Analysis of bead-level data using beadarray

Mark Dunning and Matt Ritchie

October 16, 2008

Introduction

Illumina have created an alternative microarray technology (BeadArray) based on randomly arranged beads. A specific oligonucleotide sequence is assigned to each *bead type*, which is replicated many times (typically ~ 30) on an array. A series of decoding hybridisations is used to identify each bead on the array. The high degree of replication makes robust measurements for each bead type possible.

BeadArrays are used in many applications, including gene expression studies, SNP genotyping and methylation profiling and are processed in parallel as a Sentrix Array Matrix (SAM) or BeadChip. A SAM is a plate of 96 uniquely prepared hexagonal BeadArrays, each of which contains around 1,500 bead types. The BeadChip technology comprises a series of rectangular strips on a slide with each strip containing about 24,000 bead types. For example, there are six pairs of strips on each Human-6 BeadChip. Depending on the particular assay used, the data from a BeadArray may be single channel or two-colour.

The data from Illumina BeadArrays is available in different formats. In this guide we describe how to read raw *bead-level data* from an expression array. Although this example is from a single-channel technology, the same procedure can be applied to data from two-colour platforms.

The second format is produced by Illumina's BeadStudio software. We refer to this output as *bead-summary data* as these files contain summary intensities for each bead type on each array. For details on how to use the beadarray package to read in and process data in this format, refer to the bead-summary user's guide which can be launched with the following command

```
> library(beadarray)
> beadarrayUsersGuide(topic = "beadsummary")
```

1 Obtaining bead-level data

The raw images and text files required to perform a bead-level analysis are produced by Illumina's BeadScan software. To modify BeadScan's default settings to obtain bead-level data, edit the `settings.xml` file in the main BeadScan directory to include the lines

```
<SaveTextFiles>true</SaveTextFiles>
<ExcludeOutliers>false</ExcludeOutliers>
```
and
`<IncludeXY>true</IncludeXY>`.

For further details see the instructions at `http://www.compbio.group.cam.ac.uk/Resources/illumina/`

2 Importing bead-level data

The example data used in this guide (`BeadLevelExample.zip`, 800MB) can be downloaded from

`http://www.compbio.group.cam.ac.uk/Resources/illumina/BeadLevelExample.zip`

These arrays were part of a pilot project using Human-6 version 1 BeadChips. In this example data set we have 4 different samples, two of which were supplied by Illumina (IC and IH), and two from cell lines (P and Norm). This BeadChip is a subset of the experiment analysed in the bead-summary user's guide.

2.1 Description of files

To read in bead-level data using `beadarray` you will need several files produced by Illumina's BeadScan software. We briefly describe these files below.

- text files (*required*) - a `.txt` or `.csv` file for each strip or hexagon which stores the position, identity and intensity (after background correction) of each bead. These files are usually named `ChipNumber_Array_Strip.csv` (e.g. `1475542113_A_1.csv`) and are required because of the random arrangement of probes on the array surface, which is unique for each BeadArray.
- TIFFs (*optional*) - 1 (single channel) or 2 (two-colour) for each strip on a BeadChip, or hexagon on a SAM. These are usually named using the convention `ChipNumber_Array_Strip_Channel.tif`, e.g. `1475542113_A_1_Grn.tif`, is the Cy3 (green) image for strip 1 from array A on BeadChip 1475542113. Cy5 (red) images end with the extension `_Red.tif`.
- bead annotation file (*optional*) - contains information (control status, sequence, etc.) about each bead type on the array. See the file `Human_WG6.csv` for an example. Version 2 and 3 BeadChips store this information in a `.bgx` format. Other platforms have analogous files, such as the `.opa` or `.bpm` for SNP BeadArrays.
- targets file (*recommended*) - contains sample information for each strip/array. See the file `targets.txt` for an example.
- metrics file (*optional*) - one for each each BeadChip or SAM, usually named `Metrics.txt` which contains summary information about intensity, the amount of saturation, focus and registration on the image(s) from each strip or hexagon. This data from this file may be useful for quality assessment purposes. In this example, the metrics file is not available.

The following code can be used to read the example data into R (provided that the contents of `BeadLevelExample.zip` have been extracted to the current working directory).

```
> library(beadarray)
> targets = read.table("targets.txt", sep = "\t",
+     header = TRUE, as.is = TRUE)
> targets
> BLData = readIllumina(arrayNames = targets$ArrayName,
+     textType = ".csv", targets = targets,
+     backgroundMethod = "none", annoPkg = "Humanv1")
```

The function `readIllumina` implements the image processing steps used by Illumina when `useImages=TRUE`, however, both the sharpening and background correction steps are optional.

