Analysis of Bead Summary Data using beadarray

September 27, 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

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 expression 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 MJ, Smith ML, Ritchie ME, Tavaré S, **beadarray: R classes and methods for Illumina bead-based data**, *Bioinformatics*, **23**(16):2183-2184

2 Getting help with beadarray

Wherever possible, questions about `beadarray` should be sent to the Bioconductor mailing list (bioconductor@stat.math.ethz.ch). Therefore all problems and solutions will be kept in a searchable archive. When posting to this mailing list, please state the version of `beadarray` and `R` to help to diagnose the problem. This can be done by pasting the output of running the command `sessionInfo()`.

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` (*required*) - This contains the raw, non-normalised bead summary values as output by BeadStudio. 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 (BEAD_STDEV), Number of beads used (Avg_NBEADS) and a Detection score which estimates the probability of a gene being detected above the background. When exporting this file from BeadStudio, the user is able to choose which columns to export. beadarray is able to read any combination of these columns.

  With versions 2 and 3 of BeadStudio it is possible to export annotation information. We recommend this data not be exported if the file is to be read into beadarray, as some of the special characters used in the annotation fields cause problems for the `readBeadSummaryData` function. This information can be retrieved later on from other Bioconductor packages, such as illuminaHumanv1.

- `raw_data_sample_sheet.csv` (*optional*) - 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 experiment. This is a file format that Illumina recommend for users of BeadStudio to specify the contents of each array. This file is created by the user.

- `raw_data_qcinfo.csv` (*optional*) - Gives the summarised data for each of the controls on each array, which may be useful for diagnostic purposes. This file was produced using BeadStudio version 1. 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. Refer to the Illumina documentation for information on what each control measures.

- `FullControlInfo.txt` (*optional*) - Quality control information exported by BeadStudio version 2.

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, 2 or 3 of BeadStudio. The default settings will read BeadStudio version 3 output. 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).

The following code shows how to read version 1 output by adjusting `sep` and `skip`. Version 3 output can be read by running `readBeadSummaryData` without having to specify any additional parameters.

```
> library(beadarray)
> 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)
```

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 description:
    Sample_Name: Sample_Name
    Sample_Well: Sample_Well
    ...: ...
    Sentrix_Position: Sentrix_
  Position
    (7 total)
featureData
  featureNames:
  fvarLabels and fvarMetadata description: none
experimentData: use 'experimentData(object)'
Annotation:
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"
[2] "assayData"
[3] "phenoData"
[4] "featureData"
[5] "experimentData"
[6] "annotation"
[7] ".__classVersion__"
```

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

```
[1] "exprs"      "se.exprs"
[3] "NoBeads"    "Detection"
```

```
[5] "Narrays"    "arrayStDev"
[7] "DiffScore"
```

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

