# Analysis of Bead Summary Data using beadarray

May 18, 2007

## 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 and in a separate vignette of beadarray.

*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. The methods described within this document are for the analysis of bead summary data which can be 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 S, *beadarray: An R package to Analyse Illumina BeadArrays*, R News, submitted

## 2 Getting help with beadarray

Wherever possible, please send all queries about beadarray to the Bioconductor mailing list at `bioconductor@stat.math.ethz.ch`. This will help to maintain a searchable archive of questions and responses visible to all users of the package.

## 3 Importing Bead Summary Data

The beadarray package is able to read the output of BeadStudio versions 1, 2 and 3 which comes in the form of a text file. We assume the file to have one row for each probe and a set of columns for each array, depending on which columns have been exported from BeadStudio. We prefer that the annotation columns are not exported from BeadStudio. These columns often contain unusual characters which cannot be easily read into R. If required, annotation information can be imported at a later time through other Bioconductor packages.

An example data set is included with the beadarray package and can be found as a zip file (BeadSummaryExample.zip) inside the inst/demodata directory of the beadarray download. Inside this zip you will find the raw non-normalised data, a sample sheet and a quality control file for an example 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. The normalised data and quality control information was produced using BeadStudio version 1.

## 3.1 Description of Files

We now describe the included files in more detail.

- 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.

  When exporting this file from BeadStudio, the user is able to choose which columns to export. However, beadarray is able to read any combination of these columns.

- raw_data_sample_sheet - Defines the array IDs and samples placed on each array. In order for this information to be read into `beadarray`, we require that the 4th column is a unique identifier for each array in the experiemnt. This is a file format that Illumina recommend for users of BeadStudio to specify the contents of 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 format of the quality control files differs slightly between BeadStudio versions 1 and 2. Version 1 of the software gives one averaged value for each control type, whereas version 2 gives sumarised values for each control of a particular type. The user does not have to know the version of BeadStudio used to generate the file.

The following code can be used to read the example data into R. First make sure that the contents of BeadSummaryExample.zip are extracted to the current working directory. If the quality control file and sample sheet are not available, then the raw data can be read in on it's own.

The function `readBeadSummaryData` can be made to read the ouput of either versions 1 and 2 of BeadStudio. Users may need to change the argument `sep`, which specifies if the file is comma or tab delimited and `skip` which specifies the number of lines of header information at the top of the file. Equivalent arguments are used to read the quality control file (`qc.skip` and `qc.sep`). The name of the columns containing standard errors may also change between BEAD_STDEV and BEAD_STDERR.

The `columns` argument is used to decide which column headings to read from the file and where to store the data in the object created by the function (see later).

```
> dataFile = "raw_data.csv"
> sampleSheet = "raw_data_sample_sheet.csv"
> qcFile = "raw_data_qcinfo.csv"
> BSData <- readBeadSummaryData(dataFile, qcFile = qcFile, sampleSheet = sampleSheet,
+     skip = 7, columns = list(exprs = "AVG_Signal", se.exprs = "BEAD_STDEV",
+         NoBeads = "Avg_NBEADS"), qc.columns = list(exprs = "AVG.Signal",
+         se.exprs = "SeqVAR"), qc.sep = ",", sep = ",", qc.skip = 7,
+     annoPkg = "illuminaHumanv1")
```