The background for each bead is estimated by taking the average of the 5 dimmest pixels in a local area around each bead centre. The background value is stored separately and is not subtracted from the foreground value automatically (as occurs in BeadScan output) when `backgroundMethod="none"`. When `useImages=FALSE`, the intensities are read-in directly from the text files. This option has the benefit of saving on memory, although the background intensities are no longer available, as the intensities stored in these text files have already been background corrected by BeadScan.

By default, `readIllumina` will read all arrays in the current working directory for which both text files and TIFFs can be found. For two-colour experiments you will need to set `singleChannel=FALSE` and have both red and green images available.

On Human-6 version 1 chips, the 2 strips for each array contain a different set of bead types. For the newer WG version 2 and 3 BeadChip, each bead type is (generally) represented on each strip. Access to the raw data allows the two strips to be analysed separately.

3 The BLData object

Once imported, the bead-level data is stored in an object of class *BeadLevelList*. 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 a number of slots useful for describing Illumina data. The data from each strip/array can be accessed by subsetting the *beadData* slot by the name of the array. The `$` operator can then be used to extract the appropriate column from the *data.frame*. Alternatively, the `getArrayData` function can be used.

```
> class(BLData)

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

> slotNames(BLData)

[1] "beadData"   "phenoData"  "arrayInfo"
[4] "annotation"

> an = arrayNames(BLData)
> an

 [1] "1475542113_A_1" "1475542113_A_2"
 [3] "1475542113_B_1" "1475542113_B_2"
 [5] "1475542113_C_1" "1475542113_C_2"
 [7] "1475542113_D_1" "1475542113_D_2"
 [9] "1475542113_E_1" "1475542113_E_2"
[11] "1475542113_F_1" "1475542113_F_2"

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

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

> BLData[[an[1]]]$G[1:5]

[1]  647.1598 1291.8708 4646.7958  994.2587
[5]  716.0407
```

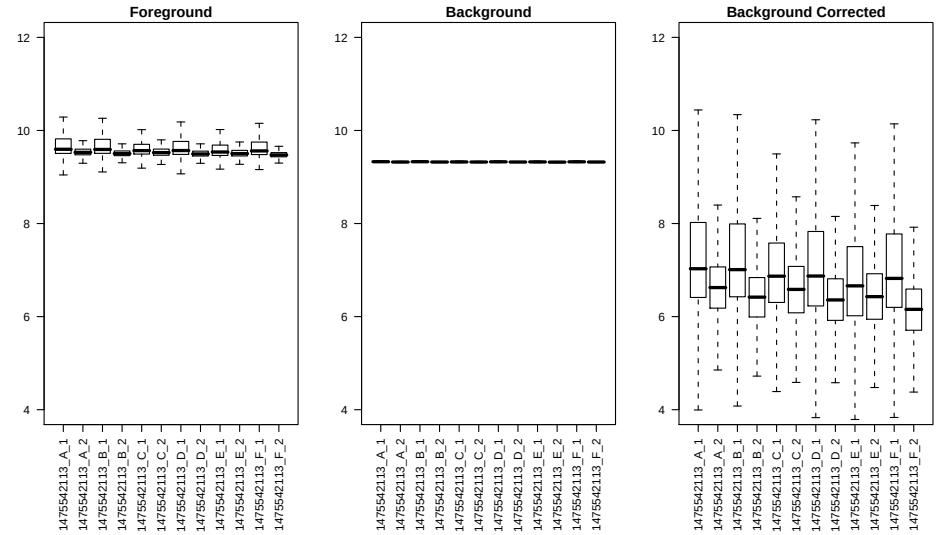
```
> BLData[[an[2]]]$Gb[1:5]

[1] 636 634 635 637 639

> 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)[1:5]

[1]  647.1598 1291.8708 4646.7958  994.2587
[5]  716.0407

> getArrayData(BLData, array = 2, what = "Gb",
+     log = FALSE)[1:5]

[1] 636 634 635 637 639
```

4 Quality assessment

The beadarray has a number of functions that help the in the assessment of the quality of an array/strip. These include boxplots, spatial plots to look for artefacts on the array surface, and plots of the control probes to check their consistency within or between arrays/strips.

