Analysis of Bead Level Data using beadarray

Mark Dunning

September 27, 2007

Introduction

beadarray is a Bioconductor package for the analysis of 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. Although the example in this document is for a single-channel expression array, the same procedure can be applied to methlyation or SNP data.

The example data used in this vignette (BeadLevelExample.zip) can be obtained as a 800 MB zip archive at:

http://criwebshare.cancerresearchuk.org/IlluminaData/BeadLevelExample.zip

1 Citing beadarray

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

Dunning MJ, Smith ML, Ritchie ME, Tavaré S, **beadarray: R classes and methods for Illumina bead-based data**, *Bioinformatics*, **23**(16):2183-2184

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. When posting to this mailing list, please state the version of beadarray and `R` to help to diagnose the problem. This can be done by pasting the output of running the command `sessionInfo()`.

3 Import

The example in this vignette shows how to read the raw data from a Human-6 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 a 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://criwebshare.cancerresearchuk.org/IlluminaResources.html

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 are required).

The 2 strips for each array have a different set of bead types attached and images from each strip 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 Sentrix Array Matrix (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), four tumour samples (P) and two 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. A targets text file can be used to define which samples have been hybridised to each array.

The function can also read in the metrics.txt file that is created by BeadScan. This file can be useful for quality control purposes

```
> library(beadarray)
> targets = read.table("targets.txt", sep = "\t", header = TRUE,
+     as.is = TRUE)
> targets
> BLData <- readIllumina(textType = ".csv", backgroundMethod = "none",
+     targets = targets, arrayNames = targets$ArrayName, metrics = TRUE)
```

4 The BLData object

Once imported, the bead level data is stored in a *BeadLevelList* object. This class can handle raw data from both single channel and two-colour BeadArrays. Due to the random nature of the technology, each array generally has a variable number of rows of intensity data, and we use an R environment variable to store this information in a memory efficient way.

The *BeadLevelList* class contains as 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 correct list name. Alternatively, `getArrayData` can be used.

```
> is(BLData)

[1] "BeadLevelList"

> class(BLData)

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

> slotNames(BLData)

[1] "beadData"    "phenoData"   "arrayInfo"   "annotation"  "beadAnno"
[6] "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]]]$G[1:10]
```

```
 [1]  647.1598 1291.8708 4646.7958  994.2587  716.0407  647.0293  646.6438
 [8]  654.3582  659.4786  816.0154
```

```
> BLData[[an[2]]]$Gb[1:10]
```

```
 [1] 636 634 635 637 639 636 637 637 636 637
```

```
> 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
```

```
> getArrayData(BLData, array = 1, what = "G", log = FALSE)[1:10]
```

```
 [1]  647.1598 1291.8708 4646.7958  994.2587  716.0407  647.0293  646.6438
 [8]  654.3582  659.4786  816.0154
```

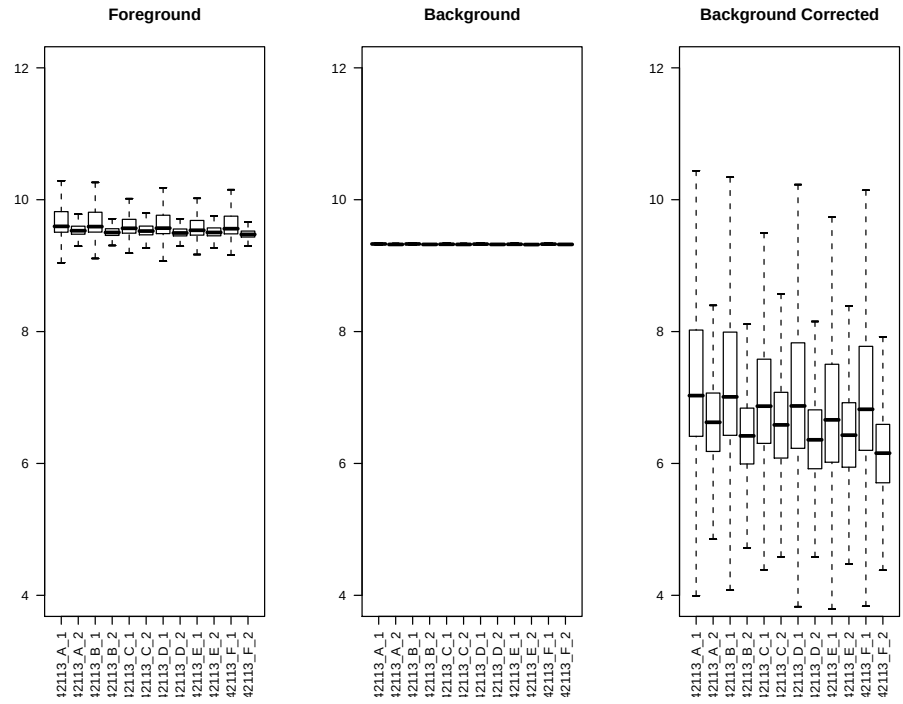
```
> getArrayData(BLData, array = 2, what = "Gb", log = FALSE)[1:10]
```

```
 [1] 636 634 635 637 639 636 637 637 636 637
```

