Analysis of Bead Level Data using beadarray

August 29, 2006

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 experiment. 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.

1 Citing beadarray

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

Dunning M, Smith M, Thorne NP, Tavaré, *beadarray: An R package to Analyse Illumina BeadArrays*, R News, submitted

2 Import

The following code shows how to read the raw data from a BeadChip into R. On this Chip we have 6 arrays where each array is made up 2 strips on the Chip surface. The raw data consists of a tif image scanned from each strip and an csv file which describes the position and identity of each bead on each strip. These csv 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 csv files are produced by Illumina's BeadScan software. To produce the csv files, BeadScan version 3.1 is required. For more details see

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

The function `readBeadLevelData` 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.

A targets file is required in order to define the location of the csv and tiff images. We also specify an ID for each array and the sample on that array. Note that `readBeadLevelData` can be used on data generated using either the SAM (96 well plates) or BeadChip technologies. The only difference is the way in which the targets file is defined. In the following example we are reading data from a BeadChip which consists of 12 strips with 24,000 genes on each. There are 6 samples on the chip with each 2 strips comprising each array. The 2 strips for each array have a different set of bead types attached and can therefore be analysed separately at this stage.

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 folder and described in the the Analysis of Bead Summary Data using beadarray vignette.

```
> targets <- readBeadTargets("targets.txt")
> targets
```

	ArrayID	Image1	xyInfo1	SampleID
1	1475542113_A_1	1475542113_A_1_Grn.tif	1475542113_A_1.csv	IC
2	1475542113_A_2	1475542113_A_2_Grn.tif	1475542113_A_2.csv	IC
3	1475542113_B_1	1475542113_B_1_Grn.tif	1475542113_B_1.csv	IH
4	1475542113_B_2	1475542113_B_2_Grn.tif	1475542113_B_2.csv	IH
5	1475542113_C_1	1475542113_C_1_Grn.tif	1475542113_C_1.csv	IC
6	1475542113_C_2	1475542113_C_2_Grn.tif	1475542113_C_2.csv	IC
7	1475542113_D_1	1475542113_D_1_Grn.tif	1475542113_D_1.csv	P
8	1475542113_D_2	1475542113_D_2_Grn.tif	1475542113_D_2.csv	P
9	1475542113_E_1	1475542113_E_1_Grn.tif	1475542113_E_1.csv	P
10	1475542113_E_2	1475542113_E_2_Grn.tif	1475542113_E_2.csv	P
11	1475542113_F_1	1475542113_F_1_Grn.tif	1475542113_F_1.csv	Norm
12	1475542113_F_2	1475542113_F_2_Grn.tif	1475542113_F_2.csv	Norm

```
> BLData <- readBeadImages(targets)
```

3 The BLData object

BLData is a type of list object and similar to the *RGList* objects found in limma. The list contains a number of matrices which hold information for each bead over all strips on the BeadChip. Note that due to random placement of beads each row will not always correspond to the same gene, unlike the *RGList* objects of limma, therefore we have to store ProbeID which gives the identity of each bead.

The matrices in BLData can be subset in the usual way.

```
> is(BLData)
```

```
[1] "BeadLevelList"   "list"            "LargeDataObject" "vector"
```

```
> names(BLData)
```

```
[1] "G"                 "Gb"                "GrnX"              "GrnY"
[5] "ProbeID"           "targets"           "backgroundSize"    "normalised"
[9] "backgroundCorrected"
```

```
> dim(BLData$R)
```

```
NULL
```

```
> BLData$R[1:5, ]
```

```
NULL
```