```
[1] 47293    18
```

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

```
[1] 47293    18
```

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

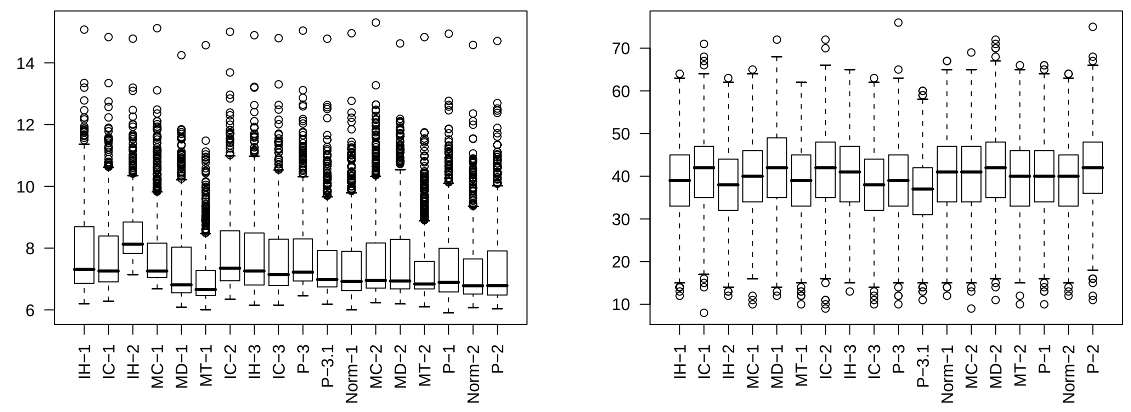
For consistency with the definition of other *ExpressionSet* objects, we now refer to the expression values as the *exprs* matrix which can be accessed using `exprs` and subset in the usual manner. Similarly, the `se.exprs` matrix can be accessed using `se.exprs`. The rows of `exprs` are named according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

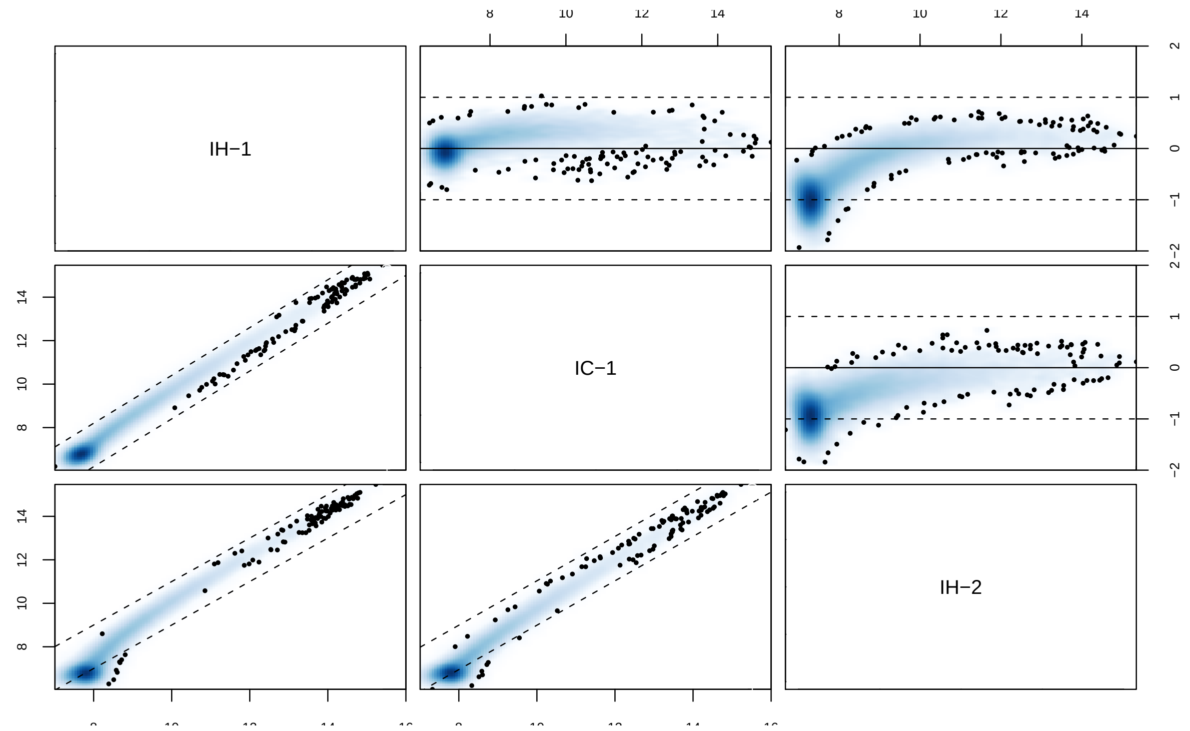


5 Normalisation and Quality Control

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.

