1 Analysis of bead level BeadArray data using beadarray

There are two methods for describing the results of a BeadArray experiment. Firstly, we can use *bead level data* whereby the position and intensity of each individual bead on an array is known. *Bead summary data* can also be used whereby a summary intensity for each bead type on an array is given. The summarised values for a particular bead type can then be compared between different arrays within an experiment.

Whilst the *beadarray* package includes methods for processing data of both kinds, bead summary data is far more widely available at the present time. Data can be data obtained using either the BeadChip (6 or 8 arrays on a slide) or SAM (arrays organised in 96 well plates) technologies. This document uses a SAM experiment as an example although BeadChip can be read in the same manner.

The purpose of this document is to give an outline of the R functions for analysing bead level data. Descriptions of how to read and analyse bead summary data are provided in a separate Vignette. Those who are familiar with the R statistical language, and in particular the *limma* package, should be able to adapt easily to our new methods of analysis. Wherever possible we used objects that are similar to those used by *limma*. Example files to read bead-level data are provided at

http://www.damtp.cam.ac.uk/user/jcm68/beadarray.html

The paper by Dunning et al *Quality Control and Low-level Statistical Analysis of Illumina Beadarrays*, Revstat **4**, 1-30, describes the analysis of bead level data and pre-processing algorithms used by Illumina in more detail than this Vignette.

1.1 Reading bead-level data

There are two sets of files that are required by our package in order to create bead-level data.

- TIFF images - These are the raw images scanned directly from each individual array on a 96-well SAM. These are provided by Illumina.

- csv files - These define the location and bead type of each individual bead on a particular array on a 96-well SAM. These are output by the BeadScan software using to scan BeadArrays (versions 3.1 and above

```
> library(beadarray)
> beadTargets = readBeadTargets()
> beadTargets

                    Image1                 xyInfo SAMPLE
1 1269941_R001_C001_Grn.tif 1269941_R001_C001.csv      6
2 1269941_R001_C002_Grn.tif 1269941_R001_C002.csv      6

> BLData = readBeadImages(beadTargets)
```

```
Reading pixels of 1269941_R001_C001_Grn.tif
Calculating background
Calculating foregound
Reading pixels of 1269941_R001_C002_Grn.tif
Calculating background
Calculating foregound
```

The default setting for `readBeadImages` is to recreate the foreground and background intensities for each bead in the same way in which they are calculated by Illumina. However, the use of sharpening and local background correction are optional (see Dunning et al for description of sharpening and local background correction). To create unsharpened bead intensities one would use:

```
> BLData.ns = readBeadImages(beadTargets, imageManipulation = "none")
```

```
Reading pixels of 1269941_R001_C001_Grn.tif
Calculating background
Calculating foregound
Reading pixels of 1269941_R001_C002_Grn.tif
Calculating background
Calculating foregound
```

Data which does not include the coordinates of bead centres may still be read into the library. It will not be possible to perform any image processing without the bead centre coordinates. Files of this type may not always have the same number of bead present so we have to specify the maximum length of any of the files to be read.

```
> files = dir()
> BLData.nonXY = readNonXYData(files, max_length = 49777)
```

1.2 The BeadLevelList Object

The data object (`BLData`) is in fact a list object but behaves like a complex sort of matrix. It can be subsetted or treated like a matrix in lots of ways. We can use the `names` command to see what items can be found in the list. `BLData` is an *BeadLevelList* object and like the *RGList* object in *limma* can contain `R`, `Rb`, `G` and `Gb` objects (i.e. foreground and background intensities of two colour data).

```
> is(BLData)
```

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

```
> names(BLData)
```

```
[1] "R"                 "Rb"                "x"                 "y"
[6] "targets"           "backgroundSize"    "normalised"        "backgroundCorrected
```

Individual items in the list can then be accessed by using the `$` operator in R. In our example we have the matrices `R` and `Rb` which are the foreground and background intensities for each bead (row) and each array (column). The

example shown here is for a single channel experiment, hence we only have a foreground intensity value in the red channel and the green channel is not used. If we had two channel data then `BLData$R` and `BLData$G` would be the red and green channels respectively. The number of rows in the matrix is the same as the number of beads present on the array and the number of columns is the same as the number of arrays. In this example we only read in two arrays, so we only have two columns. In other words, each column of the matrix represents intensities of all beads on the same array. However, due to the random placement of beads on the array, each row of the matrix does not relate to intensities of a bead of the same type (as one might expect having dealt with conventional microarray data).

