Analysis of Bead Level Data using beadarray

Mark Dunning

May 11, 2007

Introduction

beadarray is a Bioconductor package for the analysis of expression data dervied using the Illumina BeadArray platform. The package is able to analyse data generated by Illumina's BeadStudio software as well as the raw data created when arrays are scanned.

In this document we will describe how to read raw Bead Level data from a BeadArray expression array. Due to the large files generated by a BeadArray experiment, we are not currently able to offer any example data for download. Also, reading raw data into memory requires at least 1Gb of RAM at the present time. Although the example in this document is an expression array, the same procedure can be applied to methlyation or SNP raw data.

1 Citing beadarray

If you use *beadarray* for the analysis or pre-processing of BeadArray data please cite:

Dunning M, Smith M, Ritchie M, Tavaré S, *beadarray: R classes and methods for Illumina bead-based data*, R News, submitted

2 Asking for help on beadarray

Wherever possible, questions about beadarray should be sent to the Bioconductor mailing list (bioconductor@stat.math.ethz.ch). Therefore all problems and solutions will be kept in a searchable archive.

3 Import

The example in this vignette shows how to read the raw data from a Human WG6 BeadChip into R. On this chip there are 6 arrays, with each array made up 2 strips on the chip surface. The raw data consists of a tif image scanned from each strip and an text (.txt or .csv) file which describes the position and identity of each bead on each strip. These text files are required because of the random nature of BeadArrays which means we cannot rely on each position on the array having the same probe sequence attached. The tif images and txt files are produced by Illumina's BeadScan software. BeadScan version 3.1 is required with the settings.xml file in the program directory modified to include the lines `<IncludeXY>true</IncludeXY>` and `<SaveTextFiles>true</SaveTextFiles>`.

For more details see

http://www.damtp.cam.ac.uk/user/npt22

By default, `readIllumina` will read all arrays in the current working directory with both txt and tif files (for two colour experiments, both red and green images required).

The 2 strips for each array have a different set of bead types attached and at this low level of analysis, the two images can be analysed separately. The function `readIllumina` implements the image

processing steps used by Illumina. However, both the sharpening and background correction steps are optional. We estimate a background for each bead by taking the average of the 5 dimmest pixels in a local area around each bead centre. However, we do not subtract this value automatically. The same call to readIllumina will read data for two-colour SNP and methylation data as well and data from 96-well SAM experiments.The only difference would be the working directory that the command is run from.

In this example data set we have three different samples, three samples supplied by Illumina (I), three tumour samples (P) and three normals (Norm). This BeadChip is part of the same example set supplied in the BeadSummaryExamples zip file and described in the the Analysis of Bead Summary Data using beadarray vignette.

```
> library(beadarray)

Package SparseM (0.72) loaded.  To cite, see citation("SparseM")
Package quantreg (4.04) loaded.  To cite, see citation("quantreg")

> targets = read.table("targets.txt", sep = "\t", header = TRUE,
+     as.is = TRUE)
> BLData <- readIllumina(textType = ".csv", targets = targets,
+     arrayNames = targets$ArrayName)

Found 12 arrays
Reading pixels of 1475542113_A_1_Grn.tif
Calculating background
Sharpening Image
Calculating foregound
Background correcting: method = none
Reading pixels of 1475542113_A_2_Grn.tif
Calculating background
Sharpening Image
Calculating foregound
Background correcting: method = none
Reading pixels of 1475542113_B_1_Grn.tif
Calculating background
Sharpening Image
Calculating foregound
Background correcting: method = none
Reading pixels of 1475542113_B_2_Grn.tif
Calculating background
Sharpening Image
Calculating foregound
Background correcting: method = none
Reading pixels of 1475542113_C_1_Grn.tif
Calculating background
Sharpening Image
Calculating foregound
Background correcting: method = none
Reading pixels of 1475542113_C_2_Grn.tif
Calculating background
Sharpening Image
Calculating foregound
Background correcting: method = none
Reading pixels of 1475542113_D_1_Grn.tif
```

```
Calculating background
Sharpening Image
Calculating foregound
Background correcting: method = none
Reading pixels of 1475542113_D_2_Grn.tif
Calculating background
Sharpening Image
Calculating foregound
Background correcting: method = none
Reading pixels of 1475542113_E_1_Grn.tif
Calculating background
Sharpening Image
Calculating foregound
Background correcting: method = none
Reading pixels of 1475542113_E_2_Grn.tif
Calculating background
Sharpening Image
Calculating foregound
Background correcting: method = none
Reading pixels of 1475542113_F_1_Grn.tif
Calculating background
Sharpening Image
Calculating foregound
Background correcting: method = none
Reading pixels of 1475542113_F_2_Grn.tif
Calculating background
Sharpening Image
Calculating foregound
Background correcting: method = none
```