```
> plotMAXY(exprs(BSData),
+     arrays = 1:3, labels = as.character(pData(BSData)[1:3,
+         4]), pch = 16)
```

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



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.

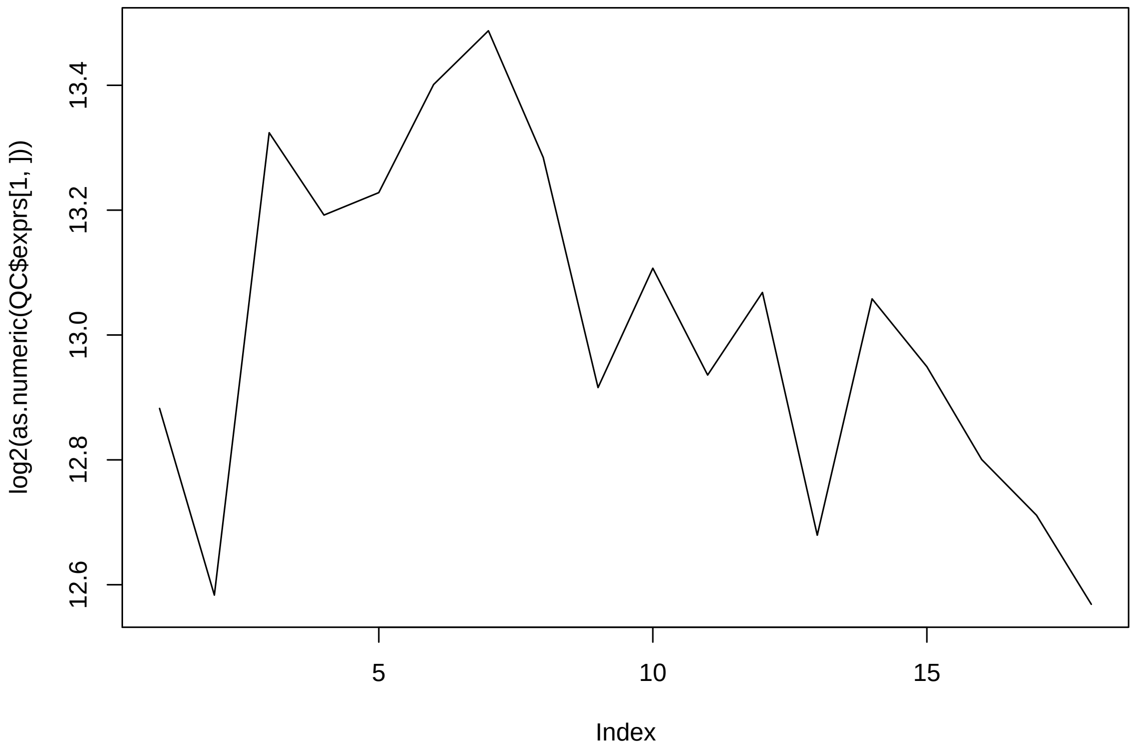
```
> QC = QCInfo(BSData)
> QC$exprs[1:3, ]
```

```
                        1475542110_F
AVG.Signal.biotin             7551.0
AVG.Signal.cy3_hyb_high      32436.0
AVG.Signal.cy3_hyb_low         816.6
                        1475542113_E
```

```
AVG.Signal.biotin            6137.2
AVG.Signal.cy3_hyb_high     28081.0
AVG.Signal.cy3_hyb_low        739.4
                       1475542114_A
AVG.Signal.biotin           10255.0
AVG.Signal.cy3_hyb_high     41451.7
AVG.Signal.cy3_hyb_low       1040.9
                       1475542114_C
AVG.Signal.biotin            9358.7
AVG.Signal.cy3_hyb_high     41116.9
AVG.Signal.cy3_hyb_low       1100.0
                       1475542110_B
AVG.Signal.biotin            9595.1
AVG.Signal.cy3_hyb_high     41957.6
AVG.Signal.cy3_hyb_low       1017.8
                       1475542114_B
AVG.Signal.biotin           10818.4
AVG.Signal.cy3_hyb_high     44276.0
AVG.Signal.cy3_hyb_low       1039.5
                       1475542110_A
AVG.Signal.biotin           11483.9
AVG.Signal.cy3_hyb_high     47204.4
AVG.Signal.cy3_hyb_low       1037.6
                       1475542110_C
AVG.Signal.biotin            9977.0
AVG.Signal.cy3_hyb_high     45043.8
AVG.Signal.cy3_hyb_low        953.8
                       1475542114_D
AVG.Signal.biotin            7727.4
AVG.Signal.cy3_hyb_high     34646.3
AVG.Signal.cy3_hyb_low        868.5
                       1475542113_A
AVG.Signal.biotin            8822.6
AVG.Signal.cy3_hyb_high     36889.3
AVG.Signal.cy3_hyb_low        944.4
                       1475542114_E
AVG.Signal.biotin            7835.9
AVG.Signal.cy3_hyb_high     33330.6
AVG.Signal.cy3_hyb_low        820.0
                       1475542113_B
AVG.Signal.biotin            8588.6
AVG.Signal.cy3_hyb_high     40451.8
AVG.Signal.cy3_hyb_low        967.3
                       1475542114_F
AVG.Signal.biotin            6559.4
AVG.Signal.cy3_hyb_high     27580.2
AVG.Signal.cy3_hyb_low        697.4
                       1475542113_C
AVG.Signal.biotin            8527.5
AVG.Signal.cy3_hyb_high     39325.6
AVG.Signal.cy3_hyb_low        949.6
```