Since BeadArray technology uses randomly assembled beads it is important to know the location and identity of every bead on the array. Therefore the `BeadLevelList` we are using in this library also contains the x and y co-ordinates for each bead and an identifier (ProbeID) for the bead type of each bead.

An example *BeadLevelList* is included with the library. However, due to space constraints, the object only contains a single array with 25000 beads on. The object may be loaded at any time with the command.

```
> data(BLData)
```

1.3 Background correction and normalisation

Using the `boxplot` function in R allows boxplots of foreground and background intensities to be compared (see Figure **??**).

```
> boxplot(as.data.frame(cbind(log2(BLData$R), log2(BLData$Rb))), names = c("Array 1 Foreground", "Ar
+     "Array 2 Foreground", "Array 2 background"), las = 2)
```
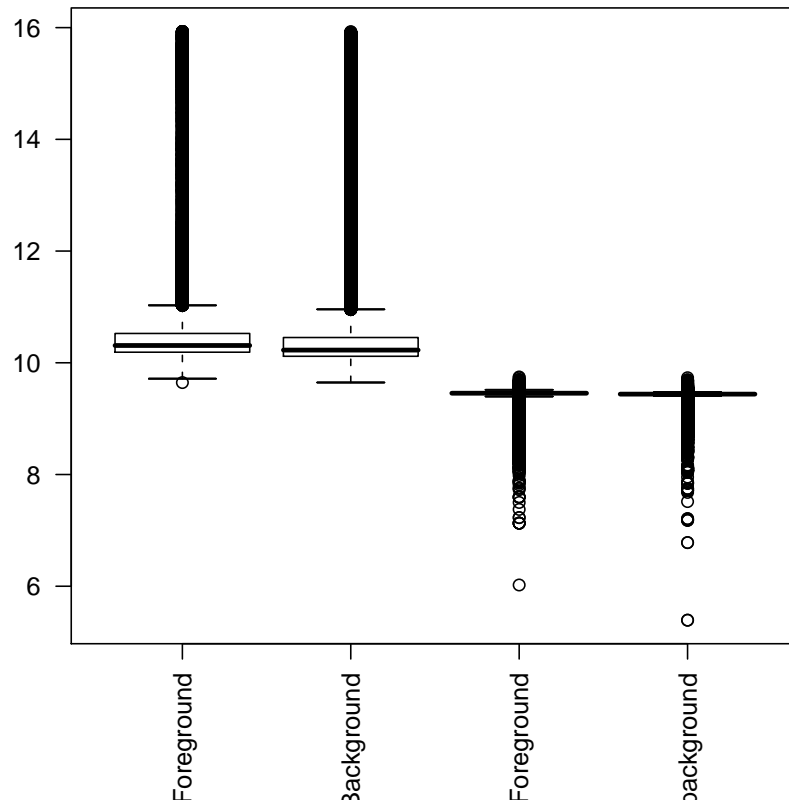
Local background correction can be be performed on the data by:

```
> BLData.c = backgroundCorrectBeads(BLData)
> boxplot(log2(BLData.c$R) ~ col(BLData.c$R), main = "Background Corrected Foreground Intensities")
```