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.

```
> par(mfrow = c(1, 2))
> boxplot(as.data.frame(log2(BLData$G[1:10000, ])), las = 2, names = as.character(targets[,
+     4]))
> boxplot(as.data.frame(log2(BLData$Gb[1:10000, ])), las = 2, names = as.character(targets[,
+     4]))
```
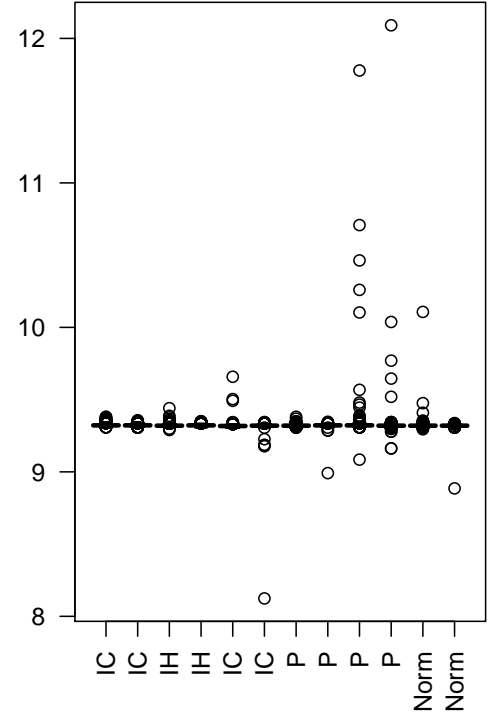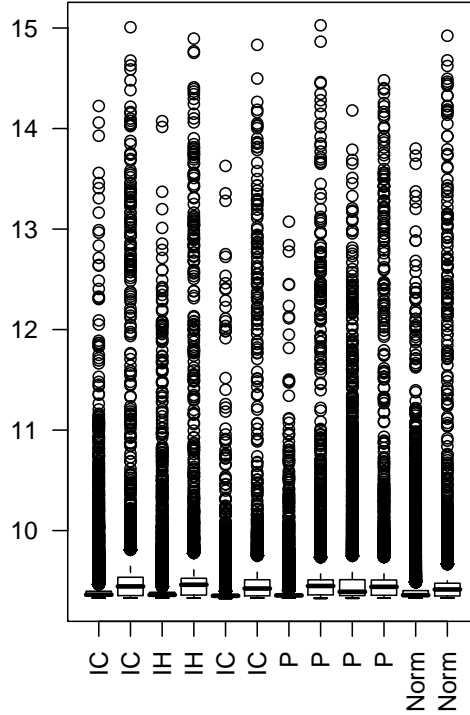
4 Bead Level Analysis