4 The BLData object

```
> is(BLData)

[1] "BeadLevelList"

> class(BLData)

[1] "BeadLevelList"
attr(,"package")
[1] "beadarray"

> slotNames(BLData)

[1] "beadData"    "phenoData"   "arrayInfo"   "beadAnno"    "scanMetrics"

> an = arrayNames(BLData)
> an

 [1] "1475542113_A_1" "1475542113_A_2" "1475542113_B_1" "1475542113_B_2"
 [5] "1475542113_C_1" "1475542113_C_2" "1475542113_D_1" "1475542113_D_2"
 [9] "1475542113_E_1" "1475542113_E_2" "1475542113_F_1" "1475542113_F_2"
```

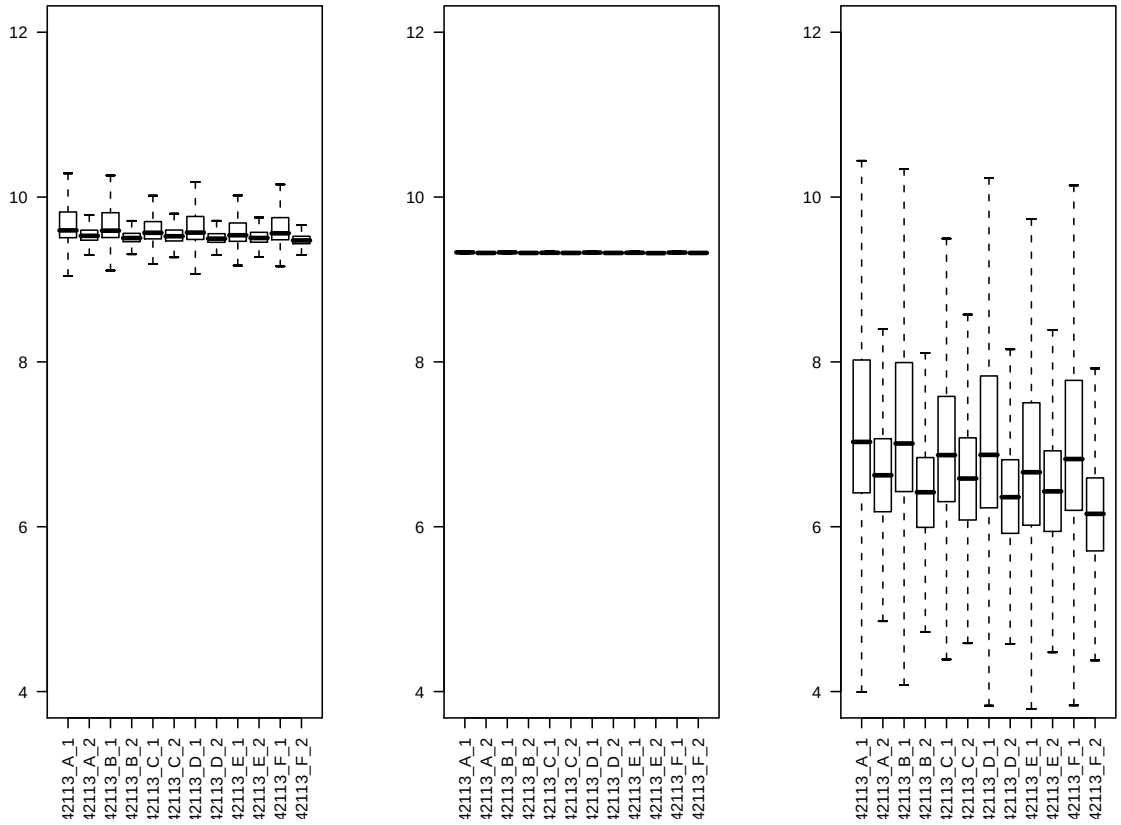
```
> names(BLData@beadData[[an[1]]])

[1] "ProbeID" "G"       "Gb"      "GrnX"    "GrnY"

> BLData[[an[1]]]$R[1:10]

NULL

> BLData[[an[2]]]$R[1:10]

NULL

> pData(BLData)

        ArrayName SampleID   Origin
1  1475542113_A_1       IC Illumina
2  1475542113_A_2       IC Illumina
3  1475542113_B_1       IH Illumina
4  1475542113_B_2       IH Illumina
5  1475542113_C_1       IC Illumina
6  1475542113_C_2       IC Illumina
7  1475542113_D_1        P   Breast
8  1475542113_D_2        P   Breast
9  1475542113_E_1        P   Breast
10 1475542113_E_2        P   Breast
11 1475542113_F_1     Norm   Normal
12 1475542113_F_2     Norm   Normal
```

