REVSTAT – Statistical Journal
Volume 4, Number 1, March 2006, 1–30

QUALITY CONTROL AND LOW-LEVEL STATISTICAL ANALYSIS OF ILLUMINA BEADARRAYS

Authors: Mark J. Dunning
– Department of Oncology, University of Cambridge, England
md392@cam.ac.uk

Natalie P. Thorne
– Department of Oncology, University of Cambridge, England
npt22@cam.ac.uk

Isabelle Camilier
– Ecole Polytechnique, 91128 Palaiseau, France
camilier@poly.polytechnique.fr

Michael L. Smith
– Department of Oncology, University of Cambridge, England
mls40@cam.ac.uk

Simon Tavaré
– Department of Oncology, University of Cambridge, England and Department of Biological Sciences, University of Southern California, USA
s.tavare@damtp.cam.ac.uk

Abstract:

- The Illumina BeadArray™ platform is a novel microarray technology based on randomly assembled arrays of beads. Each bead on the array carries copies of a single gene-specific probe with, on average, about 30 replicates of each bead type on an array. Given the encouraging results regarding the reproducibility of BeadArray™ data and high profile studies already being carried out using the BeadArray™ technology, there is likely to be an increase in the volume of BeadArray™ data available. A major advantage of BeadArray™ technology is the high degree of replication of beads of a given type. However, current analysis methods give summarised information for each bead type as output rather than information for each individual bead on the array. The *beadarray* R package is able to recreate individual bead information for arrays using raw images as input. Here, we use a particular experiment to illustrate the image processing steps used by Illumina and corresponding methods available in *beadarray*. Our investigations into BeadArray™ data have demonstrated a high degree of reproducibility both within and between arrays. However, we identified some aspects of the low-level analysis that could be improved.

Key-Words:

- *Illumina; BeadArray; beadarray; Bioconductor; microarray.*

AMS Subject Classification:

- 62P10, 92C40, 92-08.

1. BACKGROUND

A BeadArray$^{\text{TM}}$ is an array of randomly positioned, three micron diameter, silica beads. Around 10^5 copies of a particular DNA sequence of interest are covalently attached to each bead ([15]). The position and identity of each bead on the array is determined using an automatic registration algorithm ([5]) and a molecular address ([7]). The DNA sequences attached to the beads are 75 base pairs in length, with 25 base pairs used for decoding and 50 base pairs for target hybridisation. This long-oligonucleotide approach has been shown to agree well with the popular short-oligonucleotide technology used by Affymetrix ([2]).

A pool of different bead types is created, beads of the same type having the same probe sequence attached. Separately, a fibre-optic bundle is treated with acid to create wells for individual beads to fit in ([10]). The fibre-optic bundle is exposed to the bead pool, causing the beads to be randomly sampled and assembled in the wells on the surface of the bundle.

Illumina have developed two different platforms which combine multiple BeadArrays. A Sentrix$^{\text{TM}}$ Array Matrix (SAM) contains 96 arrays, each of which has approximately 50,000 beads and around 1500 distinct bead types. A BeadChip$^{\text{TM}}$ allows either 24,000 bead types to be interrogated on eight samples simultaneously or 48,000 bead types across six samples. These multiple array technologies make the BeadArray$^{\text{TM}}$ platform especially suitable for high throughput experiments ([3], [1], [8]). The distribution of bead types on an array is effectively Poisson due to the random sampling of beads from the very large bead pool. Each bead type is represented about 30 times on average with extremely low probability of any bead type being represented less than five times ([9], [7]).

Large volumes of data can be generated using a single BeadArray$^{\text{TM}}$. Given these various new array technologies, there is clearly a need for statistical tools to analyse such data. There is already a wealth of software for statistical analysis of microarray data available in R packages on *Bioconductor* ([6], `www.bioconductor.org`). One of the most commonly used packages is *limma* (Linear Models for Microarray Analysis, [12]), which is an analysis package for two-colour microarrays. *beadarray* uses a similar programming style to *limma* and has recently been submitted as a development package to Bioconductor. The source for the package can be obtained at

`http://www.bioconductor.org/packages/bioc/1.8/html/beadarray.html`.

We used the data described in [16] to demonstrate the functionality of *beadarray* and to investigate the low-level analysis, processing and quality of BeadArray$^{\text{TM}}$ data. [16] studied the expression levels of some 700 genes measured in cell lines from 60 CEU individuals used in the Hapmap project ([3], [1]). The experiment comprises five SAMs with each of the 60 individuals replicated 6–8 times. Each array on the SAM had 1471 bead types with multiple (usually two) bead types for each gene under investigation and included various control probes.

2. PROCESSING OF BEADARRAY™ DATA BY ILLUMINA

In this section we describe the steps involved in the analysis of BeadArray™ data.

2.1. Image processing

The image processing steps used by Illumina to calculate bead intensities from raw images are described in [9]. These are given below:

(i) All pixel intensities are altered using a sharpening transformation. The intensity of a particular pixel is made higher / lower if its intensity is high / low in comparison to the intensities of the pixels surrounding it.

(ii) Foreground intensities are calculated as a weighted average of signals obtained using the four pixels nearest to each bead centre as a virtual bead centre. Sharpened pixel intensities are used in the calculation.

(iii) The local background, an average of the five dimmest pixels (unsharpened intensities) within the 17×17 pixel area around each bead centre, is subtracted.

Currently, raw TIFF images (see Figure 1) are read by an Illumina BeadArray Reader, giving bead-level data which can be read into the software package BeadStudio for analysis. At present, these bead-level data are encrypted and cannot be viewed directly. Bead-summary data can be output from BeadStudio with an unlogged, averaged intensity given for each bead type along with the number of beads used to calculate the average and the standard deviation of unlogged bead intensities.

2.2. Outlier removal and creation of bead-summary data

Outliers for each bead type are detected using the unlogged intensities of all beads of the same type. Any beads with intensity more than three median absolute deviations (MAD) from the mean are classed as outliers and excluded. The background measure reported for all beads is a single value, the mean of the negative controls on an array, rather than the local background values that are subtracted from each bead intensity.

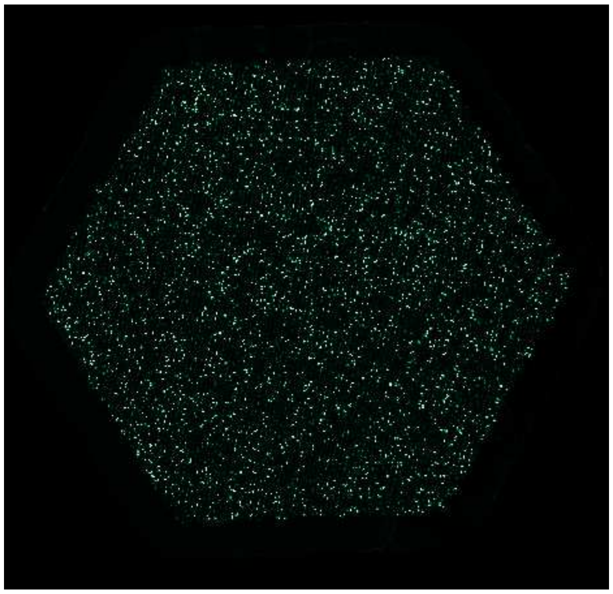

Figure 1: A raw image scanned from a BeadArray™ and viewed through BeadStudio. As with images scanned from conventional microarrays, each pixel intensity on the image represents the amount of hybridisation detected at each point on the array. Each three micron bead on the array is represented by nine pixels in a 3×3 square and are located roughly six microns apart.

2.3. Quality control of bead-summary data

Visualisation tools provided by BeadStudio include:

(i) plots of the unlogged intensities of control probes across all arrays;

(ii) scatter plots for comparing bead-summary data between two arrays;

(iii) interactive image plots of raw images with information about the intensity of each pixel (rather than bead intensity) on the image;

(iv) agglomerative clustering (average linkage method) of genes or samples using various distance and similarity measures.