We first make some experiment-wide plots which allow us to compare the signal from different arrays/strips.

```
> par(mfrow = c(1, 3), mai = c(2, 0.5, 0.2,
+     0.1))
> 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")
```



In this example we can see that the first strip from each array has a systematically higher intensity than the second strip. This is related to the design of the Human-6 version 1 chips, which contain probes for RefSeq transcripts on the first strip, and probes for less well annotated transcripts on the second strip. Besides this obvious effect, there appears to be a subtle decrease in intensity from top to bottom on this BeadChip (array A is at the top of the chip, through to array F at the bottom of the chip). Notice that the background level appears to be virtually constant both for beads on the same array and between arrays.

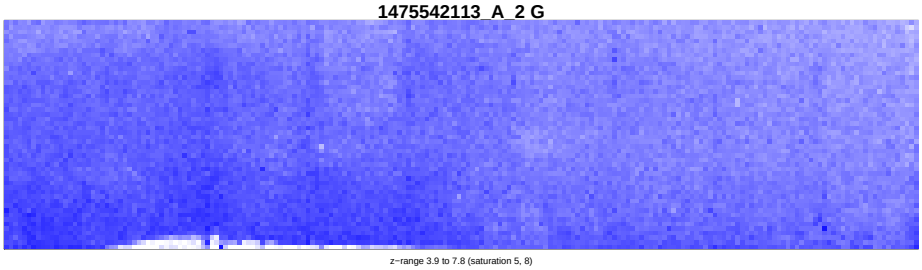
The `whatToPlot` argument of `boxplotBeads` controls which intensities are plotted for each bead. Options are `G`, `Gb` and `residG` (Cy3 residuals) for single channel data and `R`, `Rb`, `residR`, `M` (log-ratios) `residM` or `A` (average log-intensities) for two-colour data.

Background correction can be performed by the `backgroundCorrect` function if it has not already been carried out when the data was read into R by `readIllumina`. The default setting in both functions is to subtract the local background estimate from the foreground of each bead, as per BeadScan. Other options are available by changing the `method` argument in `backgroundCorrect` or the `backgroundMethod` argument in `readIllumina`.

Another useful function is `imageplot`, which generates spatial plots of the intensities which can be used to identify artefacts on the array surface which can occur from mis-handling or scanning problems. Image plots in R are more convenient than scrutinising the original TIFFs, as multiple arrays can be visualised on the one page. Note that this kind of visualisation is not possible when using summarised BeadStudio output, as the summary values are averaged over spatial positions.

Below we generate a spatial plot of the intensities on the second strip of the first array.

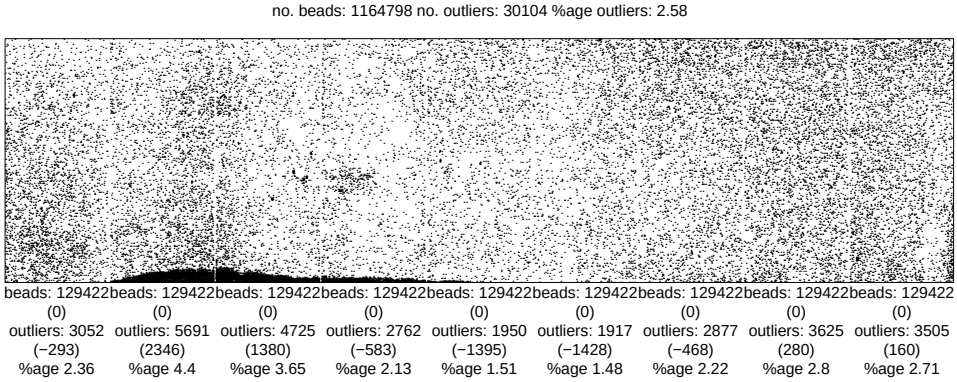
```
> i = 2
> imageplot(BLData.bc, array = i, log = TRUE,
+     nrow = 50, ncol = 200, zlim = range(5,
+         8), what = "G", main = paste(an[i],
+         "G"))
```



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, the `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. Image plots in R are also more convenient than scrutinising the original TIFFs, as multiple arrays can be visualised on the one page.