The result of `readIllumina` is an object of type *BeadLevelList*. This is our recommended class for storing the raw data from both single and two colour Illumina experiments. Unlike conventional microarrays, the number of replicates of a particular probe can vary between arrays. Therefore we require a structure that allows the number of features on each array to differ. The *BeadLevelList* class contains a number of slots useful for describing Illumina data. The intensities for each array can be accessed by first subsetting the beadData slot by the name of the array and then finding the right list name.

Boxplots can be used to compare foreground and background intensities between arrays. In this example we can see very little variation between arrays. Notice that the background level appears to be virtually constant both for beads on the same array and between arrays.

Background correction can be performed by the `backgroundCorrect` and the default settings to the function subtract the background estimate for each bead from the foreground. In this example we use the minimum method which stops any negative values from being calculated.

```
> par(mfrow = c(1, 3))
> boxplotBeads(BLData, las = 2, outline = FALSE, ylim = c(4, 12))
> boxplotBeads(BLData, las = 2, whatToPlot = "Gb", outline = FALSE,
+     ylim = c(4, 12))
> BLData.bc = backgroundCorrect(BLData, method = "subtract")
> boxplotBeads(BLData.bc, las = 2, outline = FALSE, ylim = c(4,
+     12))
```



5 Bead Level Analysis

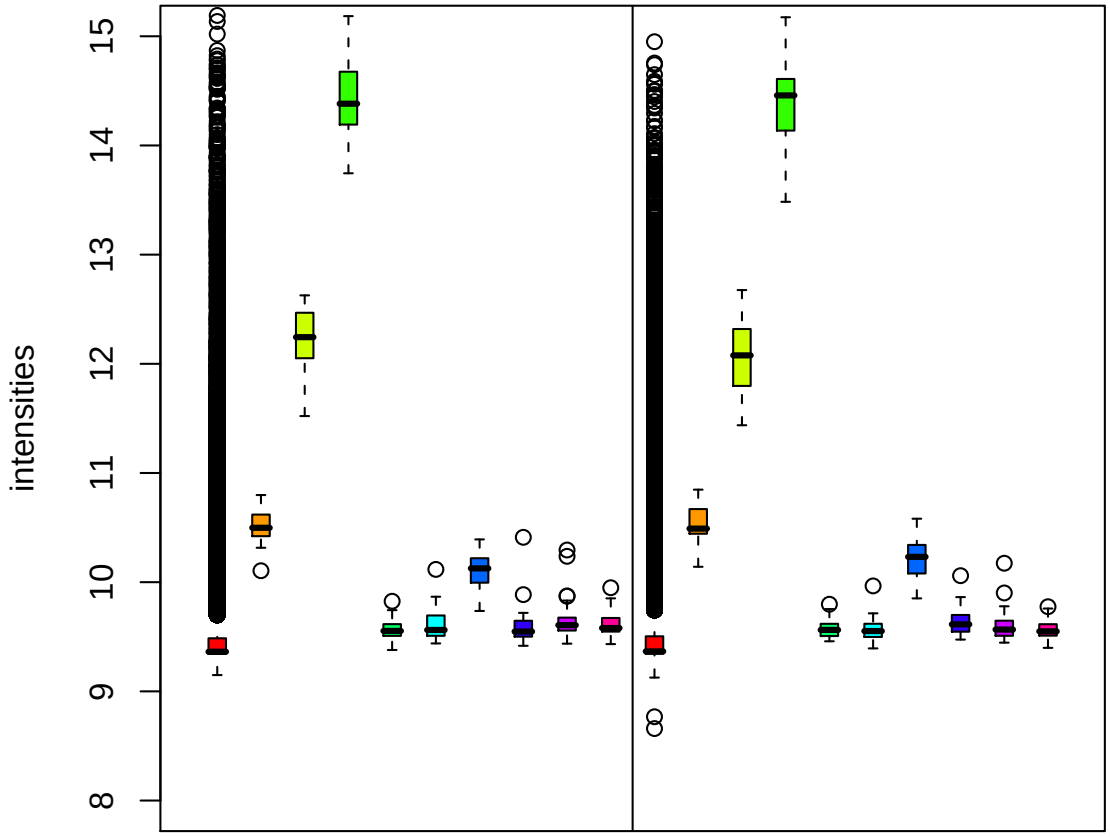
We can plot the position and location of the replicates for a particular bead type using the following code. Each BeadArray is produced using a random sampling mechanism, therefore we would expect the placement of each bead type on an array to be random.

