1 Introduction

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. The methods available for processing bead level data are discussed in: Dunning,M.J et al, Quality Control and Low-level Statistical Analysis of Illumina Beadarrays, Submitted.

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. As such the methods described within this document focus exclusivly on dealing with bead summary data. The bead summary data can be data obtained using either the BeadChip or SAM technologies. This document uses a SAM experiment as an example although BeadChip can be read in the same manner.

See References for further reading on BeadArray technology.

2 Reading and analysing bead summary data

Pre-processed bead summary data can be read directly into our library. The function used is `readBeadSummaryData` and requires a vector giving the names of text files as an input. Each text file can describe the bead summary data for a particular array in the experiment, or it may describe all arrays. Example files are provided at the following URL and each file describes a single array from a SAM experiment

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

Once the example files have been downloaded they can be read into R. By default, the files are read from the current working directory in R, but this can be changed by setting the *path* parameter. The `targets` object is used to define a vector of filenames to be read and is created by reading the beadSummaryTargets.txt file (see example file). For experiments with a large number of arrays, it may be inconvenient to create the text file. Therefore, the `readBeadSummaryData` is able to read all files in the working directory if the *targets* parameter is omitted. By default, the function looks for each of the column headings as they are listed below. It is possible to use alternative names for headings (eg nobeads instead of NoBeads). For more details on this please see the appropriate help file.

- ProbeID - an each identifier for each bead type (probe type) on the array
- AvgSig - Summary intensity produced by averaging bead intensities of all beads of a particular type
- BeadStDev - Standard deviation of all beads of a particular type, outliers excluded
- NoBeads - Number of beads used to produce average
- Detection - Average detection score for each bead type

Usage of `readBeadSummaryTargets` is as follows:

```
> targets = readBeadSummaryTargets()
> BSData = readBeadSummaryData(targets)
> BSData
```

The resulting object, a *BeadSummaryList*, is a list based object with the following sublists:

- R - averaged foreground intensities
- Rb - averaged background intensities
- probeID - unique identifer for the probe
- beadstdev - standard deviation of all beads of a particular type
- nobeads - number of beads used to produce average
- Detection - average detection score for each bead type

The first 6 entries in a *BeadSummaryList* are all matrices with the each row corresponding to a particular bead type and the columns to individual arrays.

A common cause of error when reading files is for the column names found in the files to not match the headings that R expects to find. The `readBeadSummaryTargets` function prints out the name of the file being read and also the first line of the file. This shows the column names present in the file and will help to identify problems. The columns parameter is used to change will column headings to look for in the input file. For example, if the column headings are TargetID, AVG_Signal, BEAD_STDEV, Avg_NBEADS, Detection

```
> BSData = readBeadSummaryData(columns = list(ProbeID = "TargetID",
+     AvgSig = "AVG_Signal", BeadStDev = "BEAD_STDEV", Nobeads = "Avg_NBEADS",
+     Detection = "Detection"))
```

In the example bead summary file, each file gives data for a seperate array in a experiment. It is also possible to read files containing data for more than one array using `readBeadSummaryData`. Files of this type are assumed to have a number of rows equal to the number of bead types in the experiment (eg 1500 for SAM or 24,000 for BeadChip) and the same columns for each array. The column headings are assumed to be of the form AVG_Signal-1, AVG_Signal-2,...AVG_Signal-n for n arrays. This is the standard output produced by BeadStudio. See below for a screenshot of an exampe file containing two arrays.