## 4 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. Objects of this type use a series of slots to store data.

```
> BSData
```

```
ExpressionSetIllumina (storageMode: list)
assayData: 47293 features, 18 samples
  element names: exprs, se.exprs, NoBeads, Detection, Narrays, arrayStDev, DiffScore
phenoData
  rowNames: 1, 2, ..., 18 (18 total)
  varLabels and varMetadata:
    Sample_Name: Sample_Name
    Sample_Well: Sample_Well
    ...: ...
    Sentrix_Position: Sentrix_Position
    (7 total)
featureData
  rowNames:
  varLabels and varMetadata: none
experimentData: use 'experimentData(object)'
Annotation [1] "illuminaHumanv1"
QC Information
 Available Slots:  exprs se.exprs Detection NoBeads controlType
  featureNames: AVG.Signal.biotin, AVG.Signal.cy3_hyb_high, ..., AVG.Signal.low_stringency_hyb_pm, AVG
  sampleNames: 1475542110_F, 1475542113_E, ..., 1475542113_D, 1475542113_F
```

```
> slotNames(BSData)
```

```
[1] "QC"                "assayData"         "phenoData"
[4] "featureData"       "experimentData"    "annotation"
[7] ".__classVersion__"
```

```
> names(assayData(BSData))
```

```
[1] "exprs"      "se.exprs"   "NoBeads"    "Detection"  "Narrays"
[6] "arrayStDev" "DiffScore"
```

```
> dim(assayData(BSData)$exprs)
```

```
[1] 47293    18
```

```
> dim(assayData(BSData)$BeadStDev)
```

```
NULL
```

```
> dim(assayData(BSData)$Narrays)
```

```
[1] 0 0
```

```
> 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]
```

```
               IH-1 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)[, c(4, 6)]
```

```
   Sample_Group Sentrix_ID
1          IH-1 1475542114
2          IC-1 1475542114
3          IH-2 1475542114
4          MC-1 1475542114
5          MD-1 1475542114
6          MT-1 1475542114
7          IC-2 1475542110
8          IH-3 1475542110
9          IC-3 1475542110
10          P-3 1475542110
11          P-3 1475542110
12       Norm-1 1475542110
13         MC-2 1475542113
14         MD-2 1475542113
15         MT-2 1475542113
16          P-1 1475542113
17       Norm-2 1475542113
18          P-2 1475542113
```

```
> QCInfo(BSData)$exprs[1:5, 1:4]
```

```
                        1475542110_F 1475542113_E 1475542114_A 1475542114_C
AVG.Signal.biotin             7551.0       6137.2      10255.0       9358.7
AVG.Signal.cy3_hyb_high      32436.0      28081.0      41451.7      41116.9
AVG.Signal.cy3_hyb_low          816.6         739.4        1040.9        1100.0
AVG.Signal.cy3_hyb_med        11178.2        9158.1       13176.7       13109.2
AVG.Signal.gene                 205.8         176.6         320.3         395.2
```

The data from the the raw_data file has been written to the assayData slot of the object. This slot contains a number of matrices, each of which has a column for each array in the experiment and a row for each probe. There is a matrix for each column that can be exported from BeadStudio, although only the columns that we choose to read using the `columns` parameter in `readBeadSummaryData` will be filled.

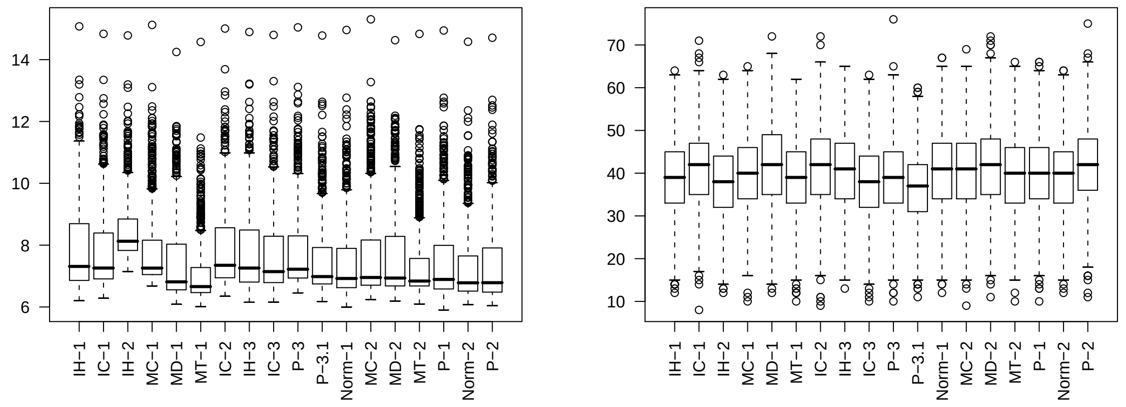
For consistency with the definition of other *ExpressionSet* objects, we now refer to the expression values as the *exprs* matrix which can be accessed using `exprs` and subset in the usual manner. Similarly, the `se.exprs` matrix can be accessed using `se.exprs`. The rows of `exprs` are named according to the row names of the original raw_data file.

Phenotypic data for the experiment can be accessed using `pData` and the QC slot contains the quality control information.

Boxplots of expression may be useful for quality control. Below we show the code to produce boxplots of the log2 intensities of each array in the experiment. Recall that there are 6 arrays per BeadChip and that differences between chip hybridised on different days may be seen. In this example the differences in intensity between arrays on the same chip and different chips do not seem too large. However, 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.[1]

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



## 5 Normalisation and Quality Control

In the 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 a higher intensity than the others. The sample on this array is replicated three times on the first chip, so comparing the MA and XY plots for the replicates of this sample can be informative.