We can also produce boxplots of bead intensities using `plotBeadIntensities`. This function takes a list of ProbeIDs and arrays as arguments and produces a boxplot for each bead type on each array grouping ProbeIDs on the same array together. Any red dots on a boxplot indicate the outliers for the bead type, these are any beads outside a 3 MAD cut-off from the mean for the bead type and are excluded from analysis. Illumina use the unlogged bead intensities for this outlier removal and this is the default option in beadarray.

In the following code we show how to plot the intensities of three different bead types on two separate arrays in the experiment. For this particular example we have to remember that all odd-numbered arrays in the experiment contain RefSeq genes whereas the even-numbered arrays contain Supplemental genes, therefore we plot the intensities of the beads on the first and third arrays.

```
> ids = unique(BLData[[an[1]]]$ProbeID)[1:10]
> ids

 [1]      0 50008 50014 50017 50020 50022 50025 50026 50035 50037

> ProbeCols = rainbow(10)
> plotBeadIntensities(BLData, arrays = c(1, 3), ProbeIDs = ids,
```

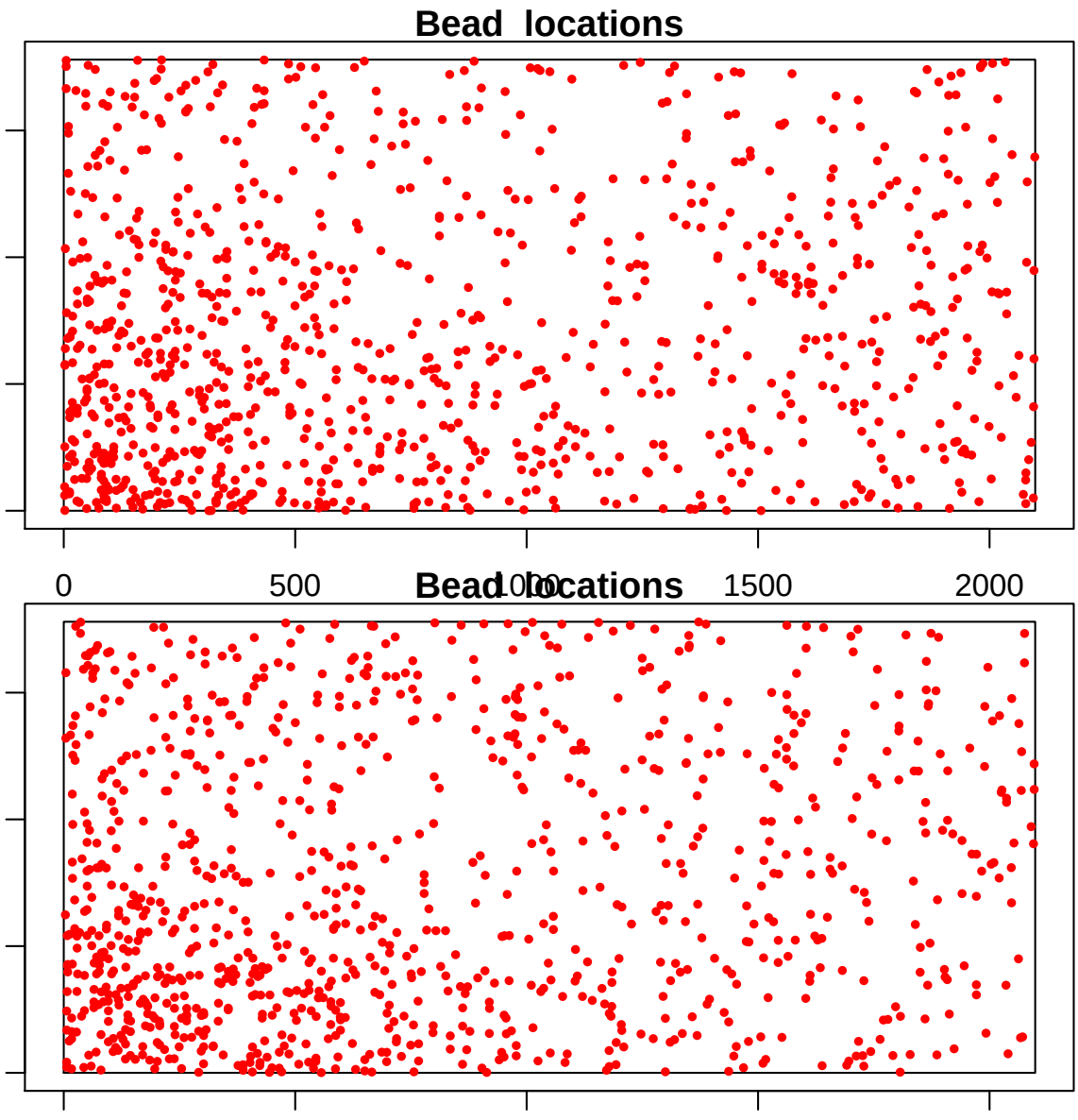
```
+     ProbeCols = ProbeCols, log = TRUE, ylim = c(8, 15))
```