	A	B	C	D	E	F	G	H	I
1	TargetID	AVG_Signal-1	BEAD_STDEV-1	Detection-1	Avg_NBEADS-1	AVG_Signal-2	BEAD_STDEV-2	Detection-2	Avg_NBEADS-2
2	GI_10047089-S	106.4	4.5	0.33644494	35	80.9	3.3	0.95437491	39
3	GI_10047091-S	190.5	10.4	0.99998383	38	130.8	7.5	1	47
4	GI_10047093-S	675	33.4	1	37	546.8	21.6	1	42
5	GI_10047097-S	432.6	16.9	1	43	323	14.1	1	44
6	GI_10047099-S	664	19.3	1	30	710.2	25.8	1	35
7	GI_10047103-S	3108.8	71.8	1	38	2214.1	76.7	1	41
8	GI_10047105-S	146.9	11.7	0.96259936	30	85.5	7	0.98882627	48
9	GI_10047115-S	1945.5	77.8	1	49	1436.9	47.6	1	59
10	GI_10047117-S	140.8	8	0.92632108	52	109.5	6.3	0.99999996	45
11	GI_10047121-S	105.5	5.2	0.3185542	32	63.9	5	0.31269067	38
12	GI_10047123-S	418.9	15.8	1	50	278.5	9.4	1	43
13	GI_10047133-A	158.8	5.7	0.99239624	40	97.2	5.9	0.99991986	40
14	GI_10047133-I	129.5	10.5	0.79732306	38	78.3	4.2	0.91234824	36
15	GI_10048402-S	88.8	5.4	0.08345505	51	70.9	5.3	0.660507	46
16	GI_10092578-S	113.1	7	0.47603397	38	73.9	5.6	0.78504785	58
17	GI_10092585-S	161.1	9.2	0.99473016	39	93.5	9.7	0.99952363	39
18	GI_10092596-S	180.8	7.3	0.99985835	34	126.9	6.6	1	38
19	GI_10092602-S	155.9	9.8	0.98839642	44	85.6	5.4	0.98899854	26
20	GI_10092603-S	144.1	7.7	0.94810805	44	74.6	5.4	0.81219389	39
21	GI_10092611-A	270.3	7.2	1	45	152.6	7.7	1	46
22	GI_10092616-S	235.9	11.2	1	59	149.1	7.9	1	63
23	GI_10092618-S	685.5	25.6	1	51	590.3	19.2	1	41

All functions described from now on use a *BeadSummaryList* object as a parameter. This object can either be created using the steps described above, or can be created from bead-level data by using the `createBeadSummaryData` function (see Dunning et al).

We can use the detection score as a preliminary indicator of the quality of each array. Using the following commands we can construct a boxplot of the detection scores for each probe. If the scores look low then it would advisable to take a closer look at the array in order to identify a reason, e.g. a bad hybridization. Boxplots of the `R` and `Rb` columns can also be informative.

```
> boxplot((BSData$Detection) ~ col(BSData$Detection), main = "Detection Scores")
```

3 Plotting Values Across Whole SAMs and Use of Control Information

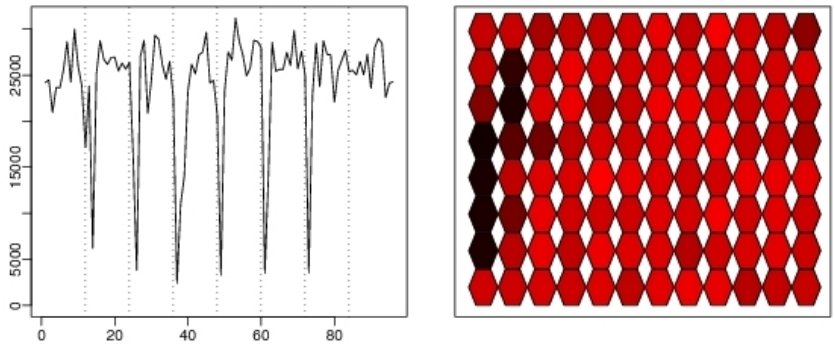
The `plotProbeVariation` function can be used to plot the variation in a particular probe / bead type on all arrays in an experiment. The input to the function is simply an *BeadSummaryList* object, and a ProbeID. The result is not very interesting for this data set as we only have 4 arrays.

```
> plotProbeVariation(BSData, ProbeID = 2)
```