The statistical analysis incorporated in BeadStudio for assessing differential expression between samples includes a Mann-Whitney test, Illumina custom (iterative robust least squares fit) or standard t-test. Normalisation choices include scaling by array averages, qspline [17] or scaling based on controls or rank invariant genes. Intensities may also have an additional background correction applied. This is based on the average of negative control genes and is called the method of background normalisation by Illumina.

3. PROCESSING OF BEADARRAY DATA using beadarray

The *beadarray* package was written to implement the analysis of BeadArray$^{\text{TM}}$ data in R in the same manner as two-colour microarrays or Affymetrix data, and to investigate image processing. An important feature of *beadarray* is the ability to access full bead-level detail for arrays rather than the bead-summary data given by BeadStudio. *beadarray* can also be used to analyse pre-processed bead-summary data created by BeadStudio.

3.1. Image processing

beadarray is able to create bead-level data by using the raw images scanned from BeadArrays, the locations of the bead centres and by implementing the steps described in Section 2.1. However, local background correction and sharpening are optional within *beadarray*, as is the background normalisation using negative controls. *beadarray* supports some of the background correction methods available in the *limma* R package, along with a selection of standard normalisation methods ([14]) found in *limma* and the *affy* packages.

3.2. Spatial plots

beadarray includes methods to identify automatically and to investigate any spatially dependent problems which may occur on BeadArrays. Users are able to screen all arrays in an ad-hoc manner to identify any arrays with unusual distributions of beads and these arrays can then be viewed in more detail.

We have implemented a test statistic to investigate the spatial randomness of a set of bead coordinates on an array. For this we divide the hexagonal array into eight sections and use a χ^2 goodness-of-fit test to assess the randomness of the number of beads found in each section (see Figure 2).

We use the χ^2 statistic to identify bead types on an array with apparent non-randomness and investigate these further with a spatial plot function. We also find it useful to apply similar χ^2 tests on the positions of the outliers so that arrays with spatial clustering of outliers can be quickly and easily identified. The raw images corresponding to the arrays can then be viewed through *beadarray* and the location of outliers in a region can be highlighted (see Figure 3). The function for displaying images can also be run interactively; clicking on a particular bead displays information such as the local background level, unsharpened intensity and identity of the bead.

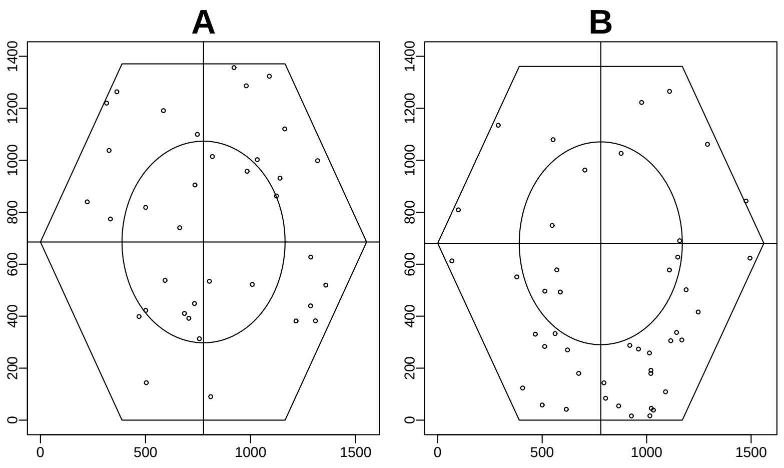

Figure 2: Using *beadarray* to assess randomness of bead positioning.
(**A**) The array is divided into eight sections of roughly equal area. The coordinates for a particular bead type are used to find the number of beads located in each of the sections. These are then compared to the expected number of beads if the beads were randomly distributed among all sections. The beads in this example are uniformly spread. (**B**) For this bead type there is a tendency for beads to be located in the lower half of the array.

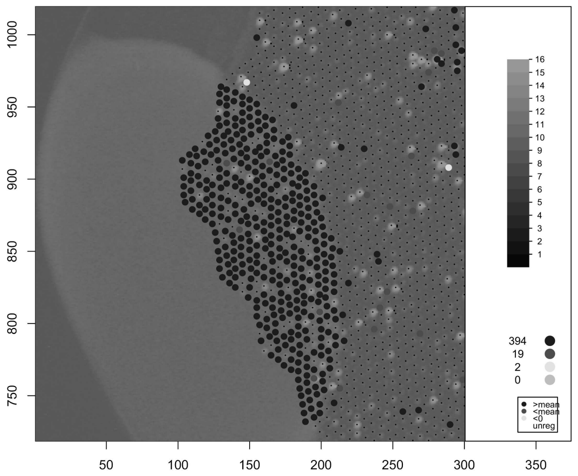

Figure 3: Viewing TIFF Images. Figure produced by *beadarray*.
beadarray allows regions on the original TIFF images to be viewed in more detail. The intensity of each pixel is given on the $\log_2$ scale with a brighter shade of green indicating a larger intensity. Bead centres are indicated by black crosses. This screenshot shows a region on an array with many outliers. Beads which are outliers are indicated by blue or red dots if their intensities are higher or lower than the mean for their bead type. Any beads which failed the decoding process can also be highlighted if desired. An interactive mode is also available whereby clicking on a particular bead gives information about that bead, such as the identity of the bead type and the foreground and background intensities. For colour picture see Supplementary Figures available at `http://www.damtp.cam.ac.uk/user/jcm68/beadarray.html`

3.3. Outlier removal

To visualise the variability of beads of the same type on the same array we plot the distance of each bead from the centre of the array against the $\log_2$ or unlogged intensity (see Figure 12). *beadarray* allows outliers to be identified using either unlogged or $\log_2$ intensities and using arbitrary numbers of MADs from the mean (the default is the Illumina setting of three MADs from the mean).

3.4. Quality Control

The average intensities of each bead type can be compared between multiple arrays using MA plots and scatter plots [4]. We can also compare the average intensity for any given bead type (not just control probes as in BeadStudio) between different arrays in the experiment and relate this information to the position of each array on the SAM using a "SAM Summary plot" (see Figure 4). The function used to create this SAM Summary plot can use any set of 96 values as input rather than just the values of control probes. For instance, one could plot the number of outliers found on each array to summarise the number of outliers on arrays over the SAM.

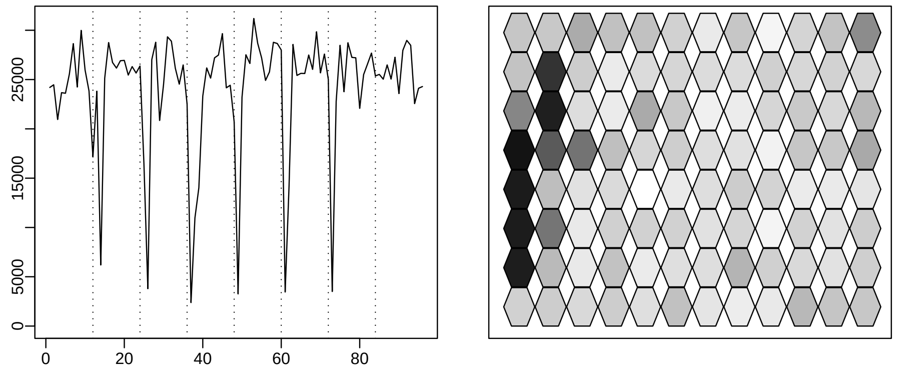

Figure 4: The SAM summary plot available in *beadarray*.
In this plot we show the bead average values for a particular control across all arrays on a SAM. In the left-hand plot the (unlogged) bead average value is plotted against array index. On the right-hand plot, 96 hexagons are shown in the same arrangement they appear on the SAM. The colour of each hexagon is related to the value of the bead average on that array; darker shades of grey indicate lower values. For this case we can see that the lowest bead average values occur on the left side of the SAM.

At present, *beadarray* is a package for quality control and low-level analysis only and does not provide any methods for determining differentially expressed genes. However, since *beadarray* was developed in the same programming style as *limma* it should be straightforward to adapt existing methods for linear modeling ([11]). Accessibility to full bead-level data also makes it easier to combine the data from different arrays in a more flexible way. In particular, it is possible to produce weighted averages for each bead type over different replicates on either the $\log_2$ or unlogged scales. In BeadStudio, only unlogged values are available and therefore averages can only be combined on the unlogged scale.