Background Corrected Foreground Intensities
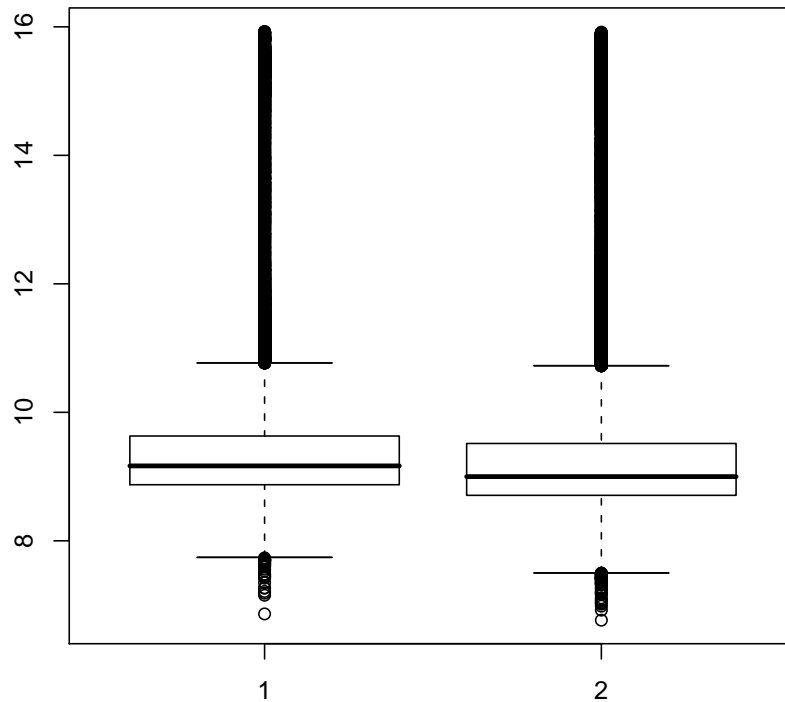
By default, the `backgroundCorrectBeads` function subtracts the values in `BLData$Rb` from `BLData$R` and stores the result in the `R` matrix of the resulting *BeadLevelList* object. Other methods are available such as *minimum* which ensures that no negative values are produced. The only normalisation methods currently supported for bead-level data are median and quantile normalisation. Compatibility with the *limma* and *affy* libraries means that other methods can easily be used.

```
> BLData.med = medianNormalise(BLData)
> BLData.q = quantileNormalise(BLData)
```

1.4 Numbers of beads

We can see which bead types are represented less than 24 times (the 5th percentile for the appropriate Poisson distribution) on the array using `findLowestCounts`. For the first array that we use:

```
> findLowestCounts(BLData, 1)[1:10]
```

```
 [1]   10   23   30   42   87  119  182  185  585  607
```

The result is a list of ProbeIDs.

5

1.5 Outliers for each bead type

The `plotBeadIntensities` function can be used to assess variation within a particular bead type. This function shows the intensity of every bead of a particular type against the distance of the bead from the centre of the array. Any outliers which exist for the bead type are marked on the plot by a red cross. As an example we can plot the intensities of all beads with probeID 2 on array 1 and determine outliers using unlogged or $\log_2$ intensities. This function also has the option of changing the number of MADs from the mean used to determine outliers by changing the n parameter.

```
> par(mfrow = c(1, 2))
> plotBeadIntensities(BLData, ProbeIDs = 2, array = 1)
> plotBeadIntensities(BLData, ProbeIDs = 2, array = 1, log = TRUE)
```
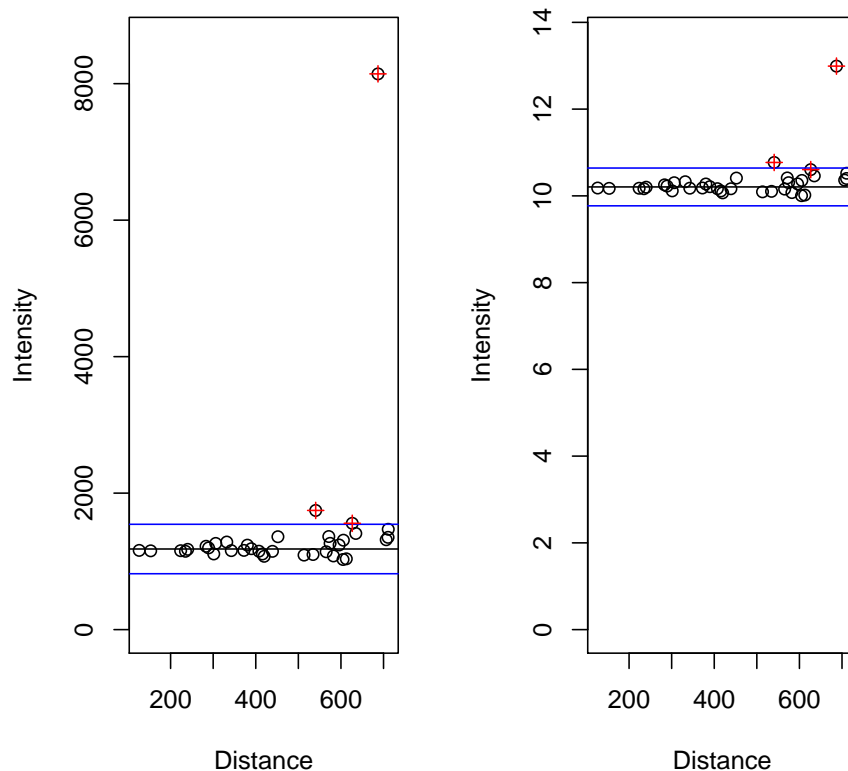