```
                        1475542110_D
AVG.Signal.biotin             7908.9
AVG.Signal.cy3_hyb_high      33515.9
AVG.Signal.cy3_hyb_low         901.8
                        1475542110_E
AVG.Signal.biotin             7134.3
AVG.Signal.cy3_hyb_high      31132.7
AVG.Signal.cy3_hyb_low         797.9
                        1475542113_D
AVG.Signal.biotin             6706.2
AVG.Signal.cy3_hyb_high      30452.9
AVG.Signal.cy3_hyb_low         828.7
                        1475542113_F
AVG.Signal.biotin             6075.0
AVG.Signal.cy3_hyb_high      25584.2
AVG.Signal.cy3_hyb_low         730.8
```

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



It is possible to use the normalisation methods available in the affy or lumi packages.

```
> library(affy)
> library(lumi)
> BSData.quantile = normaliseIllumina(BSData,
+     method = "quantile",
+     transform = "log2")
> BSData.lumi = normaliseIllumina(BSData,
+     method = "quantile",
+     transform = "vst")
```

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
```

```
> ids = rownames(exprs(BSData))
> controlids = c("lysA",
+     "pheA", "thrB", "trpF")
> controlind = match(controlids,
+     ids)
> fit = lmFit(exprs(BSData.quantile)[-controlind,
+     ], 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, genelist = ids[-controlind])
```

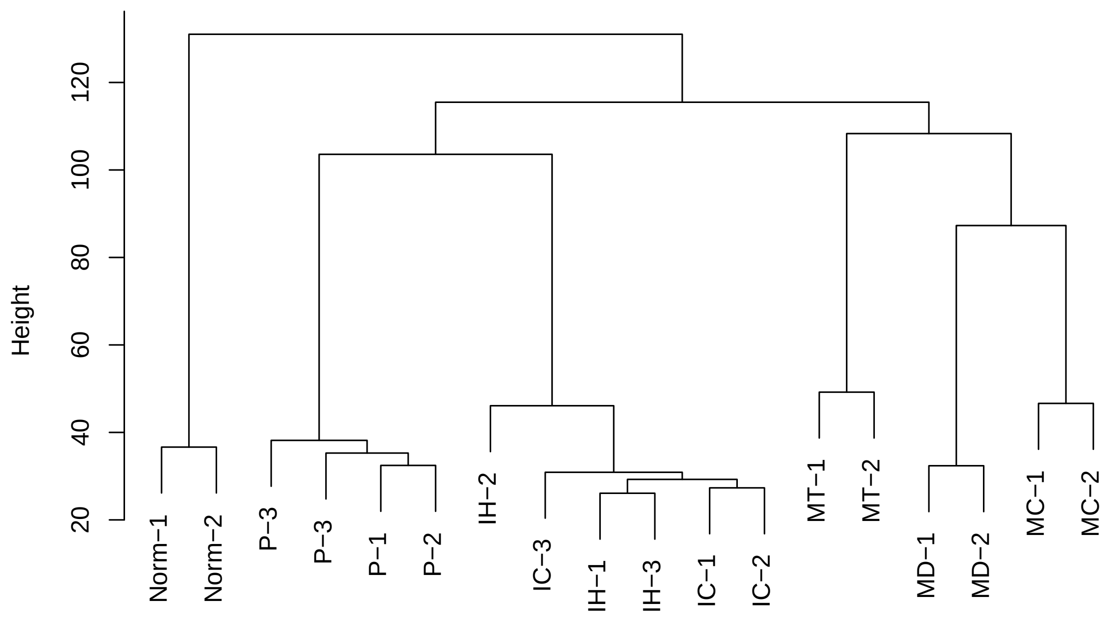
```
            ProbeID
9260  GI_28302132-S
13743 GI_34304351-S
21840  GI_4501988-S
22143  GI_4503886-S
9259  GI_28302130-S
2128  GI_15149480-S
19978 GI_42542384-S
22144  GI_4503888-S
3117  GI_18375521-A
9257  GI_28302128-S
             IvsP
9260   7.05124284
13743 -6.41967544
21840  5.50531799
22143  5.12600762
9259   6.82821239
2128  -6.21848978
19978  5.20822549
22144  4.59044439
3117  -4.89612179
9257   0.07101028
           IvsNorm
9260   6.775686661
13743 -3.060012038
21840  5.134115849
22143  5.115005439
9259   6.576394085
2128  -6.293353109
19978  4.977646454
22144  4.530531990
3117  -0.001238405
```

```
9257  -6.325215556
         PvsNorm        F
9260  -0.27555618 6324.087
13743  3.35966341 5562.350
21840 -0.37120215 4661.143
22143 -0.01100219 4360.251
9259  -0.25181830 4338.271
2128  -0.07486333 3874.901
19978 -0.23057903 3754.780
22144 -0.05991240 3323.952
3117   4.89488338 3310.720
9257  -6.39622584 3062.805
          P.Value
9260  5.413714e-38
13743 3.266853e-37
21840 3.879873e-36
22143 9.871758e-36
9259  1.059516e-35
2128  5.145356e-35
19978 7.992802e-35
22144 4.393341e-34
3117  4.645256e-34
9257  1.378672e-33
         adj.P.Val
9260  2.560091e-33
13743 7.724312e-33
21840 6.115845e-32
22143 1.002069e-31
9259  1.002069e-31
2128  4.055312e-31
19978 5.399594e-31
22144 2.440772e-30
3117  2.440772e-30
9257  6.519604e-30
```