4. RESULTS

In this section we show the results of our investigation into the methods used by Illumina for low-level analysis and image processing. This section also demonstrates some of the functionality available within *beadarray*.

4.1. Numbers and positioning of beads

The random sampling used in the construction of BeadArrays gives a random placement of beads and an average of approximately 30 of each bead type on an array. This randomness minimises the influence of spatially localised effects and lends robustness to the calculation of bead-summary data. The diagnostic tests described in Section 3 can be used to confirm the random nature of BeadArrays (see Figure 5). For every array under investigation, the mean number of beads of each bead type was found to be approximately 30 as expected.

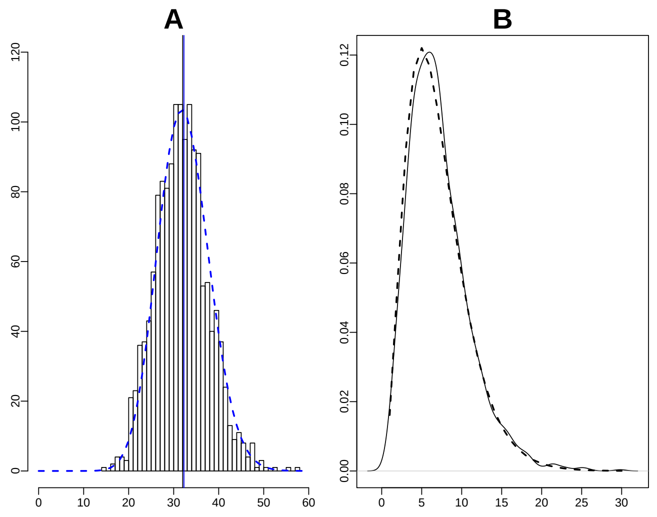

Figure 5: Random properties of BeadArrays. Figures produced by *beadarray*. (**A**) Histogram of the number of times each bead type is found on an array before outlier removal. The dotted line indicates the expected Poisson frequencies. (**B**) Distribution of the χ^2 statistics calculated for each bead type on an array. The dotted line indicates the expected χ^2 distribution.

The Poisson distribution of counts of each bead type shows that the probability of a bead type being represented less than five times is extremely low. Moreover, Illumina will not release an array where this has occurred. Even after outlier removal, no bead type on the five SAMs was found to have less than 11 beads on any array. Calculating the χ^2 statistic for all bead types on an array and repeating for all arrays confirms the random distribution of the beads.

4.2. Image processing

We used one array at a time from one SAM to look closely at the image processing steps used by Illumina. For each of these arrays, we took the raw image for the array and bead centre information provided by Illumina to calculate the foreground and local background intensities for each bead using the *beadarray* implementation of the steps described in Section 2.1. We did not perform background correction on the foreground intensities so that we could analyse the foreground and local background levels separately. Foreground intensities were also calculated without using the sharpening mask prior to averaging pixels for each bead (unsharpened intensities).

Figure 6 shows the effect of image processing on five arrays from the same SAM. Similar results were obtained for all arrays on the same SAM. From boxplot 6A we see that the unsharpened $\log_2$ intensities have a very low dynamic range and most of the bead intensities are concentrated between 10 and 11. As we are reading 16-bit images the maximum value for unsharpened $\log_2$ intensities is 16. However, if we apply sharpening to all pixels in the image and then calculate bead intensities, we affect the range of bead intensities produced. From the boxplots in 6B we see that the maximum values of bead intensities are now greater than 16 and the minimum values in the boxplots are lower than the minimum of boxplots in 6A. This implies that sharpening is increasing some bead intensities while decreasing others. The inter-quartile range of boxplots in 6B appear to be the same as for boxplots in 6A. In the boxplots in 6C we can see that the local background levels calculated for individual beads are virtually constant. Performing a local background correction on the sharpened intensities gives the intensities seen in boxplots in 6D. Similar results were produced using a global background correction (median of all local background measures). We notice that the dynamic range of boxplots in 6D are much higher than both boxplots in 6A and 6B.

Figure 7A confirms the results seen in boxplots in 6B. For most beads we see a positive difference between the sharpened and unsharpened intensities. However, it is also possible for sharpening to cause a decrease in bead intensities. This is most likely to happen for beads with low unsharpened intensities. For some beads with low intensity the effect of sharpening is very dramatic, making the beads have very low intensity. Note from Figure 6A that most unsharpened beads lie within the range 10–11 on the $\log_2$ scale so could be highly altered by sharpening. Figure 7B also shows that background correction can have a very different effect depending on the unsharpened intensity. Local background correction is done using unlogged intensities. Therefore, it is not surprising that the higher intensity beads are less affected by local background correction. Lower intensities are affected much more dramatically by local background correction. For beads less than $\log_2$ intensity 11 the effect is nearly always negative, counteracting the increased intensity generally caused by sharpening. It seems that sharpening and local background correction have a non-linear effect on the data even on the $\log_2$ scale. This phenomenon is consistent on all arrays we investigated.

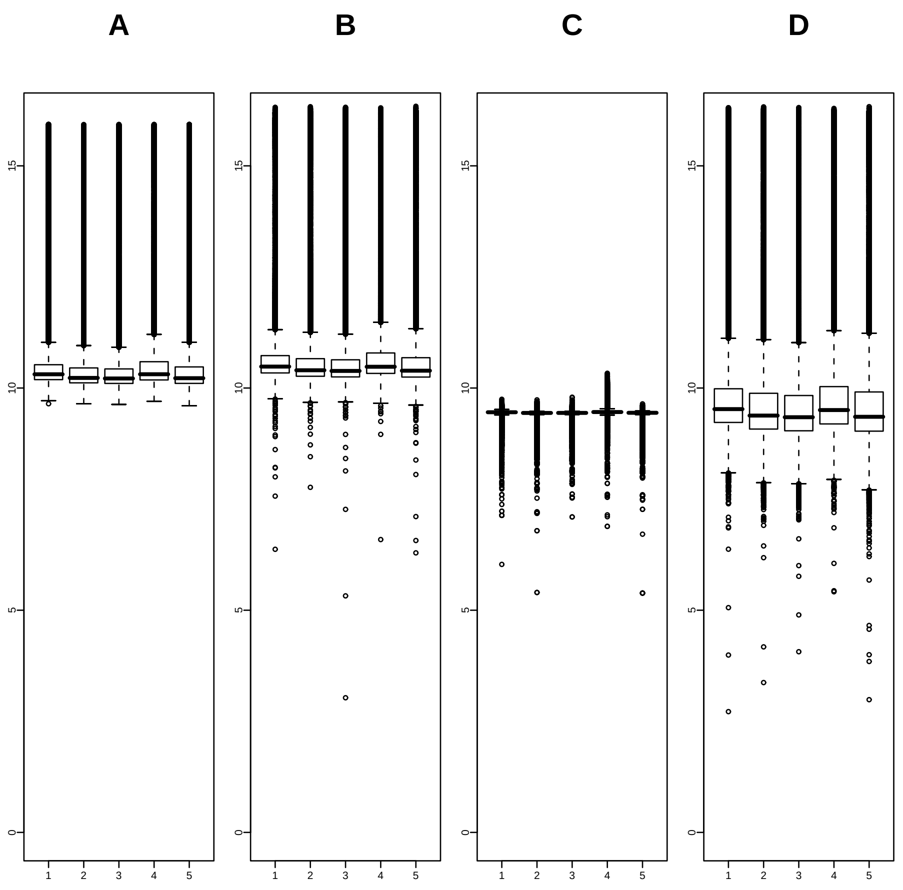

Figure 6: Effects of sharpening and background correction on raw intensities. Figures produced by *beadarray*.
(**A**) Boxplot of the unsharpened foreground intensities of all beads on the array. (**B**) Boxplot of foreground intensities of all beads on the array calculated using the sharpening mask. (**C**) Local background levels calculated for all beads on the array. (**D**) Boxplots of sharpened foreground intensities which have been background corrected by subtracting the local background. (These are the foreground intensities calculated by Illumina.)

