Analysis of Bead Summary Data using beadarray

August 29, 2006

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*, Revstat **4**, 1-30.

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. This is the format of the data output by Illumina's BeadStudio application.

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 (6 or 8 arrays on a slide) or SAM (arrays organised in 96 well plates) technologies.

1 Citing beadarray

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

Dunning M, Smith M, Thorne NP, Tavaré, *beadarray: An R package to Analyse Illumina BeadArrays*, R News, submitted

2 Importing Bead Summary Data

An example data set is included with the beadarray package and can be found in the BeadSummaryExample folder. Inside this folder you will find three Excel files that have been produced by BeadStudio and one text file. The Excel files contain the raw data, a sample sheet and a quality control file for the experiment. These data were obtained as part of a pilot study into BeadArray technology and comprises of 3 Human-6 BeadChips with 6 different samples, I, MC, MD, MT, P and Norm hybridised. MC, MD, MT and P are all tumours whereas Norm is a normal sample and I is a sample provided by Illumina.

2.1 Description of Files

- raw_data.csv - This contains the raw, non-normalised bead summary values as output by BeadStudio and is readable by Excel. Inside the file are several lines of header information followed by a data matrix with some 48,000 rows. Each row is a different gene in the experiment and the columns give different measurements for the gene. For each array, we record the summarised expression level (AVG_Signal), standard error of the bead replicates (BEADSTDEV), Number of beads used (Avg_NBEADS) and a Detection score which estimates the probability of a gene being

detected above the background. Note that whilst this data has not been normalised, it has been subjected to local background correction at the bead level prior to summarising.

- raw_data_sample_sheet - Defines the array IDs and samples placed on each array.
- raw_data_qc_info - Gives the summarised expression values for each of the controls that Illumina place on arrays and hence extremely useful for diagnostic purposes.

The following code can be used to read the example data into R

```
> targets <- readBeadSummaryTargets("targets.txt")
> targets

      DataFile               SampleSheet              QCInfo
1 raw_data.csv raw_data_sample_sheet.csv raw_data_qcinfo.csv

> BSData <- readBeadSummaryData(targets)
```

3 The BSData object

BSData is an object of type ExpressionSetIllumina which is an extension of the ExpressionSet class developed by the Biocore team used as a container for high-throughput assays. The data from the the raw_data file has been written to the assayData slot of the object, whereas the phenoData slot contains information from sample_sheet. For consistency with the definition of other *ExpressionSet* objects, we now refer to the expression values as the *exprs* matrix which can be accessed using `exprs` and subset in the usual manner. The `BeadStDev` matrix can be accessed using `se.exprs`. The rows of `exprs` are named according to the row names of the original raw_data file.

```
> BSData

Instance of ExpressionSetIllumina

assayData
  Storage mode: list
  featureNames: GI_10047089-S, GI_10047091-S, GI_10047093-S, ..., thrB, trpF (47293 total)
  Dimensions:
        exprs BeadStDev NoBeads Detection
Rows    47293     47293   47293     47293
Samples    18        18      18        18

phenoData
  sampleNames: IH.1, IC.1, IH.2, ..., Norm.2, P42.2 (18 total)
  varLabels:
    Sample_Name: Sample_Name
    Sample_Well: Sample_Well
    Sample_Plate: Sample_Plate
    Sample_Group: Sample_Group
    Pool_ID: Pool_ID
    Sentrix_ID: Sentrix_ID
    Sentrix_Position: Sentrix_Position

Experiment data
  Experimenter name:
```

```
  Laboratory:
  Contact information:
  Title:
  URL:
  PMIDs:
  No abstract available.

Annotation [1] "Illumina"
```

```
> exprs(BSData)[1:10, 1:2]
```

```
                IH.1   IC.1
GI_10047089-S   87.8  131.8
GI_10047091-S  161.8  130.8
GI_10047093-S  481.2  401.4
GI_10047099-S  633.7  483.8
GI_10047103-S 1535.6 1186.5
GI_10047105-S  247.5  210.2
GI_10047121-S  113.0  101.3
GI_10047123-S  453.9  306.8
GI_10047133-A  103.6  114.5
GI_10047133-I  118.0  123.1
```

```
> se.exprs(BSData)[1:10, 1:2]
```

```
              BEAD_STDEV.IH.1 BEAD_STDEV.IC.1
GI_10047089-S             5.1             9.5
GI_10047091-S            12.0             7.9
GI_10047093-S            21.7            24.5
GI_10047099-S            21.6            20.9
GI_10047103-S            42.7            34.5
GI_10047105-S            12.7            11.8
GI_10047121-S             6.4             8.1
GI_10047123-S            14.0            13.1
GI_10047133-A             6.8             6.0
GI_10047133-I             5.6             7.2
```

```
> pData(BSData)[, 1:6]
```