We have provided a SAM summary plot for when we want to compare the intensity of a particular bead type across all arrays on one SAM. In its most simple form we use `plotOnSAM(v)` where `v` is simply a vector of numeric values with length 96. To create the vector `v` we could make use of the function `getMeanIntensities(BSData, probe)` which will return the mean intensity of beads with probeID on each of the arrays in the experiment. For instance probeID 4279 is a housekeeping control in the example experiment, so we can do.

```
> getMeanIntensities(BSData, 4279)
```

As an example we show the variation in a hybridisation control across 96 arrays



On the left-hand plot we have array index on the x axis and on the y axis we have the corresponding value of v for each of the 96 arrays. Instantly we can see the numbers of arrays for which the value of `v` is lower.

On the right-hand plot we relate the vector `v` to the position of each array on the SAM. The arrays are numbered from 1 in top-left corner to 96 in the bottom-right corner. The colour of each hexagon is related to its value of `v`, the higher this value the brighter the shade of red (a greyscale version of the plot can also be made).

In the figure above we can see that the line in the left hand plot is very erratic and the colour of the hexagons range from black through to bright red. Both of these indicate that the values in vector `v` change greatly across the SAM. Using the right hand plot we can quickly identify which probes have the lowest intensities allowing us to easily go and investigate the possible reasons. The BeadStudio application provided by Illumina is able to produce the plot seen in the left panel of the `plotOnSAM` function output, but we feel that our method is more flexible. With our function we can plot values of any probe (not just controls) and can plot intensities on both raw and logged scale. We can also see whereabouts any potential problem arrays are located on the SAM. In the examples above we found the values of a particular bead type across all arrays and used as input to `plotOnSAM`. This plotting function is flexible because it allows any vector of numeric values with length 96 as input. For instance we could also use the number of outliers on each of the 96 arrays or the number of unregistered beads as input.

4 Comparing Samples

Now that all the arrays in the experiment have been averaged, we can see how particular bead types vary between different arrays or samples. We have implemented both XY plots and MA plots to achieve this and these can be viewed simultaneously for a series of arrays (the MAXY plot). In an XY plot, for a particular gene, we simply plot the value obtained from two different samples against each other with one sample on the x axis and one sample on the y axis. For an MA plot we plot the average intensity of each gene from the two different samples against the difference. For conventional microarrays, the MA plot can often reveal important differences between the two dyes used for hybridisation and give us an idea of the amount of noise generated by experiments.

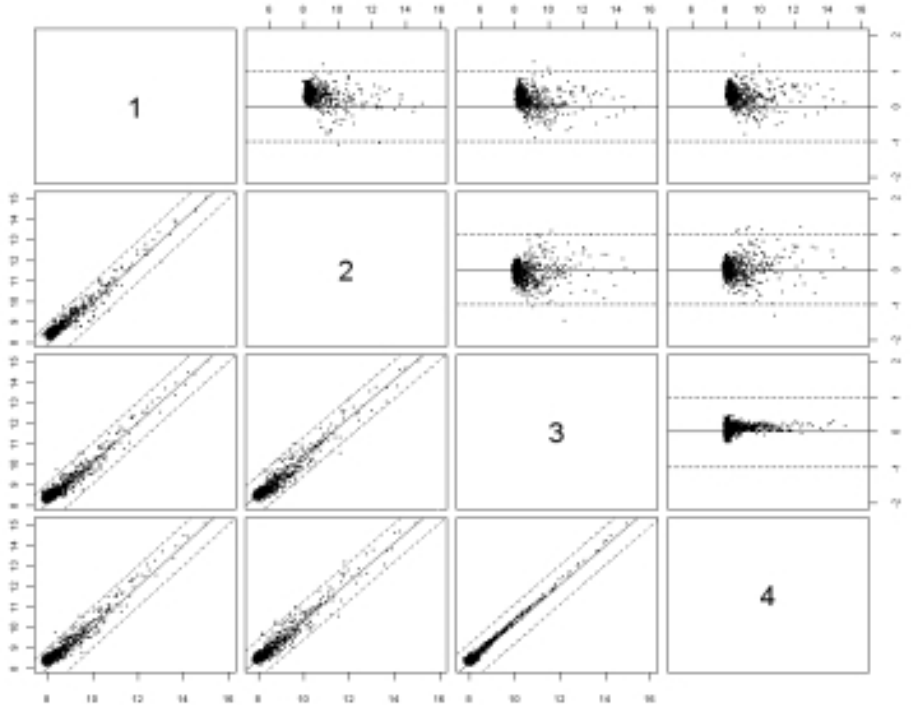
The functions created for the library are capable of making comparisons be-