We can plot the position and location of the replicates for a particular bead type using the following code. When bead intensities are plotted we show the mean of the intensities and lines 3 MAD (median absolute deviation) above and below the mean. Any beads outside this 3 MAD cut-off are classed as outliers for a particular bead type and excluded from analysis. Illumina use the unlogged bead intensities for this outlier removal and this is the default option in beadarray. The $\log_2$ intensities can be used by setting the log argument along with the number of MADs used for the cut-off.

```
> par(mfrow = c(1, 2))
> plotBeadLocations(BLData, array = 1, ProbeIDs = 50020, SAM = FALSE)
> plotBeadIntensities(BLData, array = 1, ProbeIDs = 50020, log = FALSE, n = 3)
```
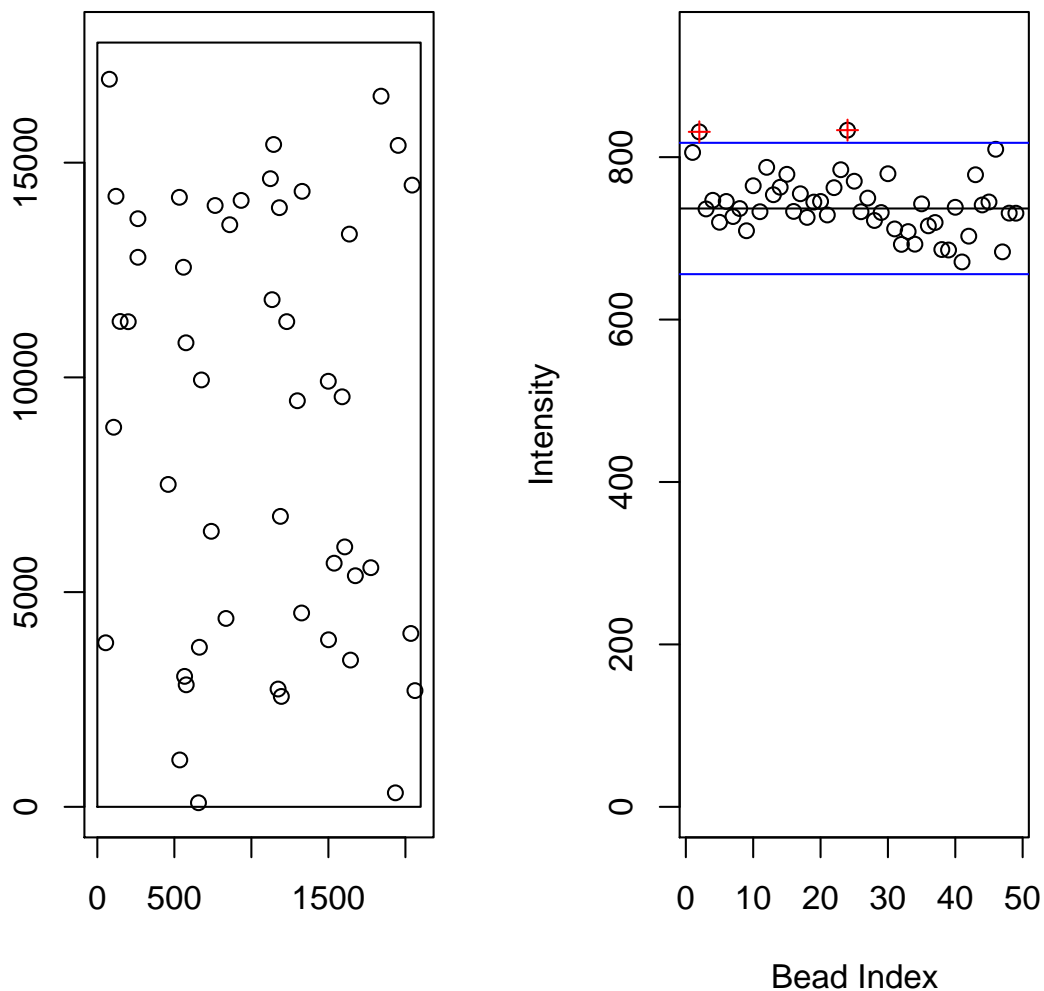
We can repeat the outlier analysis shown above for all bead types on an array using `findAllOutliers`. The result of this function is a list of row indices to identify which beads on the given array are outliers. Typically we find that the number of outliers on an array is less than 10% and both the number and location of outliers can be used as a useful diagnostic tool.

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

 [1] 81823 81845 81889 81894 81898 81956 81973 82010 82033 82037

> length(o)/nrow(BLData$G)