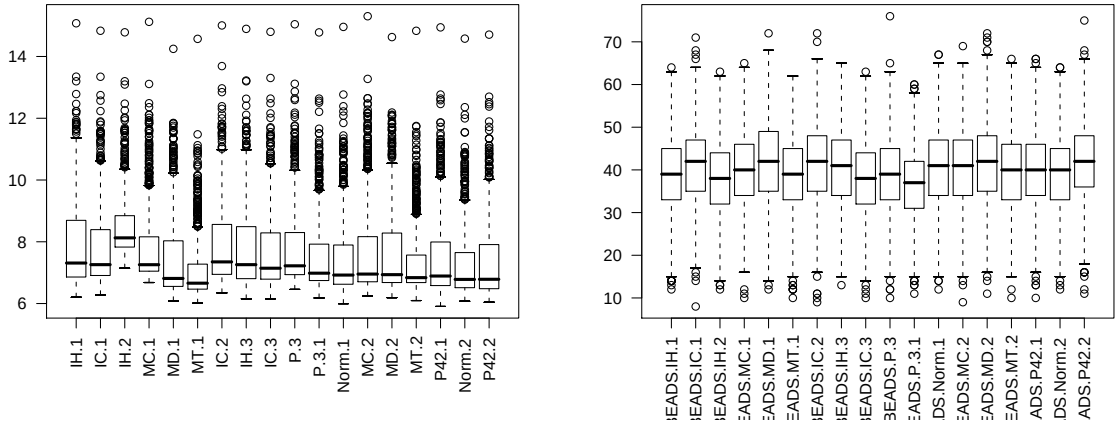
```
       Sample_Name Sample_Well Sample_Plate Sample_Group Pool_ID Sentrix_ID
IH.1            NA          NA           NA         IH-1      NA 1475542114
IC.1            NA          NA           NA         IC-1      NA 1475542114
IH.2            NA          NA           NA         IH-2      NA 1475542114
MC.1            NA          NA           NA         MC-1      NA 1475542114
MD.1            NA          NA           NA         MD-1      NA 1475542114
MT.1            NA          NA           NA         MT-1      NA 1475542114
IC.2            NA          NA           NA         IC-2      NA 1475542110
IH.3            NA          NA           NA         IH-3      NA 1475542110
IC.3            NA          NA           NA         IC-3      NA 1475542110
P.3             NA          NA           NA          P-3      NA 1475542110
P.3.1           NA          NA           NA          P-3      NA 1475542110
Norm.1          NA          NA           NA       Norm-1      NA 1475542110
MC.2            NA          NA           NA         MC-2      NA 1475542113
```

```
MD.2        NA        NA        NA        MD-2      NA 1475542113
MT.2        NA        NA        NA        MT-2      NA 1475542113
P42.1       NA        NA        NA         P-1      NA 1475542113
Norm.2      NA        NA        NA      Norm-2      NA 1475542113
P42.2       NA        NA        NA         P-2      NA 1475542113
```

Boxplots of expression may be useful for quality control. Recall that there are 6 arrays per BeadChip. We can see that the first BeadChip seems to be more variable than the others and in particular the third array on the first BeadChip could be an outlier.

Boxplots of the other slots in BSData can be easily plotted.