tween the red and green channels for the same array as well as between two arrays from a one-colour experiment. However, we did not have any two colour data when creating this document, so the following examples will be for comparing two different arrays from a one-colour experiment.

We might want to know about the variation between replicates of the same sample in an experiment. The function `pairwiseMA` is capable of producing XY plots and MA plots for a defined set of arrays with XY and MA plots shown for all pairwise comparisons between the arrays

In our experiment arrays 1,2,3 & 4 were replicates of the same sample.

```
> vec = c(1, 2, 3, 4)
> plotMAXY(BSData, vec)
```



The resulting graphic is in a 4 x 4 grid. In the first row we have MA-plots of the first array compared to arrays 2,3 and 4 and in the first column there are XY plots of the first array compared to arrays 2,3 and 4.

As we are comparing replicates of the sample we would expect to see very little variation in the plots so XY plots should be centred around the diagonal and MA plots about the horizontal

The normalisation methods used for bead-level data (median and quantile), can also be applied to bead summary data. An additional normalisation method exists for the normalisation of bead summaries. This is known as *qspline normalisation*([7]) and uses quantiles from each array and a target distribution to fit a smoothing function to each array.

```
> BSData.qsp = qsplineNormalise(BSData)
```

The `plotMAXY` function can be used to assess the performance of normalisation methods on the data.

References

[1] GUNDERSON K., KRUGLYAK S, GRAIGE S., GARCIA F., KERMANI BG., ZHAO C., CHE D., DICKINSON T., WICKHAM E., BIERLE E., ET AL. (2004). Decoding randomly ordered DNA arrays, *Genome Research*, **14**, 870-877.

[2] OLIPHANT A., BARKER D., STUELPNAGEL J., CHEE M. (2002). BeadArray Technology: Enabling an Accurate, Cost-Effective Approach to High-Throughput Genotyping, *Biotechniques*, **14**, 870-877.

[3] KUHN K., BARKER S., CHUDIN E., LIEU M., OESER S., BENNETT H., RIGAULT P., BARKER D., MCDANIEL T., CHEE M. (2004). A novel, high-performance random array platform for qualitative gene expression profiling *Genone Research*

[4] STEINBERG G., STROMSBORG K., THOMAS L., BARKER D., ZHAO C. (2004). Strategies for Covalent Attachment of DNA to Beads *Biopolymers* **73** 597–605

[5] GUNDERSON K., STEEMERS FJ., LEE G., MENDOZA LG,. CHEE M. (2005) A genome-wide scalable SNP genotyping assay using microarray technology *Nature Genetics* **5** 549–554

[6] BARNES M., FREUDENBERG J., THOMPSON S., ARONOW B., PAVLIDIS P. (2005). Experimental comparison and cross-validation of the Affymetrix and Illumina gene expression analysis platforms *Nucelic Acids Research* **33** 5914–5923

[7] WORKMAN C., JENSEN L., JARMER H., BERKA R., GAUTIER L., NIELSER H., SAXLID H., NIELSEN C., BRUNAK S., KNUDSEN S. (2002). A new non-linear normalization method for reducing variability in DNA microarray experiments *Genome Biology* **3**