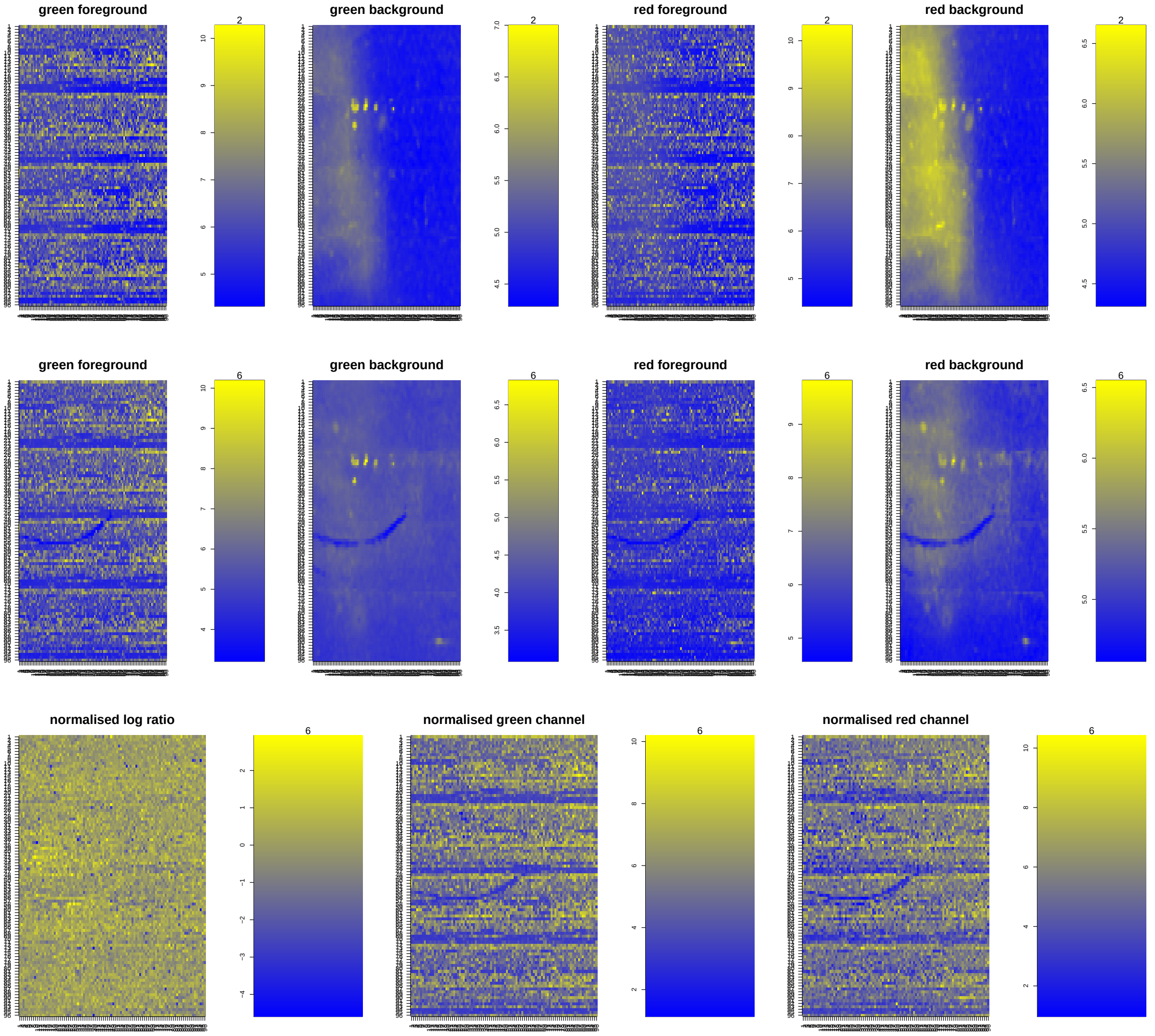

Figure 4: *Top:* visualisation of the raw data of the second hybridisation. The foreground and background of the red and green channel is shown. *Middle:* the raw data and *bottom:* normalised data of the sixth hybridisation.