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.quantile)))
> plclust(hclust(d), labels = pData(BSData)[,
+     4])
```



d
hclust (*, "complete")

So far, the results returned by the limma analysis are in terms of probes with Illumina IDs. This alone is not very useful for drawing biologically meaningful conclusions from the experiment. More useful information about each can be obtained using the illuminaHumanv1 package. In the following code, we show how to obtain chromsomal information and gene names for every probe. The rows of the expression matrix are used as a key to extract the relevant information from the environment. Note that some probes on Illumina arrays will not have a matching gene name or RefSeq ID.

```
> library(illuminaHumanv1)
> illuminaHumanv1()

Quality control information for  illuminaHumanv1
Date built: Created: Wed Apr  4 10:51:36 2007

Number of probes: 47297
Probe number missmatch: illuminaHumanv1ACCNUM; illuminaHumanv1CHRLOC; illuminaHumanv1CHR; illuminaHuma
Probe missmatch: None
Mappings found for probe based rda files:
         illuminaHumanv1ACCNUM found 26092 of 47297
         illuminaHumanv1CHRLOC found 18648 of 47297
         illuminaHumanv1CHR found 19255 of 47297
         illuminaHumanv1ENTREZID found 19258 of 47297
         illuminaHumanv1ENZYME found 2147 of 47297
         illuminaHumanv1GENENAME found 18182 of 47297
         illuminaHumanv1GO found 16772 of 47297
         illuminaHumanv1MAP found 19160 of 47297
         illuminaHumanv1OMIM found 12692 of 47297
         illuminaHumanv1PATH found 4630 of 47297
         illuminaHumanv1PMID found 19001 of 47297
         illuminaHumanv1REFSEQ found 19258 of 47297
         illuminaHumanv1SUMFUNC found 0 of 47297
         illuminaHumanv1SYMBOL found 19258 of 47297
         illuminaHumanv1UNIGENE found 19023 of 47297
Mappings found for non-probe based rda files:
          illuminaHumanv1CHRLENGTHS found 25
         illuminaHumanv1ENZYME2PROBE found 785
         illuminaHumanv1GO2ALLPROBES found 8265
         illuminaHumanv1GO2PROBE found 6093
         illuminaHumanv1ORGANISM found 1
         illuminaHumanv1PATH2PROBE found 192
         illuminaHumanv1PFAM found 15279
         illuminaHumanv1PMID2PROBE found 139948
         illuminaHumanv1PROSITE found 10793
```

```
> ids = rownames(exprs(BSData))
> chr = mget(ids, illuminaHumanv1CHR,
+     ifnotfound = NA)
> chrloc = mget(ids, illuminaHumanv1CHRLOC,
+     ifnotfound = NA)
> refseq = mget(ids, illuminaHumanv1REFSEQ,
+     ifnotfound = NA)
> genename = mget(ids, illuminaHumanv1GENENAME,
+     ifnotfound = NA)
> anno = cbind(Ill_ID = as.character(ids),
+     Chr = as.character(chr),
+     Loc = as.character(chrloc),
+     RefSeq = as.character(refseq),
+     Name = as.character(genename))
> ebFit$genes = anno
> topTable(ebFit)
```