[1] 0.03751723

> 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 = 0.1)
+ }
```
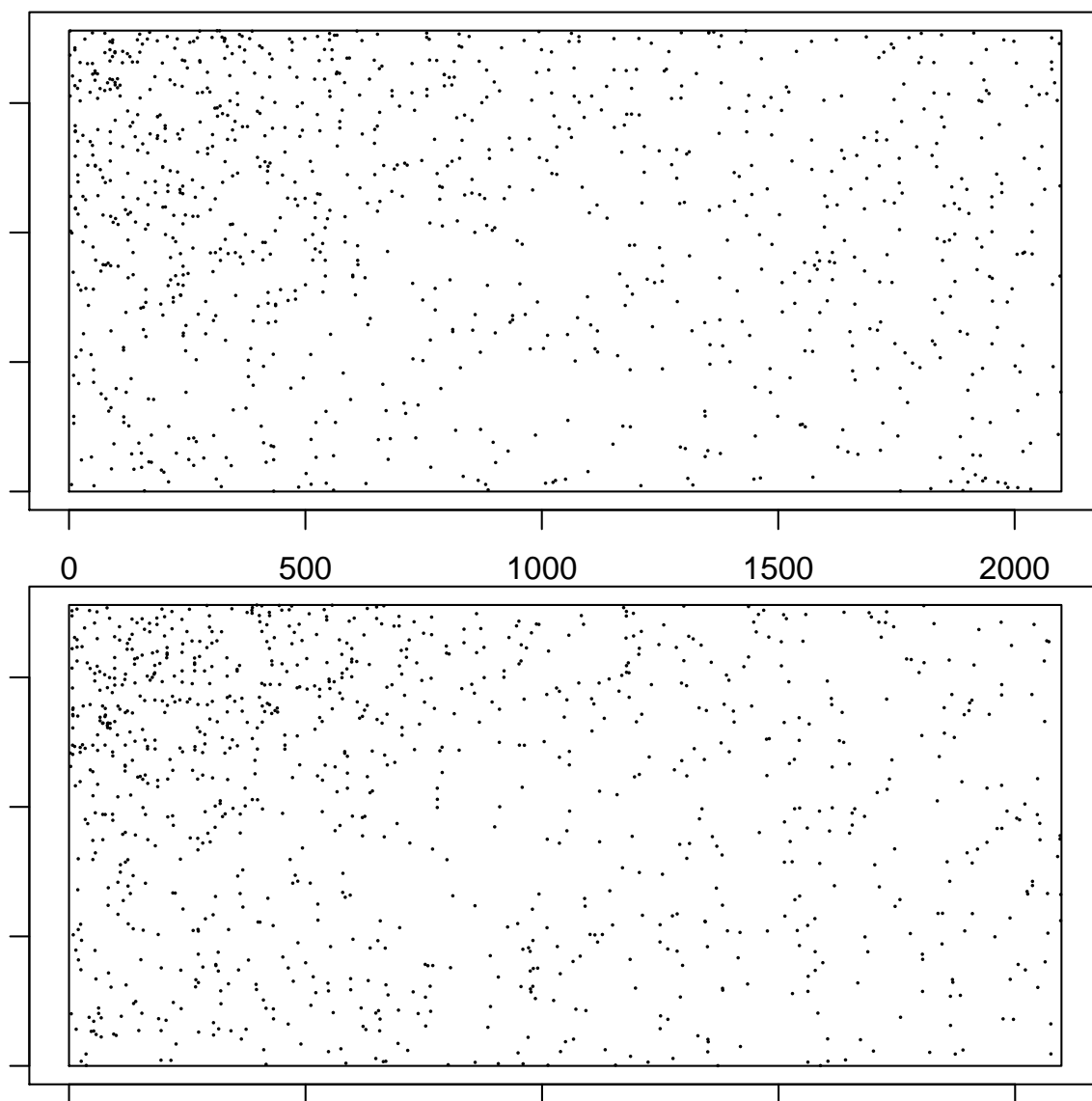

Arrays with a high proportion of outliers could be explained by areas of an array with unusually high background or foreground. Such regions can also be investigated by using image plots. To produce these plots we divide the array up into rectangles with a defined number of rows of columns. On the plot, 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) {
+     BLImagePlot(BLData, array = i, nrow = 200, ncol = 20)