We can find all the beads on an array which are outliers for their bead type by using the `findAllOutliers` function. The output of the function is an index which refers to a particular row in the *BeadLevelList* where a bead is located.

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

 [1] 1056 1070 1072 1075 1080 1099 1140 1186 1213 1241
```

The length of the list can be easily found (`length`) and compared between different arrays as a diagnostic measure for the quality of the array. Additionally, the location of all the outliers on an array can be plotted using the `plotBeadLocations` function. described below. The function `findBeadStatus` may be used for individual bead types.

```
> findBeadStatus(BLData, probes = 2, array = 1, outputValid = TRUE)

$outliers
[1] 1056 1070 1072

$valid
 [1] 1037 1038 1039 1040 1041 1042 1043 1044 1045 1046 1047 1048 1049 1050 1051 1052 1053
[26] 1063 1064 1065 1066 1067 1068 1069 1071

$nextStart
[1] 1072
```

Here, the result is a list of beads with ProbeID=2 which are outliers for this bead type, along with beads which are valid beads.

1.6 Spatial plots

The `plotBeadLocations` function can be used to plot the location of a set of beads on an array. Beads can be specified by ProbeIDs or by a list of row indices as returned by `findBeadStatus`.
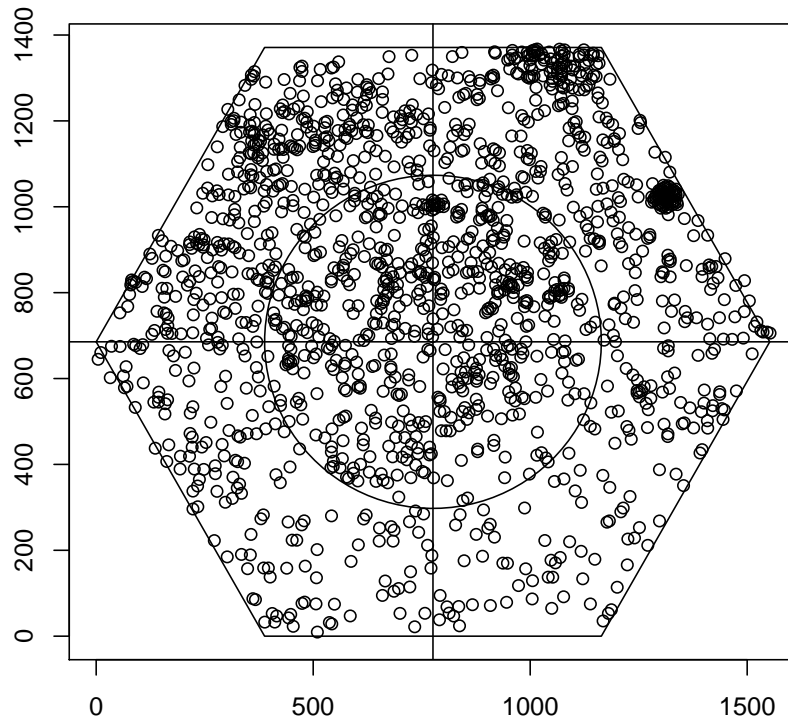
```
> plotBeadLocations(BLData, ProbeIDs = 2, array = 1)
```
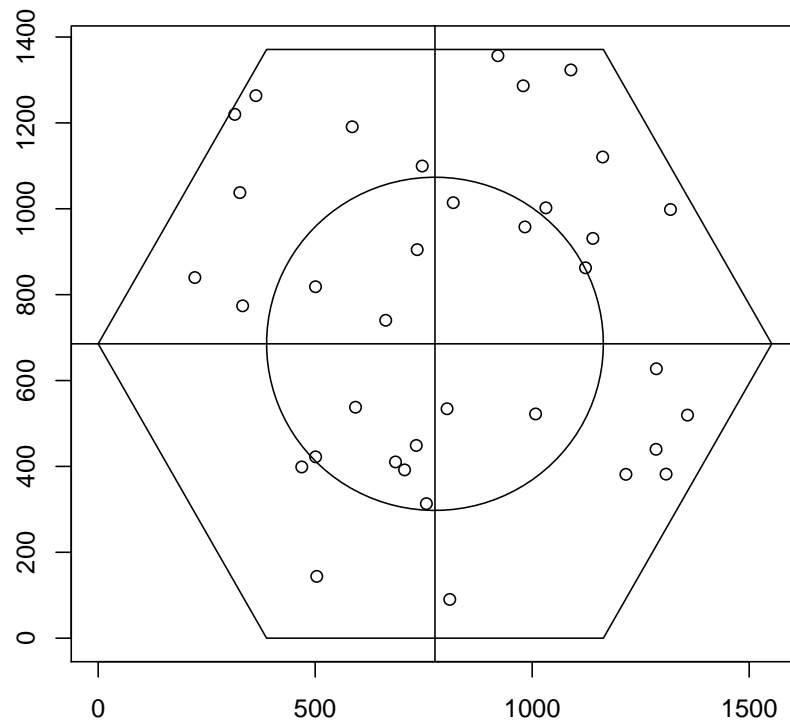
The function plots all beads on the first array with ProbeID 2. By using the
o object created above we can also plot the location of all outliers on the first
array.