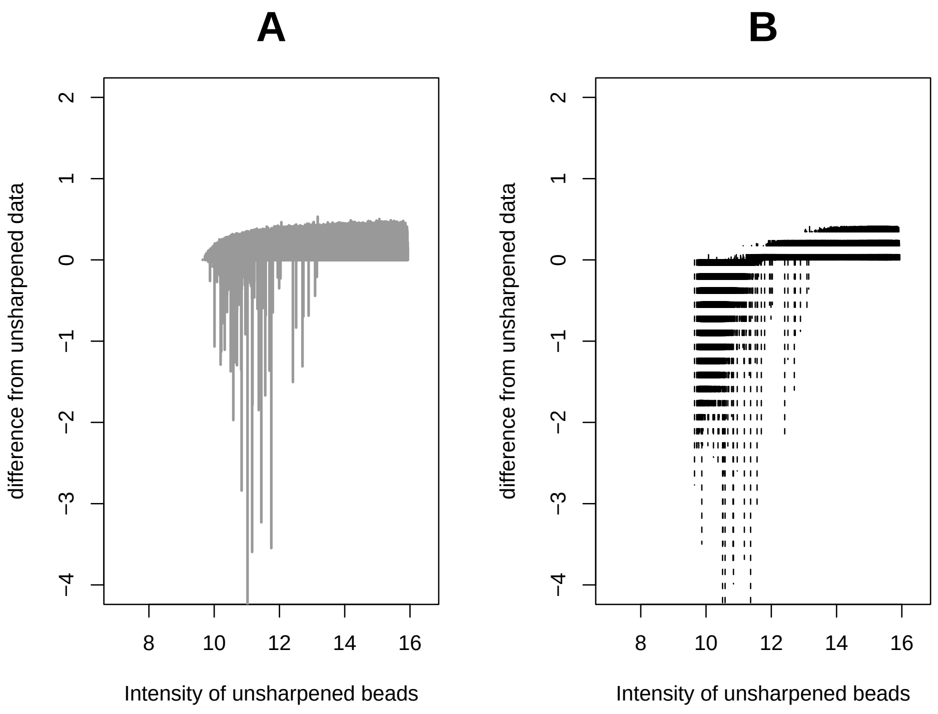

Figure 7: The effect of sharpening and local background correction on raw intensities for a particular array.
(**A**) The x axis represents the unsharpened intensities of all beads on an array and the y axis represents the difference in unsharpened and sharpened intensities for each bead. (**B**) Here, the y axis represents the difference between the unsharpened intensities of each bead and the intensity after sharpening and background correction. See Supplementary Figures at `http://www.damtp.cam.ac.uk/user/jcm68/beadarray.html` for a larger version of this figure.

4.3. Variability within bead types

Figures 6 and 7 demonstrate the global effect of sharpening and background correction on all beads on the array, but it is also of interest to know the effect on beads of the same type. In Figure 8 we show the standard deviation (SD) for all replicates of the first 50 bead types on an array both with and without sharpening and on the unlogged scale. Note that no outliers have been removed at this stage. The lowest SD is seen when foreground intensities are calculated without using sharpening or background correction. If we apply sharpening to all pixels in the image prior to calculating foreground intensities we see a slight increase in SD on the $\log_2$ scale. Background correction on the foreground intensities calculated using the sharpening mask results in a marked increase in SD.

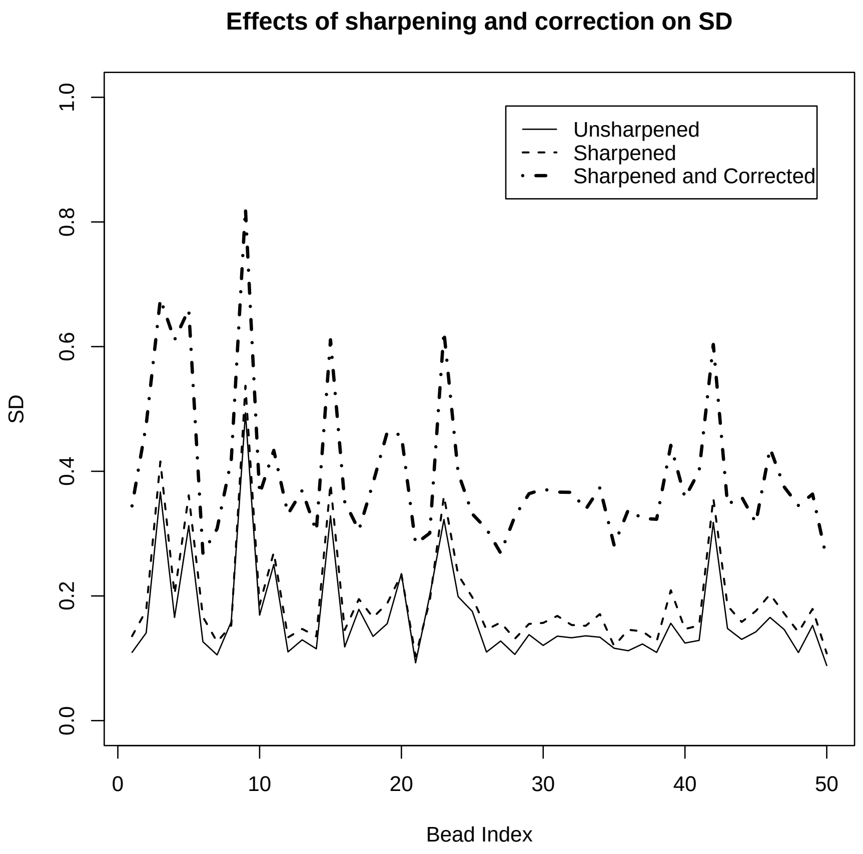

Figure 8: Effect of sharpening and local background correction on standard deviation of beads.
For the first 50 bead types on an array we show the standard deviation of all beads using $\log_2$ intensities which are unsharpened, sharpened or sharpened and background corrected.

4.4. Variability within and between arrays

We used one-way ANOVA on the unlogged foreground intensities on three replicate arrays to calculate the mean square error (MS) of each bead type within and between arrays. We repeated this using both sharpened and unsharpened intensities (see Figure 9). We again see that sharpening has the effect of increasing the variability between beads of the same type, both within and between arrays. The result is the same even when we use CV to measure the variability relative to the overall increased intensity due to sharpening. As would be expected, the variability between arrays is higher than the variability within arrays.

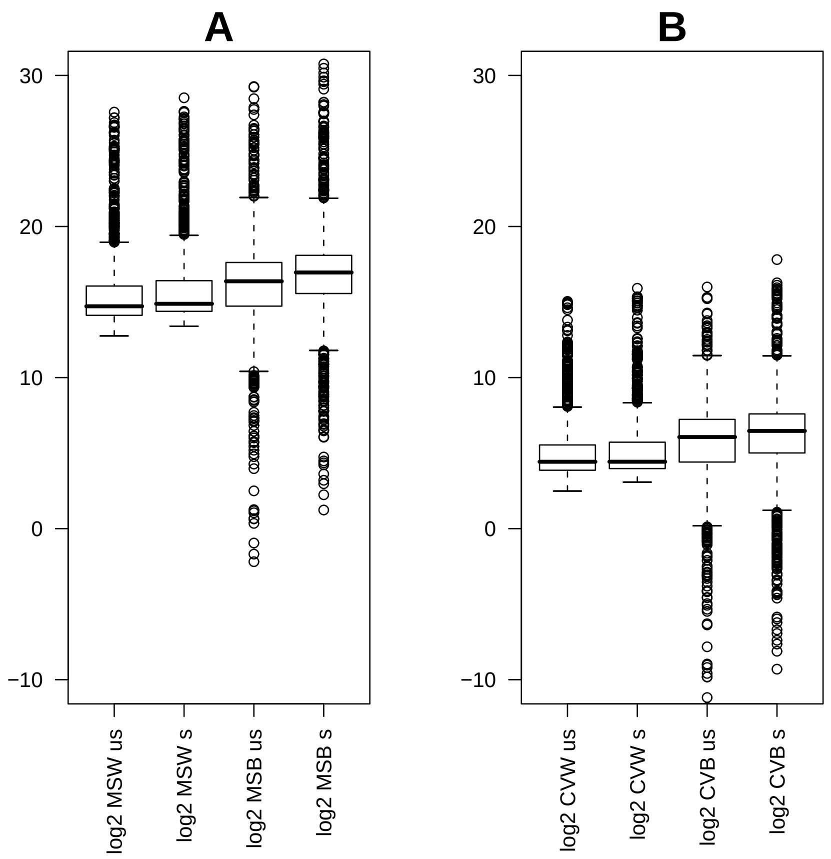

Figure 9: Within and between (replicate) array variability.
(**A**) log (base 2) of mean square error (MS) of bead types. (left to right) We show the $\log_2$ values of MS within arrays using unsharpened intensities, MS within arrays using sharpened intensities, MS between arrays using unsharpened intensities and MS between arrays using sharpened intensities.
(**B**) Coefficient of variation (CV) of bead types. (left to right) We show the CV within arrays using unsharpened intensities, CV between arrays using sharpened intensities, CV between arrays using unsharpened intensities and CV between arrays using sharpened intensities. Calculations were made on unlogged data, then the MS and CV statistics were transformed to the $\log_2$ scale.