We can repeat the outlier analysis shown above for all bead types on an array using `findAllOutliers`. The result of this function is a list of row indices to `BLData` to identify which beads on the given array are outliers. Typically we find that the number of outliers on an array is less than 10% and both the number and location of outliers can be used as a useful diagnostic tool.

```
> o = findAllOutliers(BLData, array = 1)
> o[1:10]

 [1] 81845 81894 81898 81956 81973 82010 82033 82037 82046 82072

> length(o)/numBeads(BLData)[1]

[1] 0.03525418
```

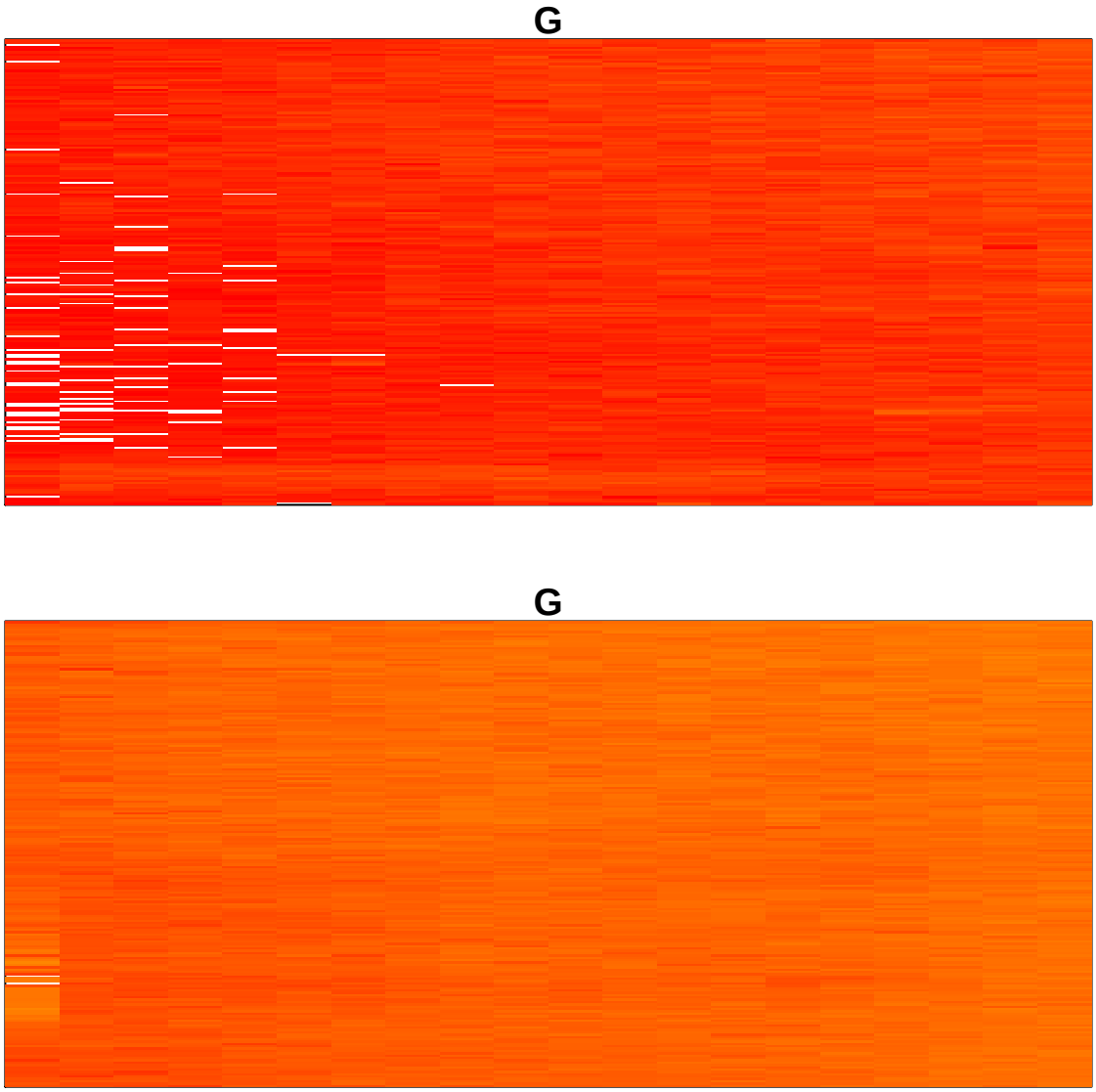
```
> par(mfrow = c(2, 1))
> par(mar = c(1, 1, 1, 1))
> for (i in 1:2) {
+     o = findAllOutliers(BLData, array = i)
+     plotBeadLocations(BLData, BeadIDs = o[1:1000], array = i,
+         SAM = FALSE, cex = 0.5, col = "red", pch = 16)
+ }
```



Arrays with a high proportion of outliers might be explained by areas of an array with unusually high background or foreground. Such regions can also be investigated by using image plots. To produce these plots we divide the array up into rectangles with a defined number of rows of columns. On the plot, the colour of the rectangle is the average of all beads lying inside that rectangle.

```
> par(mfrow = c(2, 1))
> for (i in 1:2) {
```