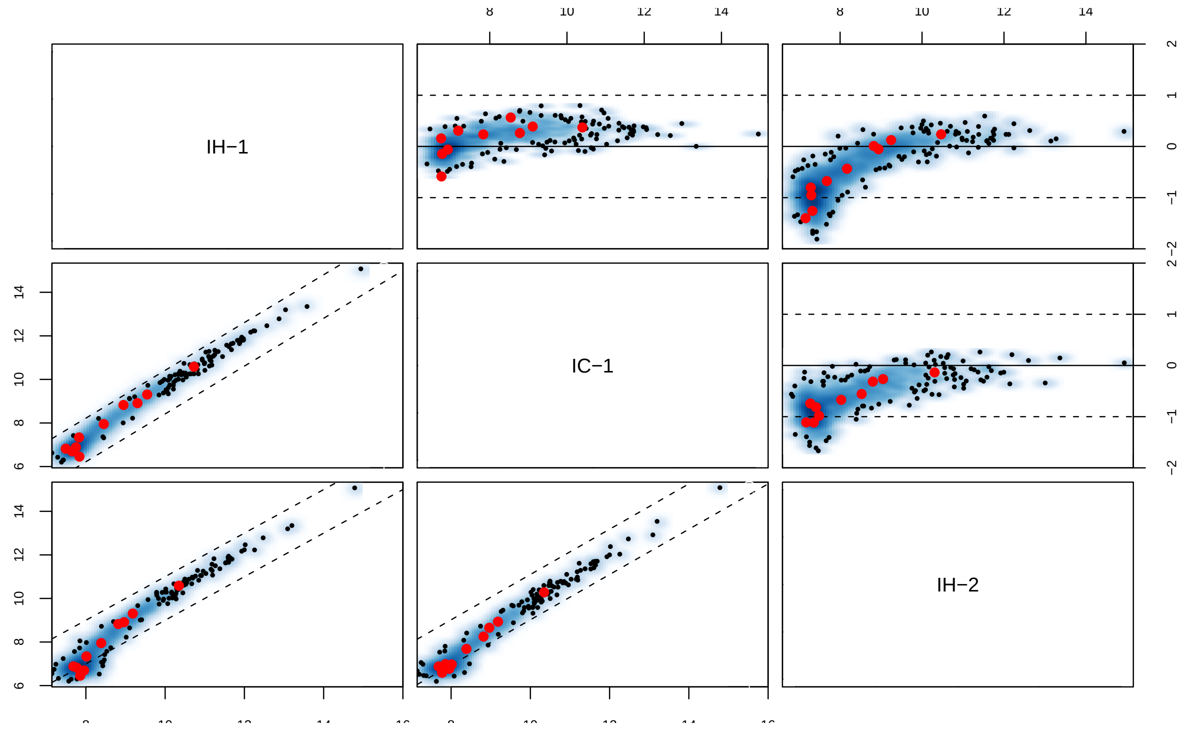
[1]We have restricted the number of points plotted in order to keep the size of this vignette small.

Particular genes of interest may be highlighted on the MA and XY plots by using the `genesToLabel` argument which should match up with the row names in `BSData`. The `labelCol` argument can be used to specify a colour for each gene. For simplicity sake we simply highlight the first ten genes in the expression matrix, a possible application might be to highlight control genes on the plot or particular genes of interest.

```
> par(mai = c(0.5, 0.5, 0.5, 0.5))
> g = rownames(exprs(BSData))[1:10]
> g
```

```
 [1] "GI_10047089-S" "GI_10047091-S" "GI_10047093-S" "GI_10047099-S"
 [5] "GI_10047103-S" "GI_10047105-S" "GI_10047121-S" "GI_10047123-S"
 [9] "GI_10047133-A" "GI_10047133-I"
```

```
> cols = rainbow(start = 0, end = 5/6, n = 10)
> plotMAXY(exprs(BSData)[1:1000, ], arrays = 1:3, genesToLabel = g,
+     labelCols = cols, labels = as.character(pData(BSData)[1:3,
+         4]), pch = 16)
```



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 different to the other replicates and normalisation is required.