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` function and the default settings of the function subtract the background estimate for each bead from the foreground.

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



The `whatToPlot` argument of `boxplotBeads` controls which intensities are plotted for each bead. Options are `G`, `Gb` and `residG` (Cy3 residuals) for single channel data and `R`, `Rb`, `residR`, `M` (log-ratios) `residM` or `A` (average log-intensities) for two-colour data.

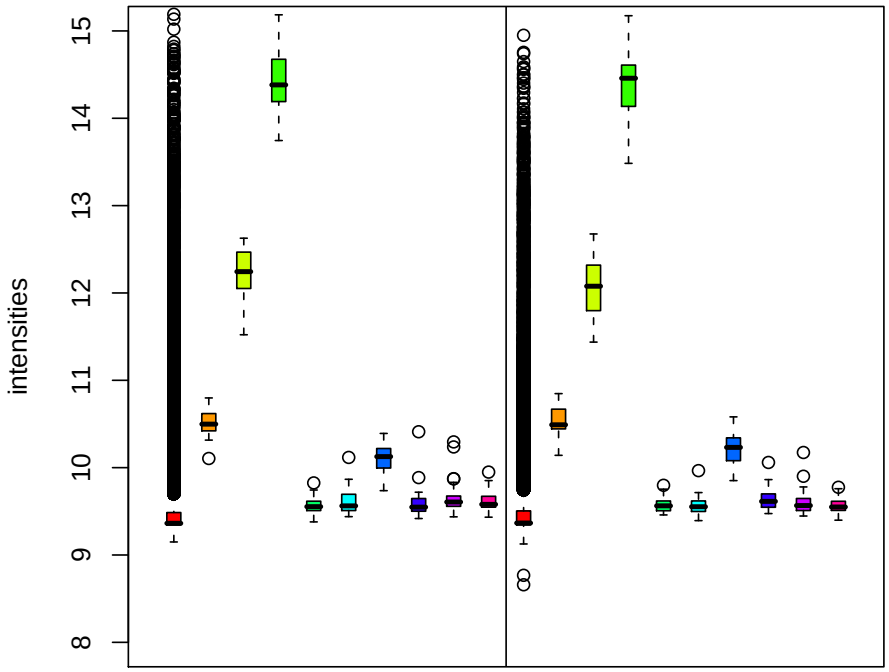
5 Bead Level Analysis

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 median absolute deviation 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.