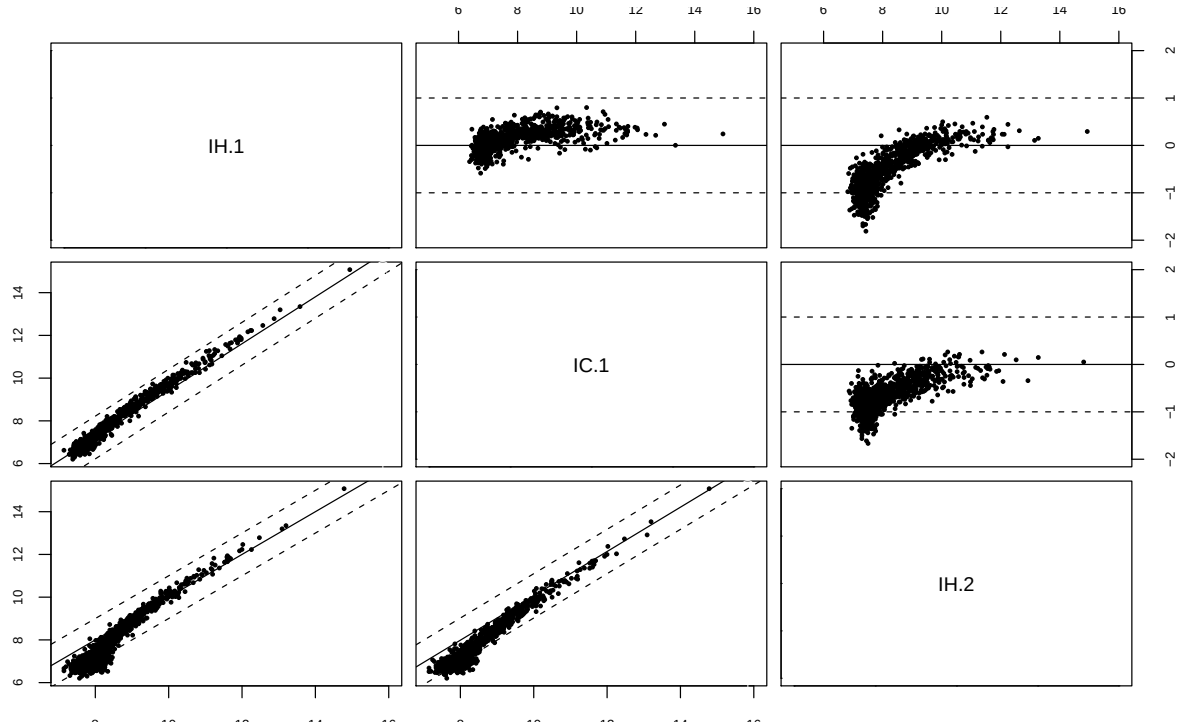
```
> par(mfrow = c(1, 2))
> boxplot(log2(exprs(BSData)[1:1000, ]), las = 2)
> boxplot(BSData@assayData$NoBeads[1:1000, ], las = 2)
```



4 Normalisation and Quality Control

In the expression boxplots we notice that there are differences in expression level across a chip and between chips. Therefore we might want to normalise the arrays in the experiment comparable. We also see the the 3rd array has significantly different intensity. The sample on this array is replicated three times on the first chip in the experiment, so comparing the MA and XY plots for the sample can be informative about the variability of this sample.

```
> plotMAXY(exprs(BSData)[1:1000, ], vec = 1:3, labels = colnames(exprs(BSData))[1:3])
```