Both XY and MA plots for a particular comparison of arrays are available separately using `plotXY` and `plotMA`

## 6 Using Quality Control Information

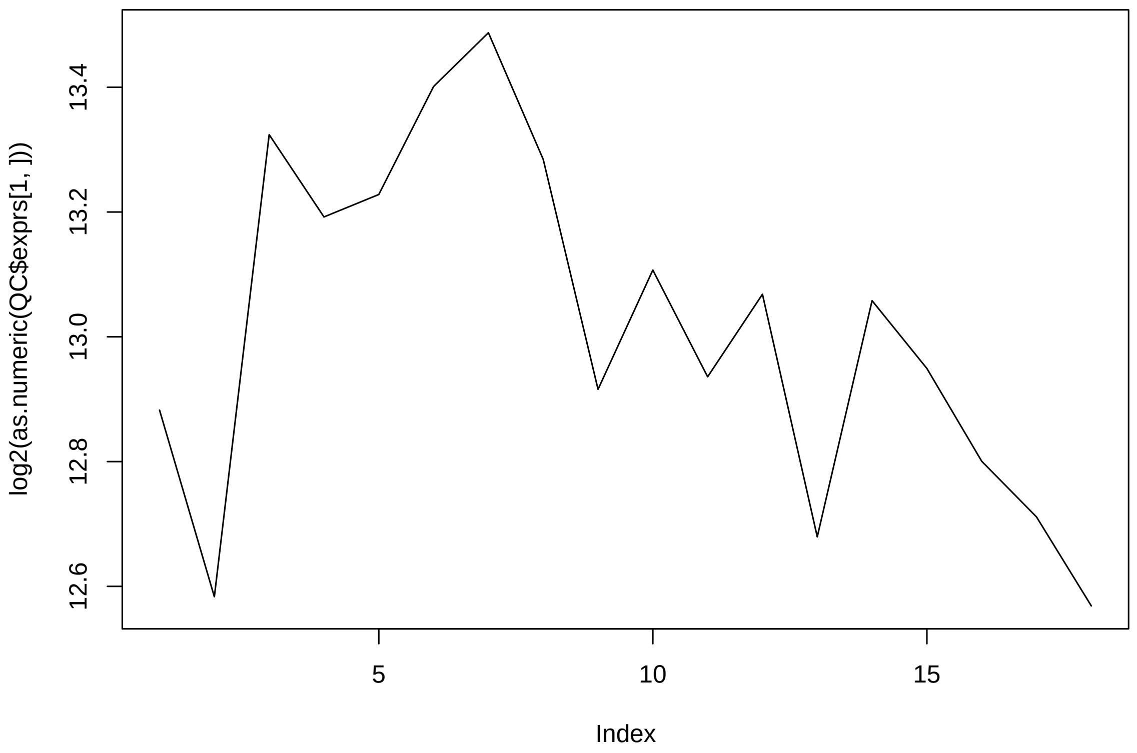
Quality control information from Illumina experiments can be exported from the BeadStudio application. This information can be read into beadarray using the `readBeadSummaryData` function or at a later time using `readQC`.

The quality control information which is read in by `readBeadSummaryData` can be plotted to provide useful diagnostic information. To retrieve this quality control data we can use the `QCInfo` function. Alternatively, quality control information can be read using `readQC`. QC is a list object and each item can be accessed using the $ operator to give a matrix. The row names of the matrix indicates the control types. In the following example we plot the values of the Biotin control across all arrays.

```
> QC = QCInfo(BSData)
> QC$exprs[1:3, ]
```

```
                       1475542110_F 1475542113_E 1475542114_A 1475542114_C
AVG.Signal.biotin            7551.0       6137.2      10255.0       9358.7
AVG.Signal.cy3_hyb_high     32436.0      28081.0      41451.7      41116.9
AVG.Signal.cy3_hyb_low        816.6        739.4       1040.9       1100.0
                       1475542110_B 1475542114_B 1475542110_A 1475542110_C
AVG.Signal.biotin            9595.1      10818.4      11483.9       9977.0
AVG.Signal.cy3_hyb_high     41957.6      44276.0      47204.4      45043.8
AVG.Signal.cy3_hyb_low       1017.8       1039.5       1037.6        953.8
                       1475542114_D 1475542113_A 1475542114_E 1475542113_B
AVG.Signal.biotin            7727.4       8822.6       7835.9       8588.6
AVG.Signal.cy3_hyb_high     34646.3      36889.3      33330.6      40451.8
AVG.Signal.cy3_hyb_low        868.5        944.4        820.0        967.3
                       1475542114_F 1475542113_C 1475542110_D 1475542110_E
AVG.Signal.biotin            6559.4       8527.5       7908.9       7134.3
AVG.Signal.cy3_hyb_high     27580.2      39325.6      33515.9      31132.7
AVG.Signal.cy3_hyb_low        697.4        949.6        901.8        797.9
                       1475542113_D 1475542113_F
AVG.Signal.biotin            6706.2       6075.0
AVG.Signal.cy3_hyb_high     30452.9      25584.2
AVG.Signal.cy3_hyb_low        828.7        730.8
```

```
> plot(log2(as.numeric(QC$exprs[1, ])), type = "l")
```