4.5. Effect of sharpening on outliers

Since sharpening has been shown to increase the variability of beads of the same type, we might also expect sharpening to have an effect on the outliers on an array. To investigate this in more detail we first used the unsharpened ($\log_2$) bead intensities and calculated the MAD for each bead. This was repeated using sharpened intensities and the MADs before and after sharpening were plotted (see Figure 10). Most beads are seen to have MADs < 3 both with and without sharpening. However, some beads which have MADs > 3 without sharpening have MADs < 3 with sharpening (dark grey). In other words these beads were outliers without sharpening and the process of sharpening has made them no longer outliers. Similarly, some outliers are created by sharpening (light grey). The number of outliers created and removed by sharpening seems to be about the same. In general, we observe that beads have a lower MAD after sharpening. Further investigation revealed that out of the 1420 outliers detected on this array using sharpened intensities, 366 were removed by sharpening and 255 outliers were created by sharpening.

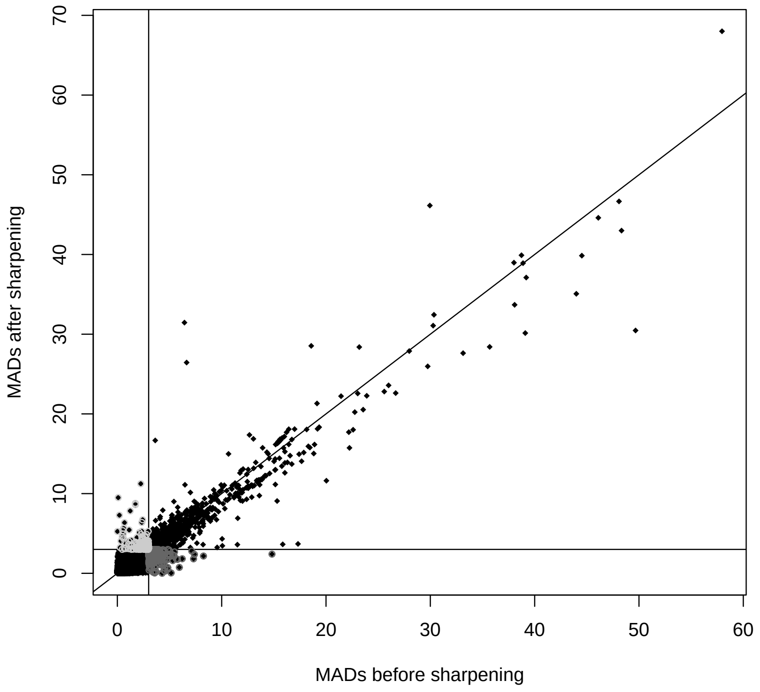

Figure 10: The effect of sharpening on outliers.
For all beads on an array we show the MAD of the bead calculated with and without using sharpening. The horizontal and vertical lines indicate 3 MADs (i.e. the cut-off that Illumina use to determine outlier beads). Dark grey spots indicate beads that are outliers without using sharpening but not outliers after sharpening whilst light grey spots indicate beads that are not outliers before sharpening but are outliers after sharpening.

In Figure 11 we show the total number of outliers on 12 arrays and how many outliers are created or removed by sharpening. It can be seen that the number of outliers created by sharpening is slightly higher than the number removed by sharpening. The majority of outliers for a particular array are unaffected by sharpening.

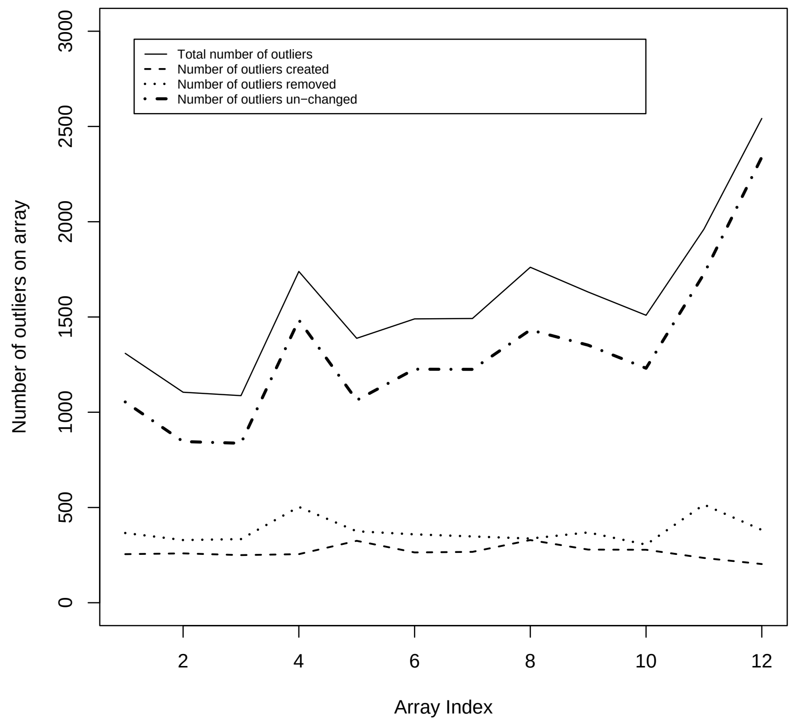

Figure 11: The effect of sharpening on outliers.
For 12 arrays we show the number of outliers which are removed and created by sharpening along with the total number of outliers on the array and outliers unchanged by sharpening.

4.6. The number of outliers on arrays

The number of outliers on each array is on the order of 1000–3000 beads; roughly one to two outliers per bead type or 1%–6% of total beads. These numbers are consistent across all arrays in the 96 array SAM and for all SAMs. In rare cases, as many as six or seven outliers were found for some bead types. However, only 0.5% of bead types on the 96 arrays had more than five outliers detected.

4.7. Outlier detection

As described in Section 3, *beadarray* offers a more flexible method for determining the outliers for a given bead type. We now use the sharpened, non-background corrected intensities for a particular bead type on an array to demonstrate this flexibility (see Figure 12). The number of outliers detected for a particular bead type is dependent on the choice of scale used (unlogged or $\log_2$). In Figure 12 it can be seen that this choice has little effect on the total number of outliers that are detected on an array.