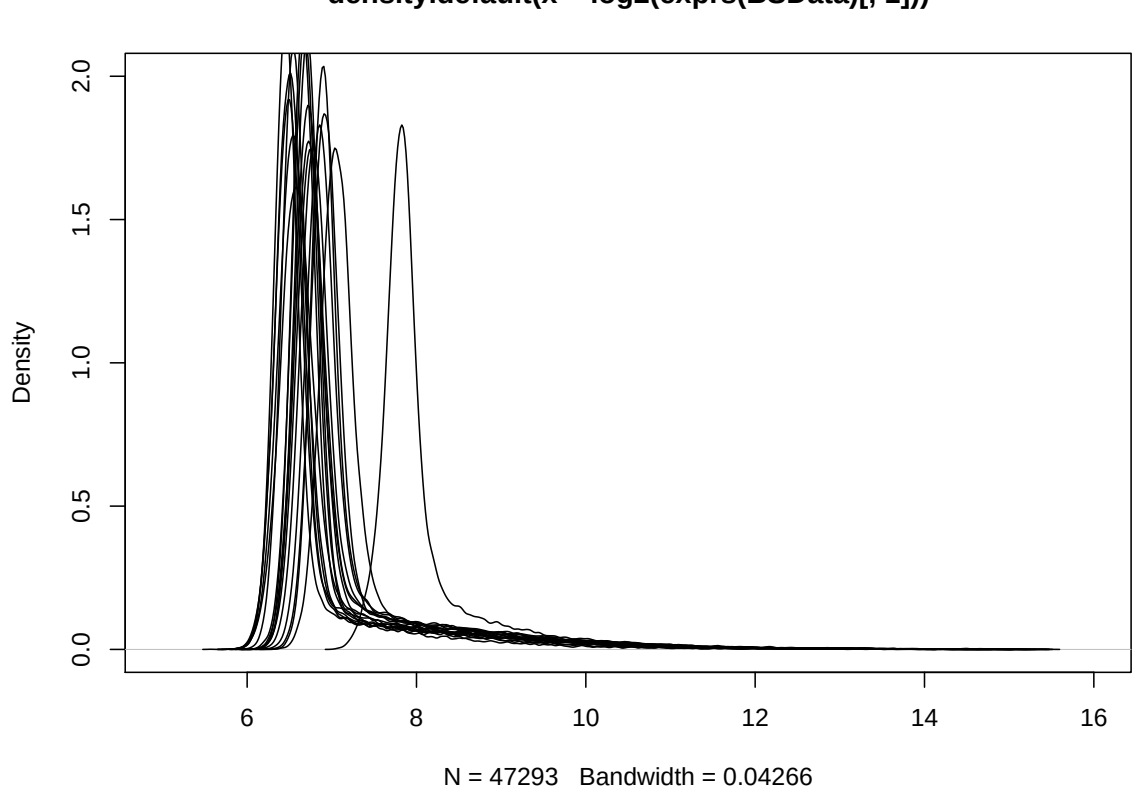


In the top right corner we see the MA plots for all pairwise comparisons involving the 3 arrays. On an MA plot, for each gene we plot the average of the expression levels on the two arrays on the x axis and the difference in the measurements on the y axis. For replicate arrays we would expect all genes to be unchanged between the two samples and hence most points on the plot to lie along the line y=0. In the lower left corner of the MAXY plot we see the XY plot and for replicate arrays we would expect to see most points along the diagonal $y = x$. From this MAXY plot it is obvious that the third array is significantly different to the other replicates and should probably be discarded.

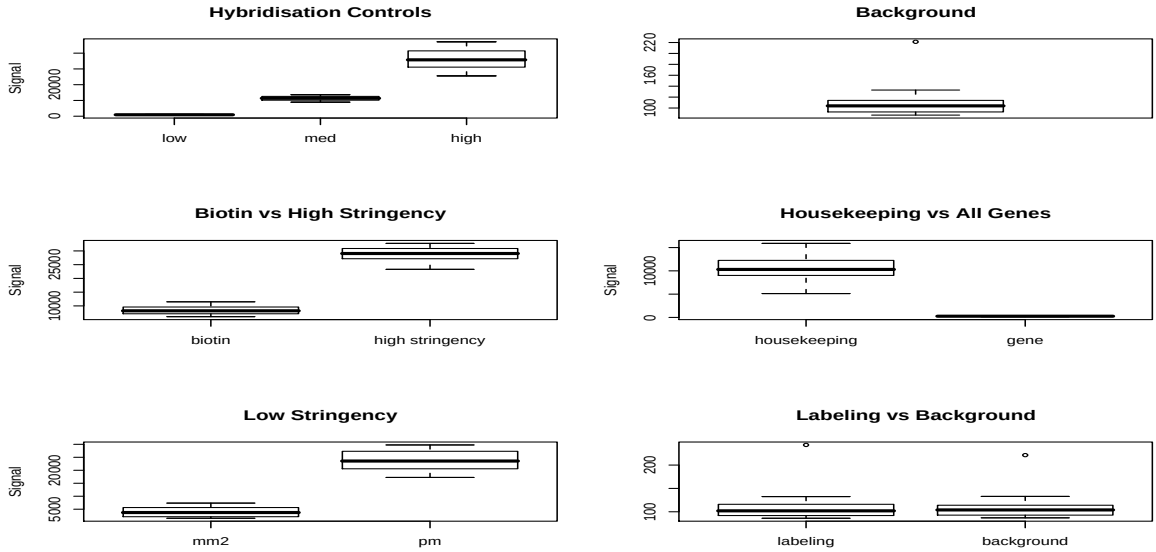
Quality control information can be read using the readQC function and plotted in a similar way to BeadStudio. The plotQC function displays an overview plot of all the controls types similar to functionality available in BeadStudio. The `QC` object itself is an *assayData* object with Signal, Var and Detection matrices.