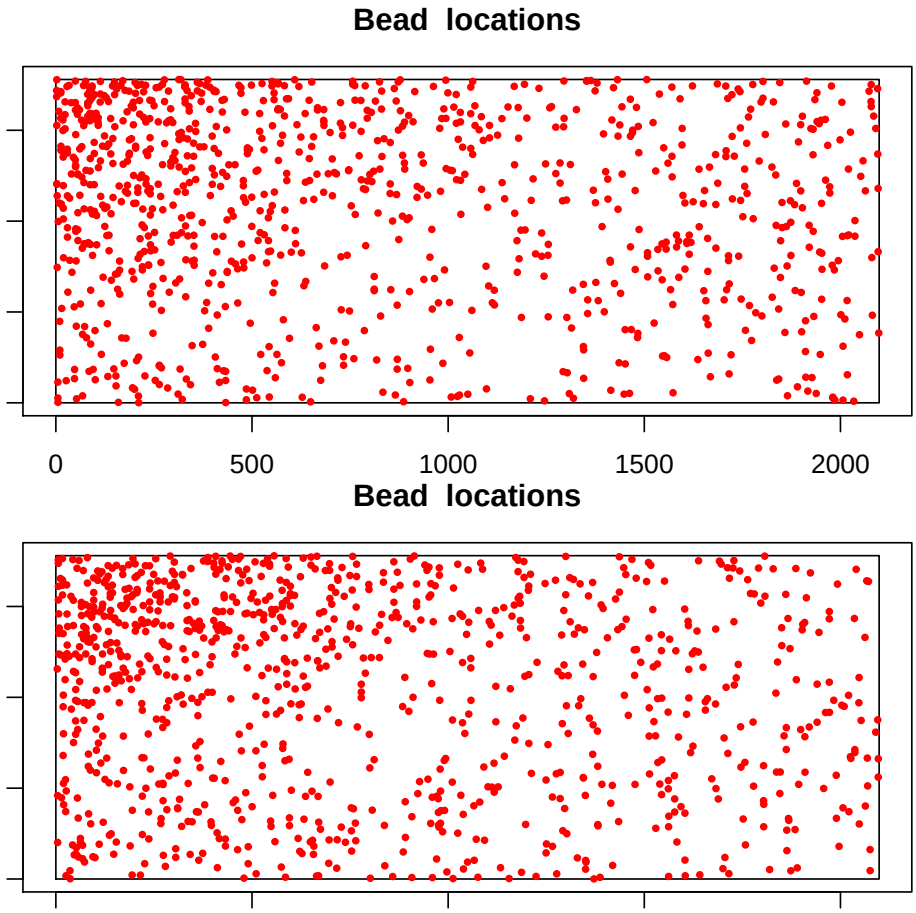
In the following code we show how to plot the intensities of three different bead types on two separate arrays in the experiment. For this particular example we have to remember that all odd-numbered arrays in the experiment contain RefSeq genes whereas the even-numbered arrays contain Supplemental genes,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



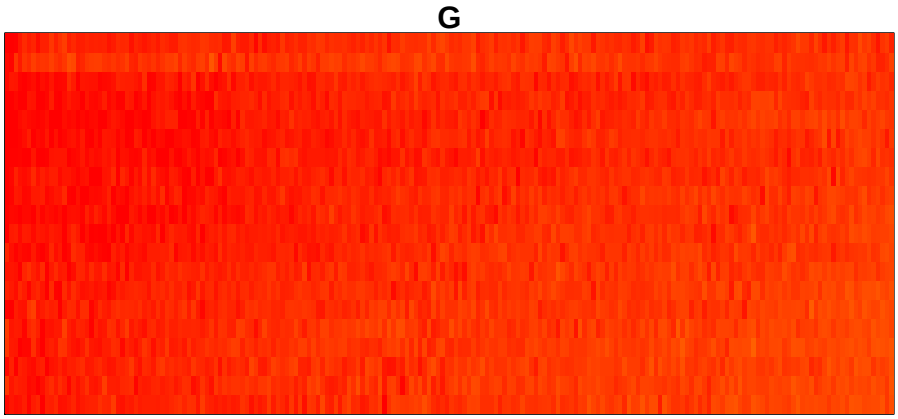
We can repeat the outlier analysis shown above for all bead types on an array using `findAllOutliers`. The result of this function is a list of row indices to `BLData` to identify which beads on the given array are outliers. Typically we find that the number of outliers on an array is less than 10% and 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, 3, 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)
+ }
```

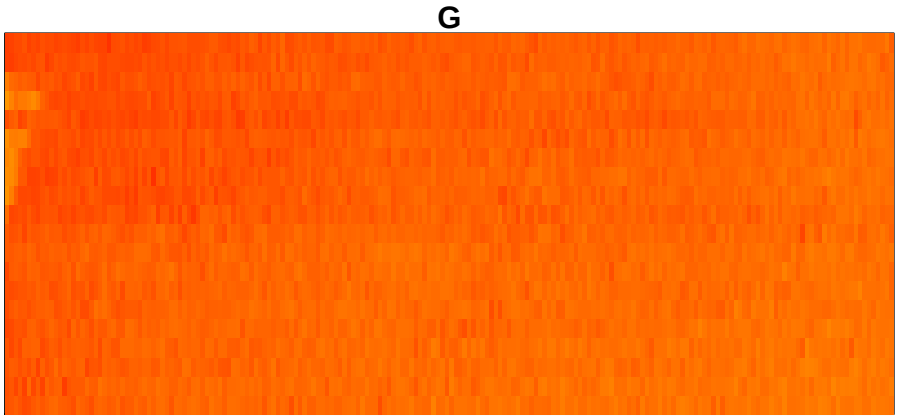


Spatial artefacts on the array surface can occur from mis-handling or scanning problems. Image plots can be used to identify these artefacts, and with the raw bead-level data, we can plot false images of each array. This kind of visualisation is not possible when using the summarised BeadStudio output, as the summary values are averaged over spatial positions. Image plots in R are also more convenient than scrutinising the original tiffs, as multiple arrays can be visualised on the one page.