It is possible to use the normalisation methods available in the affy such as quantile, qspline or others.

```
> library(affy)
> BSData.quantile = assayDataElementReplace(BSData, "exprs", normalize.quantiles(as.matrix(exprs(BSData))))
> BSData.qspline = assayDataElementReplace(BSData, "exprs", normalize.qspline(as.matrix(exprs(BSData))))
> row.names(BSData.quantile@assayData$exprs) = row.names(exprs(BSData))
```

## 7 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
```

```
      I MC MD MT P Norm
 [1,] 1  0  0  0 0    0
 [2,] 1  0  0  0 0    0
 [3,] 1  0  0  0 0    0
 [4,] 0  1  0  0 0    0
 [5,] 0  0  1  0 0    0
 [6,] 0  0  0  1 0    0
 [7,] 1  0  0  0 0    0
 [8,] 1  0  0  0 0    0
 [9,] 1  0  0  0 0    0
[10,] 0  0  0  0 1    0
[11,] 0  0  0  0 1    0
[12,] 0  0  0  0 0    1
[13,] 0  1  0  0 0    0
[14,] 0  0  1  0 0    0
[15,] 0  0  0  1 0    0
[16,] 0  0  0  0 1    0
[17,] 0  0  0  0 0    1
[18,] 0  0  0  0 1    0
```

```
> fit = lmFit(log2(exprs(BSData.quantile)), 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)
```