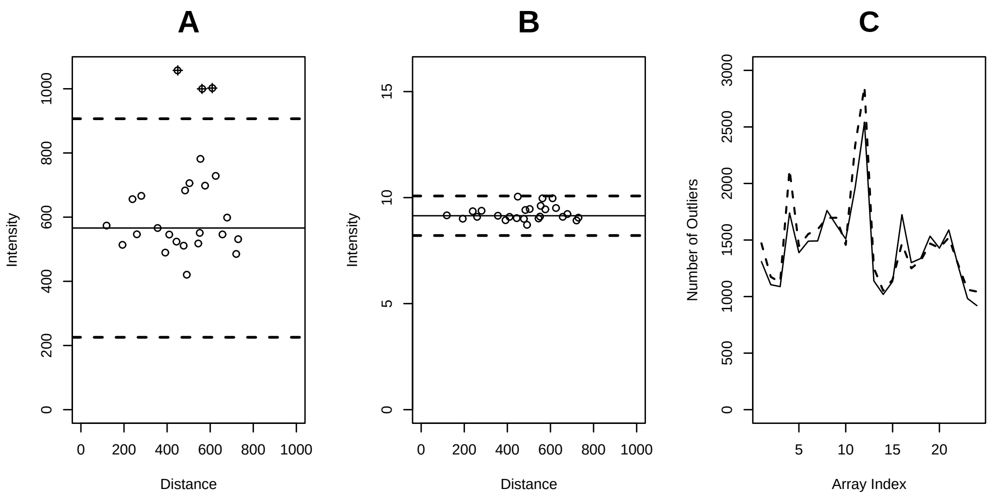

Figure 12: Outlier detection using *beadarray* and the effect of the choice of scale. (**A**) Unlogged intensities of all beads of a particular type on the same array are plotted on the y axis. The dotted horizontal lines represent a shift of 3 MADs from the mean (solid horizontal line). Beads outside the dotted lines are outliers for this bead type. Plotting the distance of each bead from the centre along the x axis allows for the possibility of identifying spatial effects on the array. (**B**) Intensities for the same bead type shown on the $\log_2$ scale. As before dotted lines represent a shift of 3 MADs from the mean. (**C**) The number of outliers detected in the sharpened, non-background corrected intensities on 24 arrays using either the $\log_2$ or unlogged scale.

4.8. Spatial plots of outliers

It is important to know where outliers are located on arrays as this can indicate possible spatial artifacts. Figure 3 shows a region on the far left tip of one of the arrays under investigation. It is quite apparent that this part of the array contains a significant spatial artifact. The consequence of this artifact is that beads lying within the affected area show an increased hybridisation level and are subsequently classified as outliers. Note that Figure 3 serves as an extreme example and spatial artifacts were only detected on a small number of arrays. Spatial artifacts were found more often around the edges of arrays and can be automatically identified through the *beadarray* package using the χ^2 statistic without systematically viewing each array in the experiment.

4.9. Effect of image processing on bead averages

We have previously shown that sharpening and background correction have the effect of increasing the SD of replicate beads. We now ask if this increase in SD among replicates has an effect on the resultant bead averages. In Figure 13 we plot the average $\log_2$ values (with outliers on the $\log_2$ scale removed) for four control probes across 12 arrays without sharpening, with sharpening and with both sharpening and background correction. Although the averages calculated using sharpened intensities and averages calculated using sharpened and background corrected intensities are lower, we can clearly see the same trend in all sets of averages. Controls A, B and C are housekeeping controls so show high intensity across all arrays. The effect of background correction on these averages seems minimal. The final control is a negative control and therefore we would expect it to have low intensity across all arrays. The averages calculated using background corrected intensities are much lower. As we saw previously, the effect of background correction is much greater on lower intensity beads.

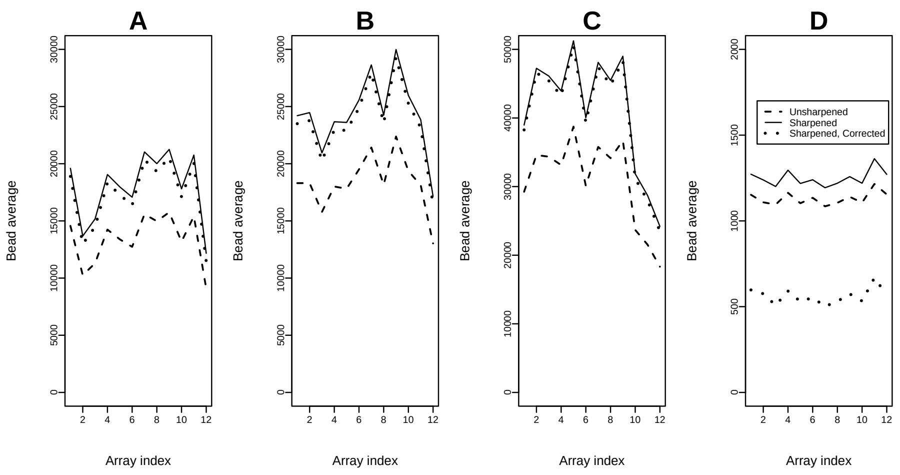

Figure 13: Effect of sharpening on bead summaries for particular control probes. Figures produced by *beadarray*.
The (unlogged) variation in bead-summary values calculated using unsharpened, sharpened and sharpened background corrected bead intensities for 12 arrays (x axes) in the same experiment. A, B and C show the results for three hybridisation controls respectively and D shows the results for a negative control across the 12 arrays.

For each bead type on each array we calculated the difference between the bead averages obtained using unsharpened and sharpened intensities (see Figure 14). This shows that the difference between the averages calculated using sharpened or unsharpened intensities is around 0.2. Furthermore, for particular bead types, the difference between averages is fairly constant across arrays (the SD is 0.01).

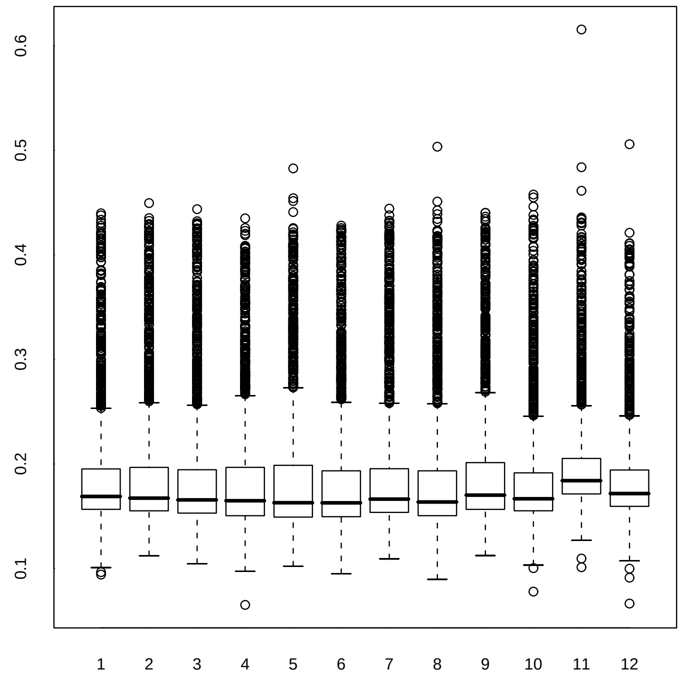

Figure 14: Effect of sharpening on all bead averages on an array. For 12 arrays we show the difference between the $\log_2$ bead averages calculated using unsharpened or sharpened intensities for every bead type on the array.

4.10. Variability between arrays

The generation of bead-summary data allows for conventional plots to be used to compare probe intensities between arrays. We used the bead-summary data from the first four arrays on the SAM (the first three of which are replicate arrays) to make MA and scatter plots (Figure 15) and density plots (Figure 16). Comparing replicate arrays allows us to observe both random and systematic sources of variation. In Figure 15, the majority of points lie along the diagonal for the scatter plots and along the central line for the MA plots. There is little noise in the MA plots between replicate arrays and only some intensity dependent bias is apparent (i.e. between Arrays 1 and 2). Comparisons involving the fourth array show more variation as this array is a different sample to the other three. In Figure 16 we see that the distribution of bead-summary intensities for these arrays are similarly shaped. Apart from the obvious need for location normalisation between these arrays, non-linear effects in the bead-summary intensities between them are minimal. Similar observations were made between arrays across the whole SAM. Comparing the average intensities of beads on the first array to all other 95 arrays gave a median correlation coefficient of 0.98.