+ }
```
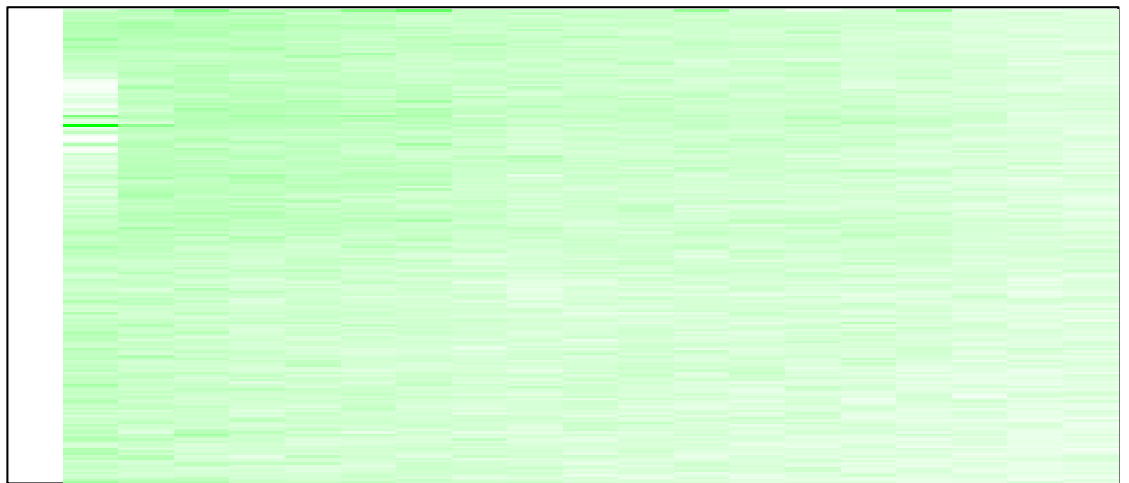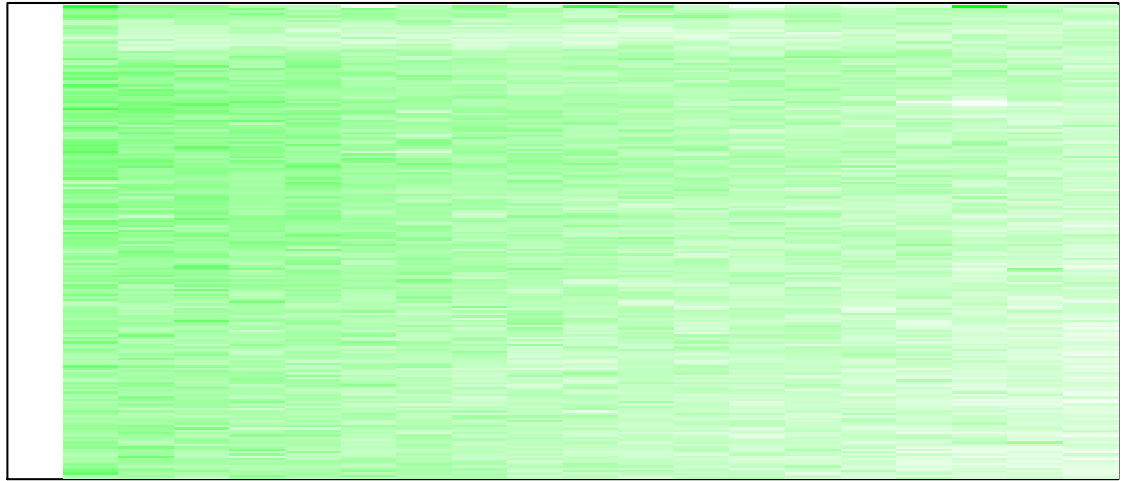
The default for options for `BLImagePlot` plot average over the foreground intensities inside each rectangle. By using the `whatToPlot` argument we can also plot background or values contained in each other matrix in `BLData`. The colours for low and high intensity spots can also be changed.

```
> par(mfrow = c(2, 1))
> for (i in 1:2) {
+     BLImagePlot(BLData, array = i, nrow = 200, ncol = 20, what = "Gb", low = "yellow",