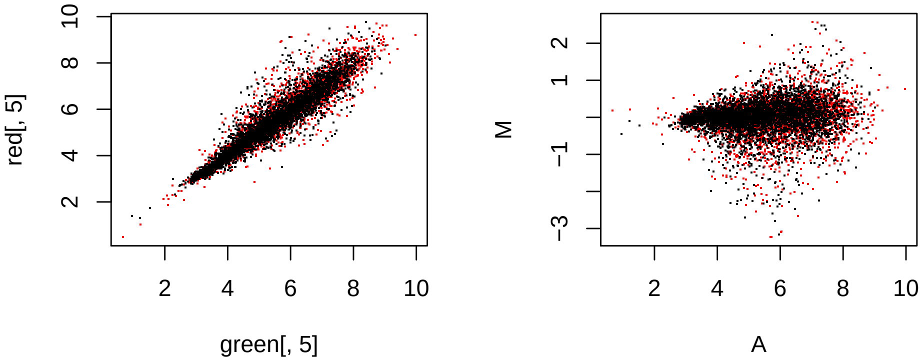

Figure 5: The green versus the red channel of the fifth hybridisation is shown on the left hand side. A different, but similar view on the same hybridisation is shown on the right hand side, i.e. the average intensity versus its log ratio. All spots belonging to the first eight subgrids are coloured in red instead of black.

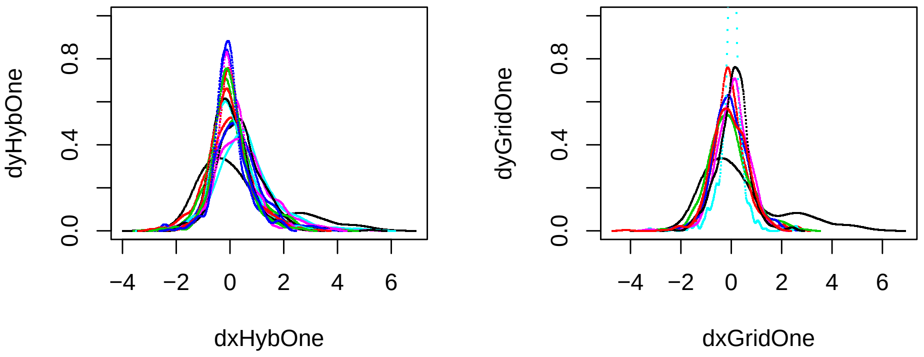

Figure 6: The graphic on the left hand side characterises the densities of log ratios for each of the sixteen subgrids of the first hybridisation. The estimated density of the first subgrid for each of the eight hybridisations is shown in the right graphic.

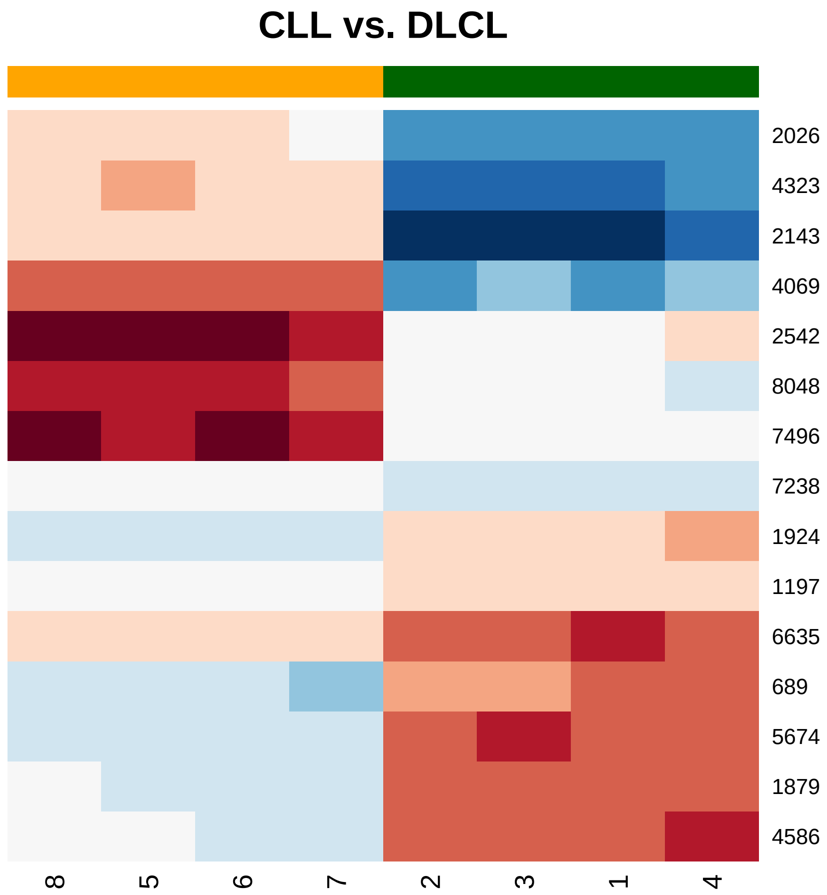

Figure 7: Log-ratios of selected genes are visualised in colours ranging from blue to white to red. Each row represents a gene and each column a sample. The tumour types are indicated by green and orange bars on the top of the graphic.