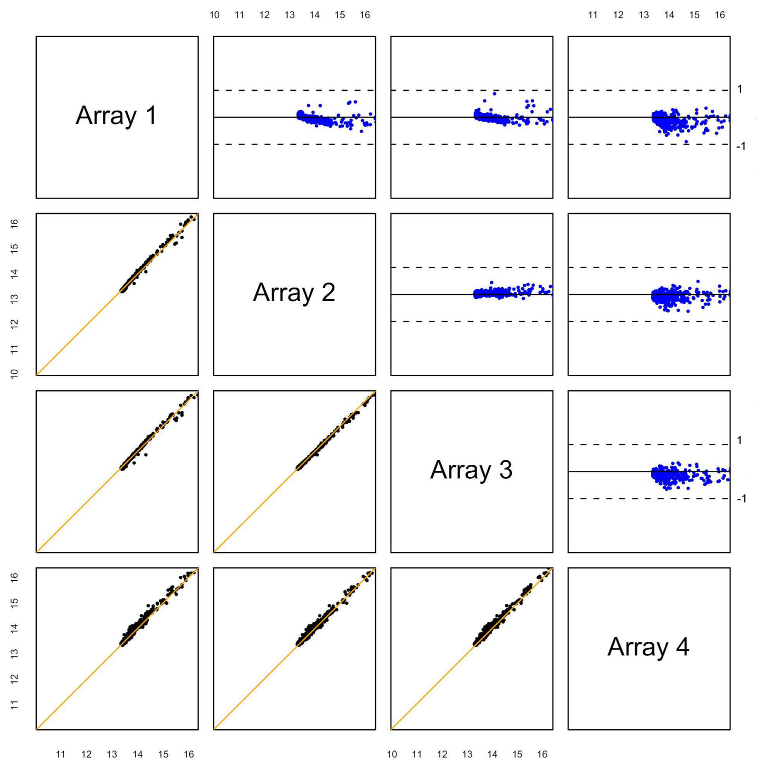

Figure 15: Comparing average bead-type intensities. Figures produced by *beadarray*. Scatter and MA-plots for the first four arrays on a SAM are shown. The intensities shown have been sharpened but not normalised or background corrected. The first three arrays are replicates of the same sample, hence display less variation.

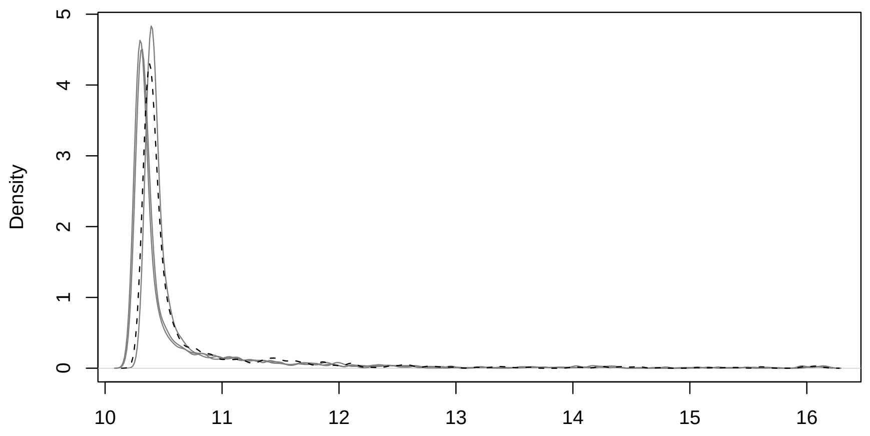

Figure 16: Comparing density plots.
We show the density plots of the bead averages for the four arrays shown in Figure 15.

5. DISCUSSION

Our preliminary investigations suggest a high degree of reproducibility with BeadArray^{TM} data. Arrays are seen to exhibit highly similar distributions even before any normalisation has taken place. The precision of replicates of the same bead type is high both within and between arrays. Given the low variability of BeadArrays there might be a danger of over-normalising the data and removing important biological information. Preliminary investigations suggest that a quantile or qspline [17] normalisation is sufficient (data not shown [16]) for bead-summary data. However, we feel that in general such normalisation options should be investigated and performed on the bead-level rather than bead-summary data.

Our main finding concerning image processing was that the sharpening transformation used prior to calculating foreground intensities causes an increase in variability. The transformation is designed in such a way that high intensity pixels (with respect to their neighbours) are made even higher and low intensity pixels are made lower. Therefore, it is not surprising to find intensities greater than 16 on the $\log_2$ scale after sharpening. However, it appears that the intensities of beads of the same type are being altered independently of each other and this causes an increase in variance. As a result of sharpening, we observed that the bead type averages increase by around 0.2 and their SD's increase by 0.01.

The background values for beads were found to be virtually constant within arrays and also across arrays. Correcting using the local background measure is effectively equivalent to using a global value. Background corrected data show much more variability among beads of the same type. Automatic background correction on BeadArray^{TM} data cannot therefore be recommended. *beadarray* does not perform correction automatically, thereby allowing foreground and background levels to be analysed separately. It should be emphasised that the bead-summary data produced by BeadStudio is automatically background corrected using the local background measures. Therefore any attempts to correct pre-processed bead-summary data (as given by the normalisation methods supplied by BeadStudio) may have an adverse effect.

The *beadarray* package can be used to highlight and understand problems that can occur with BeadArrays. We found random numbers and positioning of beads on all arrays. The random positioning minimises the effects of spatial artifacts on bead-summary data; such artifacts were rare. Due to the high replication of beads, any beads which occur inside such regions of unusual intensity are declared as outliers and can be removed from analysis without affecting the bead average values too much. The distribution of beads gives, on average, about 30 beads of each type. Depending on the scale of intensities used to detect outliers (either unlogged or $\log_2$) we might expect one or two of these beads to be detected as outliers. For the five SAMs in the investigation, no bead type on any array was found to have less than 11 replicates after outlier removal.

The number of outliers for each bead type can be detected by using either unlogged or $\log_2$ intensities. If we apply a $\log_2$ transformation to the data we decrease the range of the intensities. The purpose of such a transformation is to make changes in intensity comparable across the whole intensity range. Converting to the $\log_2$ scale also tends to make the variability more constant ([13]). Outliers which appear extreme on the unlogged scale will be much closer to the mean on the $\log_2$ scale. Therefore, it might be more consistent to use $\log_2$ intensities to calculate outliers if the intention is to use $\log_2$ intensities in analysis. In practice, the decision to use unlogged or $\log_2$ intensities to determine outliers had very little effect on the bead averages produced. Bead averages show very low variability across replicate arrays of the same sample.

We believe that *beadarray* offers a very flexible platform for the analysis of BeadArray$^{\text{TM}}$ data. By recreating bead-level data from scratch, users are given access to more information about each individual bead on an array. Making sharpening and background correction optional gives the opportunity to use diagnostic checks and make an informed choice about how data should be pre-processed. As the amount of BeadArray$^{\text{TM}}$ data available is relatively small we can only make recommendations for how data should be pre-processed based on our experience whilst developing the package. Analysing bead-summary data using *beadarray* offers a greater range of plotting tools than existing methods. Most importantly, all functions can deal with intensities on the $\log_2$ scale as is common for microarray analysis. Whilst *beadarray* does not currently provide any methods for detecting differential expression, these will be implemented in future versions. *beadarray* can also provide a more flexible analysis of bead-summary data pre-processed by Illumina.

In our study we have demonstrated how *beadarray* can be used for quality control and low-level analysis. We have presented some findings about the impact of the image processing steps used by Illumina on a particular experiment. The conclusions we reach in this paper may not indeed be valid in all cases and subsequent experiments will need to be analysed in a similar manner before more general conclusions can be reached. At the time of developing the package, only SAM data were available to us. We are currently expanding the package to include the analysis of BeadChip$^{\text{TM}}$ data. An investigation into normalisation methods will be linked with an implementation of methods for assessing differential expression. It will be of interest to see how sharpening and background correction affect the genes that are selected as differentially expressed.

ACKNOWLEDGMENTS

We thank Brenda Kahl and Semyon Kruglyak (Illumina), Barbara Stranger, Matthew Forrest and Manolis Dermitzakis (Wellcome Trust Sanger Institute) and Andrew Lynch (University of Cambridge) for many helpful discussions during the devlopment of this work. We also thank James Brenton (University of Cambridge) for useful advice on reading images into R. The authors were supported by grants from the US National Institutes of Health, Cancer Research UK and the Medical Research Council. Simon Tavaré is a Royal Society / Wolfson Research Merit Award holder.