Density plots of all arrays in the experiment can be used to motivate the need for normalisation. In the following we see that the distributions of the arrays are generally comparable.

```
> plot(density(log2(exprs(BSData)[, 1])), xlim = range(5, 16),
+     ylim = range(0, 2), type = "n")
> for (i in 1:18) {
+     lines(density(log2(exprs(BSData)[, i])))
+ }
```

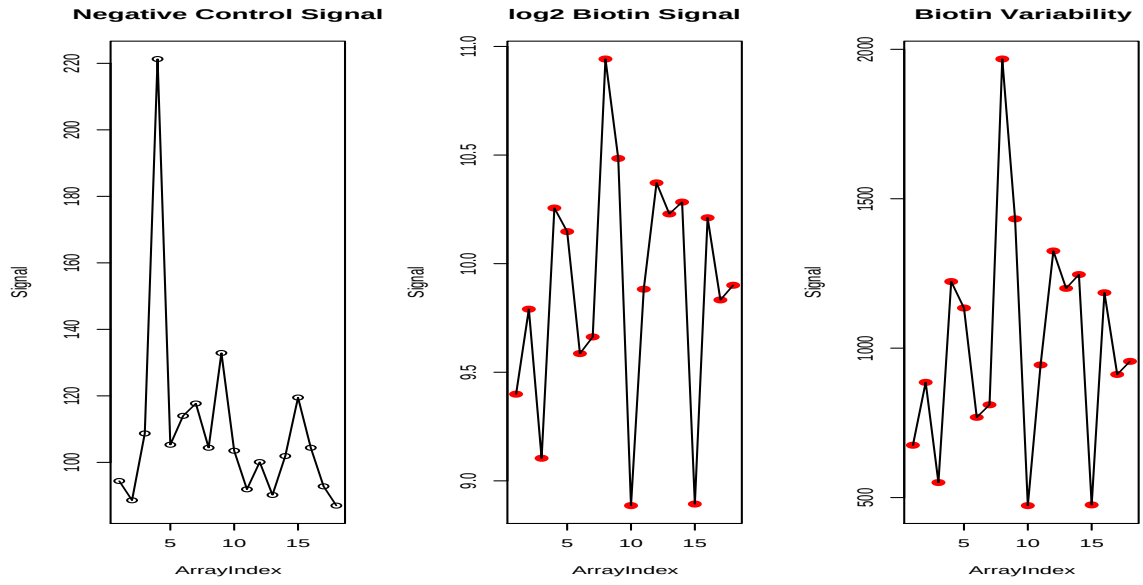



```
> QC = readQC(targets[1, 3])
> QC$Signal[1:3, ]

              Biotin cy3_high cy3_low cy3_med  gene      hs   house labeling
1475542110_F  7551.0  32436.0   816.6 11178.2 205.8 29498.3  7914.2     92.9
1475542113_E  6137.2  28081.0   739.4  9158.1 176.6 23302.4  6680.7     86.1
1475542114_A 10255.0  41451.7  1040.9 13176.7 320.3 30390.5 15902.3    106.0
                 mm      pm negative
1475542110_F 3584.5 21807.1     94.4
1475542113_E 1516.5 18619.5     88.6
1475542114_A 5738.7 27314.2    108.7

> plotQC(QC)
```