```
> par(mfrow = c(2, 1))
> for (i in 1:2) {
+     imageplot(BLData, array = i, nrow = 20, ncol = 200, zlim = range(9,
+         10), low = "yellow", high = "red", what = "G")
+ }

Warning: 2 NA, NaN or Inf values, which will be ignored.
Check your data or try setting log="FALSE"
```



z−range 9.6 to 10.1 (saturation 9, 10)



z−range 9.4 to 9.8 (saturation 9, 10)

In the `imageplot` function, the argument `whatToPlot` is used to choose the quantity to display. For single channel data, `whatToPlot` can be set to `G`, `Gb` or `residG` to plot the Cy3 foreground, background or foreground residuals respectively. For two-colour data, the Cy5 foreground (`R`), background (`Rb`), log-ratios (`M`), average log-intensities (`A`) and residuals
(`residR`, `residM`) can be plotted by changing `whatToPlot`. Because of the high number of beads on each array, `imageplot` function maps a grid of size specified by the `nrow` and `ncol` arguments onto the array surface and averages the intensities of the beads within each section of the grid.

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. Alternative methods for removing outliers, such as using a trimmed mean or median can be used by changing the `method` argument to `createBeadSummarData`.

By default, we summarise the values for the green channel. In the case of two-colour data, one may wish to create summary values for the red and green channels separately or summarise the log-ratios for each bead. These can be achieved by setting the `what` argument to `RG` or `M` respectively.

```
> BSData = createBeadSummaryData(BLData, imagesPerArray = 2)
```

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 currently requires at least 1Gb of RAM for BeadChip data.

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

This vignette was built using the following packages:

```
> sessionInfo()

R version 2.6.0 Under development (unstable) (2007-08-12 r42483)
x86_64-unknown-linux-gnu

locale:
LC_CTYPE=en_GB.UTF-8;LC_NUMERIC=C;LC_TIME=en_GB.UTF-8;LC_COLLATE=en_GB.UTF-8;LC_MONETARY=en_GB.UTF-8;L

attached base packages:
[1] tools     stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
[1] beadarray_1.5.14       affy_1.15.7            preprocessCore_0.99.12
[4] affyio_1.3.2           geneplotter_1.15.6     lattice_0.14-16
[7] annotate_1.13.4        Biobase_1.15.26        limma_2.11.9

loaded via a namespace (and not attached):
[1] grid_2.6.0         KernSmooth_2.22-19 RColorBrewer_0.2-3
```


Acknowledgements

We are grateful to Inma Spiteri for providing the BeadChip data set distributed for this vignette. We also thank Julie Addison, Tom Hardcastle, John Marioni, Inma Spiteri and Anna Git for their helpful feedback on the exercises in this tutorial.