+         high = "red")
+ }
```
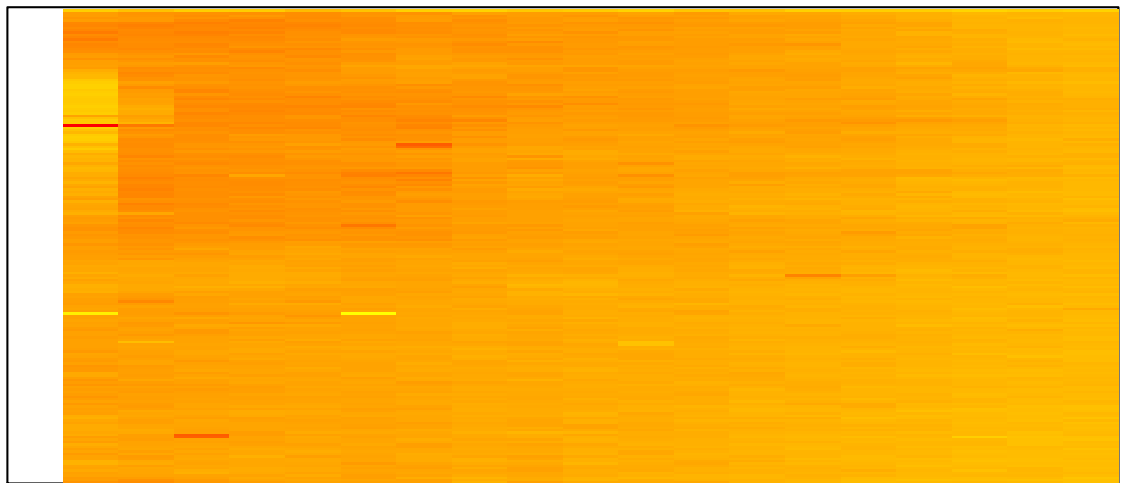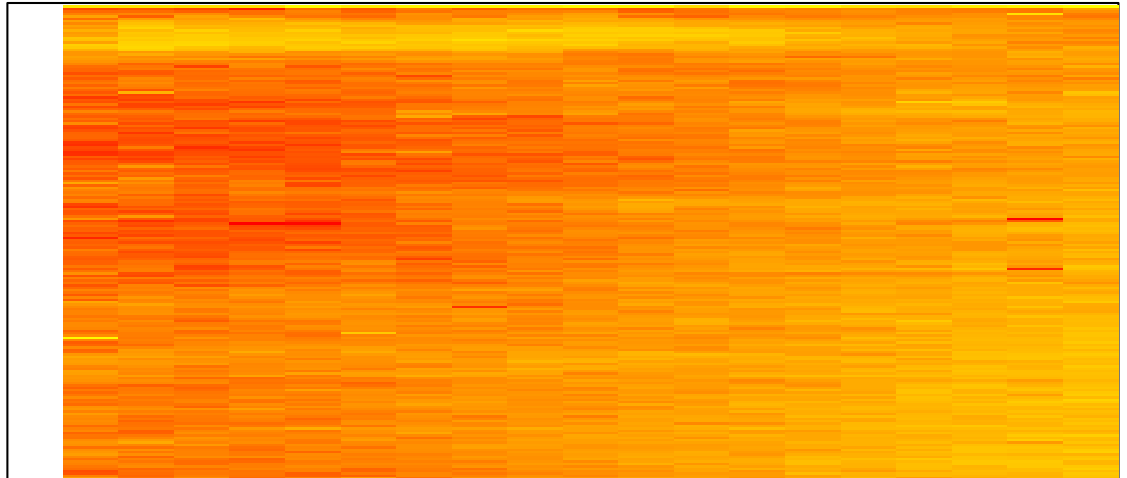
The `createBeadSummaryData` function can be used to summarise the values for each probe. Outliers are removed using a cut-off of 3 MADS and the mean of the remaining beads is used as the summary value. At this point we combine the two strips for each array, leaving us with 6 columns now instead of 12.

```
> BSData = createBeadSummaryData(BLData)
```

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.