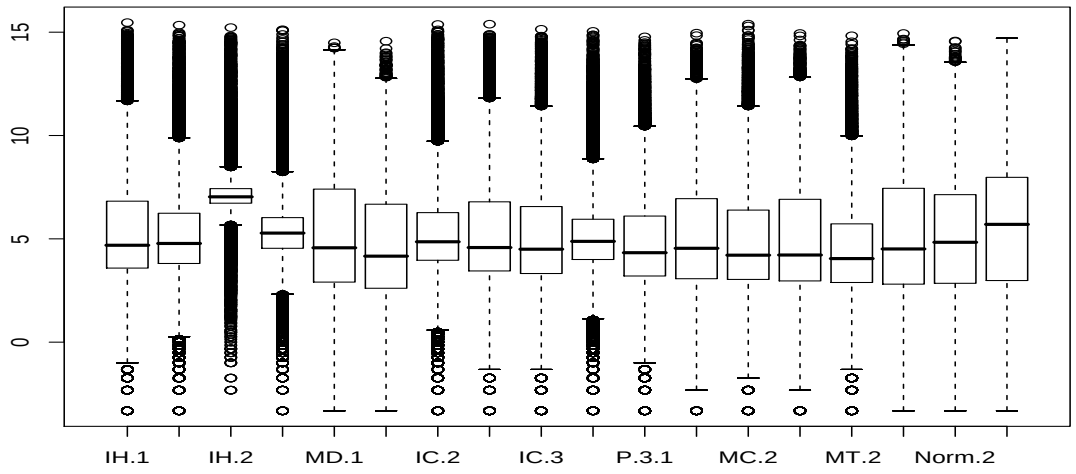
The `singleQCPlot` function allows a particular control type to be plotted across all samples. The type argument must match one of the column names of QC$Signal and the `what` argument selects which of the Signal, Var and Detection matrices to plot. Additional plotting arguments such as a title for the plot, plotting character etc can also be passed to the function. We can also choose to plot on the $\log_2$ scale.

```
> par(mfrow = c(1, 3))
> singleQCPlot(QC, type = "negative", main = "Negative Control Signal",
+     what = "Signal")
> singleQCPlot(QC, type = "Biotin", log = TRUE, pch = 16, col = "red",
+     lwd = 2, lty = 2, what = "Var", main = "log2 Biotin Signal")
> singleQCPlot(QC, type = "Biotin", pch = 16, col = "red", lwd = 2,
+     lty = 2, what = "Var", main = "Biotin Variability")
```



Illumina also use this quality control information to normalise bead summary data. In a procedure known as background normalisation, the averaged values of all negative controls on a particular array are subtracted from the summarised expression of each gene. This normalisation can be repeated by the function `backgroundNormalise`. The effect of this normalisation is to remove the effects of non-specific binding from the expression values. This effect is more noticable for genes with low expression level and hence can produce negative values. Note that when updating the expression values with the normalised values we use the `assayDataElementReplace` function.

```
> BSData.bgnorm = assayDataElementReplace(BSData, "exprs", backgroundNormalise(exprs(BSData),
+     QC))
> range(exprs(BSData)[, 1])

[1]    65.9 45311.7

> range(exprs(BSData.bgnorm)[, 1])

[1]  -142.7 45221.5

> boxplot(log2(exprs(BSData.bgnorm)))
```



It is possible to use the normalisation methods available in the affy such as quantile, qspline or others. The method of rank invariant normalisation recommended by Illumina may also be applied once a suitable target distribution has been defined. In the following example we define this to be the mean of each row before using the `normalize.invariantset` to find a set of invariant genes and define a normalising curve using this set and the target distribution.

```
> library(affy)
> BSData.quantile = assayDataElementReplace(BSData, "exprs", normalize.quantiles(as.matrix(exprs(BSData))))
> BSData.qspline = assayDataElementReplace(BSData, "exprs", normalize.qspline(as.matrix(exprs(BSData))))
> T = apply(exprs(BSData.bgnorm), 1, mean)
> BSData.rankinv = assayDataElementReplace(BSData.bgnorm, "exprs",
+     rankInvariantNormalise(exprs(BSData.bgnorm), T))
```