```
                 ID        IvsP      IvsNorm     PvsNorm        F      P.Value
9260  GI_28302132-S  7.08670571  6.808981535 -0.27772417 6279.771 1.009580e-37
13743 GI_34304351-S -6.44976596 -3.077451340  3.37231462 5521.628 6.039122e-37
21840  GI_4501988-S  5.54128477  5.166640940 -0.37464383 4650.397 6.567391e-36
22143  GI_4503886-S  5.15617933  5.144907385 -0.01127194 4339.786 1.715804e-35
9259  GI_28302130-S  6.86187520  6.607825095 -0.25405011 4298.623 1.958668e-35
2128  GI_15149480-S -6.24812250 -6.323759039 -0.07563654 3837.138 9.482842e-35
19978 GI_42542384-S  5.23927534  5.006838290 -0.23243705 3732.388 1.392636e-34
22144  GI_4503888-S  4.61468838  4.554406233 -0.06028215 3311.443 7.331637e-34
3117  GI_18375521-A -4.92498629 -0.001106266  4.92388002 3296.103 7.819572e-34
9257  GI_28302128-S  0.07210736 -6.351676194 -6.42378355 3032.558 2.484962e-33
         adj.P.Val
9260  4.774606e-33
13743 1.428041e-32
21840 1.035305e-31
22143 1.852626e-31
9259  1.852626e-31
2128  7.474534e-31
19978 9.408848e-31
```

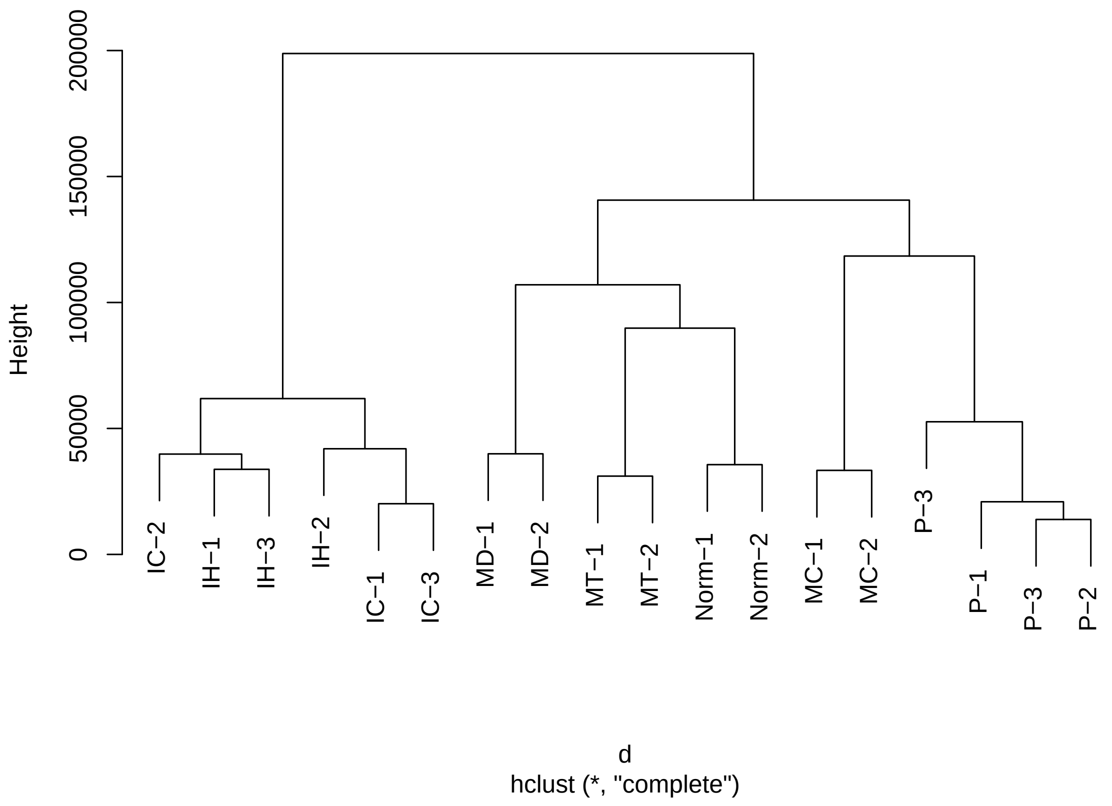
```
22144 4.109011e-30
3117  4.109011e-30
9257  1.071209e-29
```

```
> results <- topTable(ebFit, number = 40, sort.by = "B", resort.by = "M")
```

## 8 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 = pData(BSData)[, 4])
```



```
> library(biomaRt)
> ensembl <- useMart("ensembl", dataset = "hsapiens_gene_ensembl")

Checking attributes and filters ... ok

> illuids = results$ID
> BM <- getBM(attributes = c("illumina_v1", "entrezgene", "go",
+     "go_description"), filters = "illumina_v1", values = illuids,
+     mart = ensembl)
> BM[1:20, ]
```

```
     illumina_v1 entrezgene         go                        go_description
1  GI_13027795-S       4320 GO:0004249               stromelysin 3 activity
2  GI_13027795-S       4320 GO:0005509                  calcium ion binding
3  GI_13027795-S       4320 GO:0005578   proteinaceous extracellular matrix
4  GI_13027795-S       4320 GO:0006508                          proteolysis
5  GI_13027795-S       4320 GO:0007275 multicellular organismal development
6  GI_13027795-S       4320 GO:0008270                     zinc ion binding
7  GI_13027795-S       4320 GO:0009653   anatomical structure morphogenesis
8  GI_13027795-S       4320 GO:0030574            collagen catabolic process
9  GI_14456711-S       3039 GO:0005344          oxygen transporter activity
10 GI_14456711-S       3039 GO:0005506                     iron ion binding
11 GI_14456711-S       3039 GO:0005515                      protein binding
12 GI_14456711-S       3039 GO:0005833                    hemoglobin complex
13 GI_14456711-S       3039 GO:0006810                            transport
14 GI_14456711-S       3039 GO:0015671                     oxygen transport
15 GI_14456711-S       3039 GO:0019825                       oxygen binding
16 GI_14456711-S       3039 GO:0020037                         heme binding
17 GI_14456711-S       3039 GO:0046872                    metal ion binding
18 GI_14456711-S       3040 GO:0005344          oxygen transporter activity
19 GI_14456711-S       3040 GO:0005506                     iron ion binding
20 GI_14456711-S       3040 GO:0005515                      protein binding
```