Recall that the BeadArray technology includes replicates of each bead type on every array ($\sim$ 30 replicates on a Human-6 version 1 array). By default, BeadStudio removes outliers greater than 3 median absolute deviations (MADs) from the median prior to calculating the bead summary values. Illumina perform these calculations on the original scale, which has a tendency to identify more outliers above the median. We prefer to do this on the log_2 scale, which allows outliers above and below the median to be detected more evenly. In the example below, the `outlierPlot` function is used to generate a spatial plot of the outliers.

```
> outlierPlot(BLData.bc, array = i, log = TRUE,
+     plot = TRUE)
```



The artefact on the bottom edge of this strip can be seen clearly in both spatial plots. Such artefacts can be systematically detected and masked from further analysis using the BASH (**b**ead**a**rray **s**ubversion of **H**arshlight) tool. For details on how to use BASH, refer to the user guide which can be launched with the following command

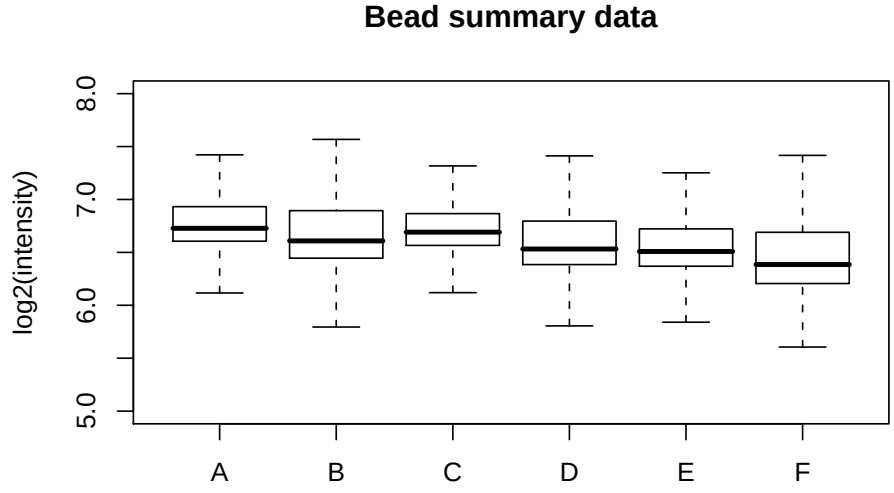
```
> beadarrayUsersGuide(topic = "BASH")
```

Another useful function which generates a number of diagnostic plots with the one command is `calculateBeadLevelScores`. This function creates a spatial plot of the outliers along with plots of the signal of various controls for each array/strip in a *BeadLevelList* and summarises these graphics in an html report.

5 Summarisation

The `createBeadSummaryData` function can be used to summarise the replicate intensities from each bead type. By default, outliers are removed using the 3 MADs rule, and the mean of the remaining beads is taken as the summary value. The number of beads used in the calculation and standard error are also stored. Alternative summarisation methods, such as calculating a trimmed mean, median or mean can be applied by changing the `method` argument of `createBeadSummaryData`. If the outliers have already been removed (which will occur if BeadScan's `settings.xml` file includes the line `<ExcludeOutliers>true</ExcludeOutliers>`), then the mean should be calculated with `method="mean"`. At this point we can combine the two strips for each array by using the `imagesPerArray` argument, leaving us with 6 columns instead of 12. 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. This can be achieved by setting the `what` argument to `RG` or `M` respectively.

```
> BSData = createBeadSummaryData(BLData.bc,
+     imagesPerArray = 2)
> boxplot(as.data.frame(log2(exprs(BSData))),
+     outline = FALSE, ylab = "log2(intensity)",
+     main = "Bead summary data", ylim = c(5,
+         8), names = c("A", "B", "C", "D",
+         "E", "F"))
```



The default settings for `createBeadSummaryData` assumes that the same probes are to be found on each strip/array in the experiment, which will be true in general. The `BSData` object can be analysed further using the functions described in the bead-summary user's guide.

This user guide was built using the following packages:

```
> sessionInfo()

R version 2.8.0 RC (2008-10-12 r46696)
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=C;LC_MESSAGES

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

other attached packages:
 [1] beadarray_1.9.11     sma_0.5.15
 [3] hwriter_0.93         affy_1.19.4
 [5] preprocessCore_1.3.4 affyio_1.9.1
 [7] geneplotter_1.19.6   annotate_1.19.3
 [9] xtable_1.5-4         AnnotationDbi_1.3.12
[11] RSQLite_0.7-0        DBI_0.2-4
[13] lattice_0.17-15      Biobase_2.1.7
[15] limma_2.15.15        weaver_1.7.0
[17] codetools_0.2-1      digest_0.3.1

loaded via a namespace (and not attached):
[1] grid_2.8.0         KernSmooth_2.22-22
[3] RColorBrewer_1.0-2
```

Acknowledgements

We are grateful to Inma Spiteri for providing the data set used in this guide.