REFERENCES

[1] Altshuler, D.; Brooks, L.D.; Chakravarti, A.; Collins, F.S.; Daly, M.J.; Donnelly, P. and the International HapMap Consortium (2005). A haplotype map of the human genome, *Nature*, **437**, 1299–1320.

[2] Barnes, M.; Freudenberg, J.; Thompson, S.; Aronow, B. and Pavlidis, P. (2005). Experimental comparison and cross-validation of the Affymetrix and Illumina gene expression analysis platforms, *Nucleic Acids Res.*, **33**, 5914–5923.

[3] The International HapMap Consortium (2003). The International HapMap Project, *Nature*, **426**, 789–796.

[4] Dudoit, S.; Yang, YH.; Speed, T.P. and Callow, M.J. (2002). Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments, *Statistica Sinica*, **12**, 111–140.

[5] Galinsky, V.L. (2003). Automatic registration of microarray images. II. Hexagonal grid, *Bioinformatics*, **19**, 1832–1836.

[6] Gentleman, R.C.; Carey, V.J.; Bates, D.M.; Bolstad, B.; Dettling, M.; Dudoit, S.; Ellis, B.; Gautier, L.; Ge, Y.; Gentry, J.; Hornik, K.; Hothorn, T.; Huber, W.; Iacus, S.; Irizarry, R.; Leisch, F.; Li, C.; Maechler, M.; Rossini, A.J.; Sawitzki, G.; Smith, C.; Smyth, G.; Tierney, L.; Yang, J.Y. and Zhang, J. (2004). Bioconductor: open software development for computational biology and bioinformatics, *Genome Biol.*, **5**, R80.

[7] Gunderson, K.L.; Kruglyak, S.; Graige, M.S.; Garcia, F.; Kermani, B.G.; Zhao, C.; Che, D.; Dickinson, T.; Wickham, E.; Bierle, J.; Doucet, D.; Milewski, M.; Yang, R.; Siegmund, C.; Haas, J.; Zhou, L.; Oliphant, A.; Fan, J.B.; Barnard, S. and Chee, M.S. (2004). Decoding randomly ordered DNA arrays, *Genome Res.*, **14**, 870–877.

[8] Gunderson, K.L.; Steemers, F.J.; Lee, G.; Mendoza, L.G. and Chee, M.S. (2005). A genome-wide scalable SNP genotyping assay using microarray technology, *Nat. Genet.*, **37**, 549–554.

[9] Kuhn, K.; Baker, S.C.; Chudin, E.; Lieu, M.H.; Oeser, S.; Bennett, H.; Rigault, P.; Barker, D.; McDaniel, T.K. and Chee, M.S. (2004). A novel, high-performance random array platform for quantitative gene expression profiling, *Genome Res.*, **14**, 2347–2356.

[10] Oliphant, A.; Barker, D.L.; Stuelpnagel, J.R. and Chee, M.S. (2002). BeadArray technology: enabling an accurate, cost-effective approach to high-throughput genotyping, *Biotechniques*, **Suppl.**, 56–58, 60-61.

[11] Smyth, G.K. (2004). Linear models and empirical Bayes methods for assessing differential expression in microarray experiments, *Statistical Applications in Genetics and Molecular Biology*, **3**, 113–136.

[12] Smyth, G.K. (2005). *Limma: linear models for microarray data.* In "Bioinformatics and Computational Biology Solutions using R and Bioconductor" (R. Gentleman, V. Carey, W. Huber, R. Irizarry and S. Dudoit, Eds.), Springer, New York, 397–420.

[13] Smyth, G.K.; Michaud, J. and Scott, H.S. (2005). Use of within-array replicate spots for assessing differential expression in microarray experiments, *Bioinformatics*, **21**, 2067–2075.

[14] Smyth, G.K. and Speed, T. (2003). Normalization of cDNA microarray data, *Methods*, **31**, 265–273.

[15] Steinberg, G.; Stromsborg, K.; Thomas, L.; Barker, D. and Zhao, C. (2004). Strategies for covalent attachment of DNA to beads, *Biopolymers*, **73**, 597–605.

[16] Stranger, B.E.; Forrest, M.S.; Clark, A.G.; Minichiello, M.J.; Deutsch, S.; Lyle, R.; Hunt, S.; Kahl, B.; Antonarakis, S.E.; Tavaré, S.; Deloukas, P. and Dermitzakis, E.T. (2005). Genome-wide associations of gene expression variation in humans, *PLoS Genet.*, **1**(6), e26.

[17] Workman, C.; Jensen, L.J.; Jarmer, H.; Berka, R.; Gautier, L.; Nielser, H.B.; Saxild, H.H.; Nielsen, C.; Brunak, S. and Knudsen, S. (2002). A new non-linear normalization method for reducing variability in DNA microarray experiments, *Genome Biol.*, **3**, research0048.

6. APPENDIX

The purpose of this appendix 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 the Vignette distributed with *beadarray*. 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`

6.1. 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.

Before these files can be read into R, we first convert the TIFF files into PGM files. This can be done be using the *ImageMagick* utility[1]. A batch file is included with this library to convert automatically all the TIFF files in a directory.

If the correct csv and pgm files are available we can read these data into R using `readBeadImages`. This function requires a `beadTargets` object that can be read directly from a beadTargets.txt file. This `beadTargets` object specifies the filename of each image and csv file to be read.

In our example dataset we have two single channel BeadArray™ hybridisations. Since the arrays are hybridised with one target only (one-colour), we need only to specify one image file for each array. The beadTargets.txt file, pgm and csv files are provided at `http://www.damtp.cam.ac.uk/user/jcm68/beadarray.html`. Once downloaded, these files can be read into R. The R working directory can be set to the folder containing these files using the `setwd` function or the file menu (GUI implementation only). Alternatively the *path* argument in `readBeadImages` can be set to the current directory. The commands to read the data into R are then as follows:

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

               Image1                 xyInfo SAMPLE
1269941_R001_C001.pgm 1269941_R001_C001.csv      6
1269941_R001_C002.pgm 1269941_R001_C002.csv      6

> BLData = readBeadImages(beadTargets)

Calculating foreground intensities for 1269941_R001_C001.pgm
Calculating background intensities.
Calculating foreground intensities for 1269941_R001_C002.pgm
Calculating background intensities.
```

[1]available from `www.imagemagick.org` — version 6.2.2 or later is required.

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 section 3). To create unsharpened bead intensities one would use:

```
> BLData.ns = readBeadImages(beadTargets, sharpen = FALSE)
```

6.2. The BLData 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"
 [4] "y"                   "probeID"             "targets"
 [7] "sharpened"           "backgroundSize"      "normalised"
 [10] "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$^{\text{TM}}$ 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.

6.3. Background correction and normalisation

Using the `boxplot` function in R allows boxplots of foreground and background intensities to be compared (see Figure 6).

```
> boxplot(log2(BLData$R)~col(BLData$R))
```

Background correction can be be performed on the data by:

```
> BLData.c = backgroundCorrectBeads(BLData)
```

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.

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

6.4. Numbers of beads

Figure 5A can reproduced using the command

```
> histBeadCounts(BLData, array=1)
```

Additonally we can also 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
```

For clarity only the first 10 results returned by the function are shown.

6.5. Outliers for each bead type

The `plotBeadIntensities` function was used to produce Figure 12. 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, probe=2, array=1)
> plotBeadIntensities(BLData, probe=2, array=1, log = TRUE)
```

The function `findOutliers` is used within `plotBeadIntensities` to determine the outliers for a particular bead type on an array. The function `findMostOutliers` can be used to find which bead types have more than a set number of outliers (the default is 5 outliers).

```
> findMostOutliers(BLData, array=1)
```

```
 [1]  807 1702 2458 5244 5917 6015 6117
```

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 between 1 and 49777 which refers to a particular bead on the array (beadID).

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

```
 [1] 44634  1263  8245   342 23176  6270  8898 31023  4273 15610
```

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.

6.6. 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 a list of ProbeIDs or beadIDs. Figure 2A can be generated by using:

```
> 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 Section 3.2, 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.

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

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.

6.7. 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 Section 3.3. 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 supplied with *beadarray*.