```
> plotBeadLocations(BLData, beadIDs = o, array = 1)
```

The `plotBeadLocations` provides a quick diagnostic check for the distribution of a set of beads. As described in Dunning et al., we have also implemented a χ^2 statistic to quantify the non-randomness of bead distributions. This χ^2 test can be applied to all bead types on an array and the probeID of those with the highest value can be returned by:

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

 [1]  213  278  606  658  791  800  936  960  961 1071
```

Shown above are the ProbeIDs for the first 10 bead types with a χ^2 statistic greater than 14 (chosen because this is the 5th percentile of the appropriate χ^2 distribution). Any of these bead types can be investigated further by using the `plotBeadLocations` function.

Any regions on an array found to have a high proportion of outliers can be investigated further by the `displayTIFFImage` function. See Figure 3 of Dunning et al.

```
> displayTIFFImage(BLData, array = 1, a = 1000:1400, b = 1200:1400)

[1] "1269941_R001_C001_Grn.tif"
[1] "finding the outliers...."
```

9

```
[1] "<mean 13"
[1] ">mean 100"
[1] "negative intensity 0"
```

The example above loads the original image for array 1 (the name of which is stored in the targets object) and displays the intensities of pixels with x in the range from $1000 : 1400$ and y in the range $1200 : 1400$.

The intensity of every pixel in the plot is represented by a shade of green, with brighter colours indicating a higher value. The blue and red spots indicate the position of outliers in the particular region with blue indicating beads with intensity higher than the mean for that bead type and red being beads with intensity lower than the average for their bead type. Yellow spots on the picture represent beads which have been calculated to have a negative foreground intensity. The black crosses show where the bead centres are located. Any beads which failed the decoding process can also be highlighted by setting the *showUnregistered* parameter.

The plot can also be made interactive by setting the *locateBeads* parameter to TRUE. We can then click on any bead centre and display the foreground and background intensities for this bead as well as a measure of the raw intensity.

We feel that `displayTIFFImage` gives more useful information about the raw images than the equivalent function included in BeadStudio. In BeadStudio, the user can explore the TIFF images and see the intensity of each individual pixel. However, the identity of each bead on the image is not given and there is no information about outliers.

1.7 Creating bead summary data

Bead summary data can be created using the bead level data. In producing these summaries we must first remove outliers for each bead type as described in Dunning et al. This averaging is done by the `createBeadSummaryData` function and the method of detecting outliers can be specified by changing the *log* (for unlogged or logged parameters) and n (number of MADs) parameters.

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

The structure of the resulting object is described in greater detail in the Vignette *Analysis of bead summary data with beadarray* supplied with *beadarray*.