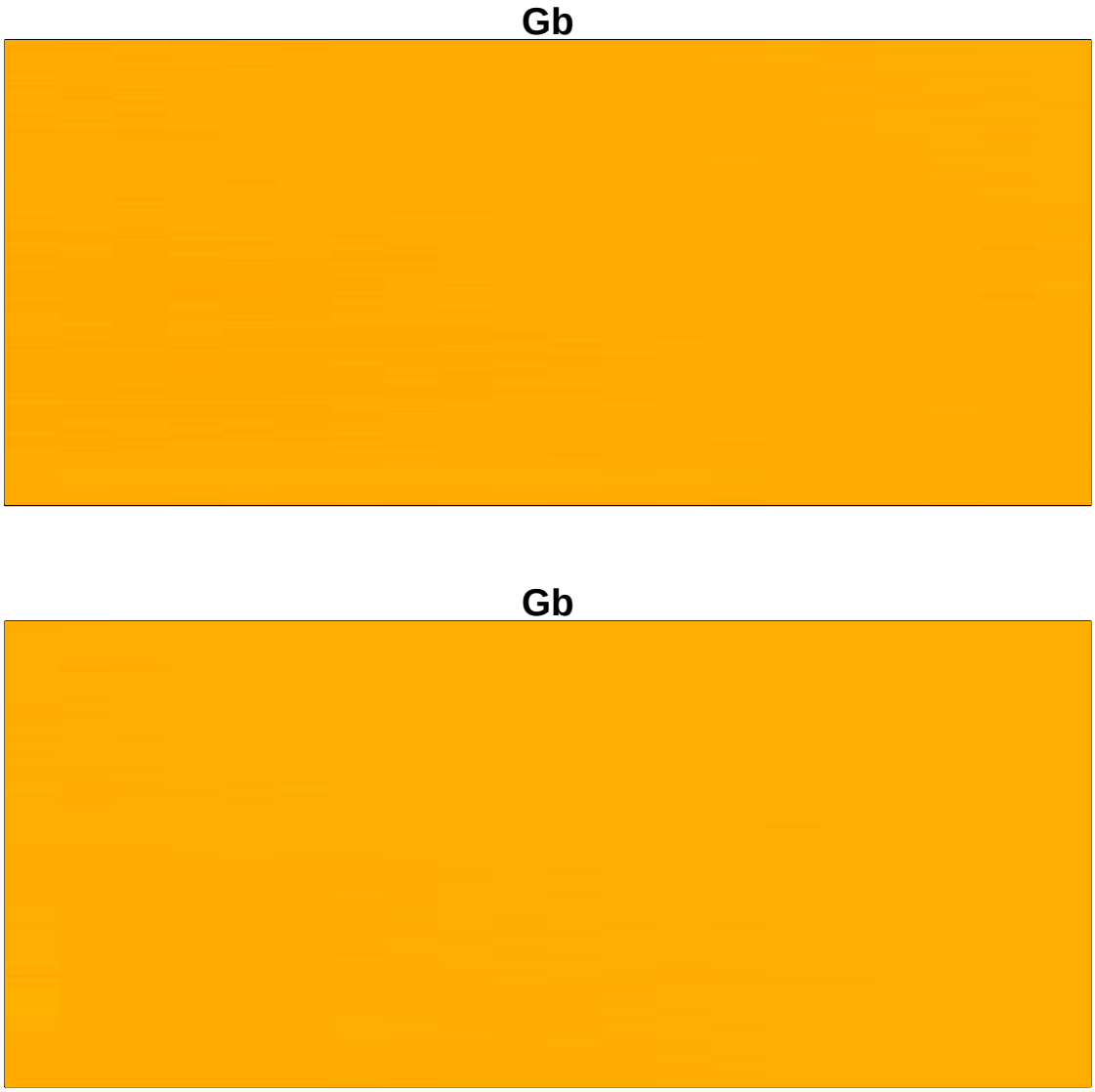
```
+     imageplot(BLData, array = i, nrow = 200, ncol = 20, zlim = range(9,
+         10), low = "yellow", high = "red", what = "G")
+ }
```



The default options for `BLImagePlot` plot average over the foreground intensities inside each rectangle. The number of rows and columns to divide the array into can be changed by using `nrow` and `ncol` parameters. By using the `whatToPlot` argument we can also plot background or values contained in any other slot in `BLData`. The colours used to plot low and high values can be changed by `low` and `high` respectively whereas `zlim` specifies which values to associate with these colours. Setting `zlim` to the same value for a series of plots allows imageplots to be compared more easily.

```
> par(mfrow = c(2, 1))
> for (i in 1:2) {
+     imageplot(BLData, array = i, nrow = 200, ncol = 20, what = "Gb",
```

```
+         low = "yellow", high = "red", zlim = range(9, 10))
+ }
```



The `createBeadSummaryData` function can be used to summarise the values for each probe. Unlike Affymetrix technoloy, each replicate of a bead contains the same probe sequence and therefore using a straight average of the replicates should be valid. Outliers are removed using a cut-off of 3 MADS and the mean of the remaining beads is used as the summary value. At this point we combine the two strips for each array by using the `imagesPerArray` argument, leaving us with 6 columns now instead of 12. The `createBeadSummaryData` function can be used to summarise the values for each probe. Outliers are removed using a cut-off of 3 MADS and the mean of the remaining beads is used as the summary value. At this point we combine the two strips for each array by using the `imagesPerArray` argument, leaving us with 6 columns now instead of 12.

```
> BSData = createBeadSummaryData(BLData, imagesPerArray = 1)
```

The default settings for `createBeadSummaryData` assume that the same probes are to be found on each array in the experiment as this will be true in general. At present, `createBeadSummaryData` is a memory intensive operation and requires at least 1Gb of RAM.

The `BSData` can be analysed using functionality described in the Analysis of Bead Summary Data vignette.