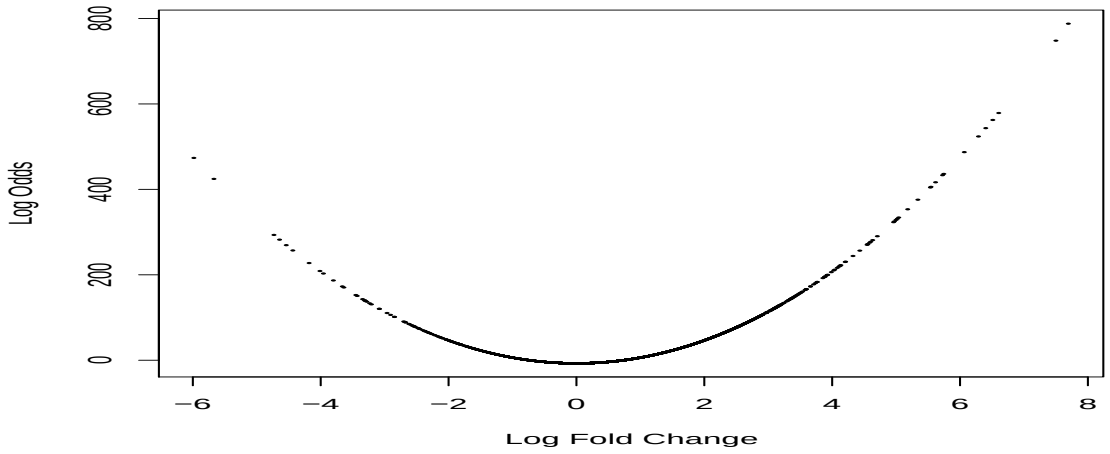
5 Differential Expression

Research into the best method for detecting differential expression for BeadArray data is still work in progress. In the meantime, users are able to use the `lmFit` and `eBayes` functions from limma on the matrix `exprs(BSdata)` with a $\log_2$ transformation applied.

The following code shows how to set up a design matrix for the example experiment combining the I, MC, MD, MT, P and Normal samples together. We then define contrasts comparing the I samples to the P samples and I to Normal and perform an empirical bayes shrinkage. In this particular experiment, the I and P samples are completely different so we would expect to see plenty of differentially expressed genes.

For more information about `lmFit` and `eBayes` please see the comprehensive limma documentation.

```
> design = matrix(nrow = 18, ncol = 6, 0)
> colnames(design) = c("I", "MC", "MD", "MT", "P", "Norm")
> design[which(strtrim(colnames(exprs(BSData)), 1) == "I"), 1] = 1
> design[which(strtrim(colnames(exprs(BSData)), 2) == "MC"), 2] = 1
> design[which(strtrim(colnames(exprs(BSData)), 2) == "MD"), 3] = 1
> design[which(strtrim(colnames(exprs(BSData)), 2) == "MT"), 4] = 1
> design[which(strtrim(colnames(exprs(BSData)), 1) == "P"), 5] = 1
> design[which(strtrim(colnames(exprs(BSData)), 1) == "N"), 6] = 1
> design
> fit = lmFit(log2(exprs(BSData)), design)
> cont.matrix = makeContrasts(IvsP = I - P, IvsNorm = I - Norm,
+     PvsNorm = P - Norm, levels = design)
> fit = contrasts.fit(fit, cont.matrix)
> ebFit = eBayes(fit)
> topTable(ebFit)
> volcanoplot(ebFit)
```



The algorithm for the Illumina method is implemented in the function `IlluminaDE` although it not completely accurate at present. To compare array 1 in the experiment to array 10 (ie comparing an I sample to a P) we would use.

```
> library(MASS)
> df = IlluminaDE(exprs(BSData), QC$Signal, se.exprs(BSData), QC$Var,
+     cond = 10, ref = 1)
```

6 Further Analysis

The clustering functionality available in BeadStudio can be easily performed through `R` using the `hlcust` once a distance matrix has been defined. In this example we see that the clusters correspond well to the different sample types. The `heatmap` function could also be used in a similar manner and principal components analysis is possible using `princomp`.

```
> d = dist(t(exprs(BSData)))
> plclust(hclust(d), labels = rownames(pData(BSData)))
```