```
             Ill_ID Chr
9260  GI_28302132-S  11
13743 GI_34304351-S   X
21840  GI_4501988-S   4
22143  GI_4503886-S   X
9259  GI_28302130-S  11
2128  GI_15149480-S   2
19978 GI_42542384-S  12
22144  GI_4503888-S   X
3117  GI_18375521-A   1
9257  GI_28302128-S  11
                     Loc
9260            -5230996
13743          152413604
21840           74520796
22143 c(49221981, 49212423)
9259            -5226077
```

```
2128            189547343
19978           -54634156
22144            49103620
3117           -103114611
9257             -5203271
                                                                           RefSeq
9260                                                c("NM_000184", "NP_000175")
13743                                               c("NM_001711", "NP_001702")
21840                                               c("NM_001134", "NP_001125")
22143                                               c("NM_001476", "NP_001467")
9259                                                c("NM_000559", "NP_000550")
2128                                                c("NM_000090", "NP_000081")
19978                                               c("NM_006928", "NP_008859")
22144                                               c("NM_001477", "NP_001468")
3117  c("NM_001854", "NP_001845", "NM_080629", "NP_542196", "NM_080630", "NP_542197")
9257                                                c("NM_000518", "NP_000509")
                                                                             Name
9260                                                          hemoglobin, gamma G
13743                                                                    biglycan
21840                                                           alpha-fetoprotein
22143                                                                G antigen 6
9259                                                          hemoglobin, gamma A
2128  collagen, type III, alpha 1 (Ehlers-Danlos syndrome type IV, autosomal dominant)
19978                                                     silver homolog (mouse)
22144                                                                G antigen 7B
3117                                                    collagen, type XI, alpha 1
9257                                                             hemoglobin, beta
             IvsP
9260   7.05124284
13743 -6.41967544
21840  5.50531799
22143  5.12600762
9259   6.82821239
2128  -6.21848978
19978  5.20822549
22144  4.59044439
3117  -4.89612179
9257   0.07101028
           IvsNorm
9260   6.775686661
13743 -3.060012038
21840  5.134115849
22143  5.115005439
9259   6.576394085
2128  -6.293353109
19978  4.977646454
22144  4.530531990
3117  -0.001238405
9257  -6.325215556
          PvsNorm        F
9260  -0.27555618 6324.087
```

```
13743  3.35966341 5562.350
21840 -0.37120215 4661.143
22143 -0.01100219 4360.251
9259  -0.25181830 4338.271
2128  -0.07486333 3874.901
19978 -0.23057903 3754.780
22144 -0.05991240 3323.952
3117   4.89488338 3310.720
9257  -6.39622584 3062.805
           P.Value
9260  5.413714e-38
13743 3.266853e-37
21840 3.879873e-36
22143 9.871758e-36
9259  1.059516e-35
2128  5.145356e-35
19978 7.992802e-35
22144 4.393341e-34
3117  4.645256e-34
9257  1.378672e-33
         adj.P.Val
9260  2.560091e-33
13743 7.724312e-33
21840 6.115845e-32
22143 1.002069e-31
9259  1.002069e-31
2128  4.055312e-31
19978 5.399594e-31
22144 2.440772e-30
3117  2.440772e-30
9257  6.519604e-30
```

This vignette was built using the following packages:

```
> sessionInfo()
```

```
R version 2.6.0 Under development (unstable) (2007-08-12 r42483)
x86_64-unknown-linux-gnu

locale:
LC_CTYPE=en_GB.UTF-8;LC_NUMERIC=C;LC_TIME=en_GB.UTF-8;LC_COLLATE=en_GB.UTF-8;LC_MONETARY=en_GB.UTF-8;L

attached base packages:
[1] tools     stats
[3] graphics  grDevices
[5] utils     datasets
[7] methods   base

other attached packages:
 [1] illuminaHumanv1_1.2.0
 [2] lumi_1.3.34
 [3] mgcv_1.3-22
 [4] beadarray_1.5.14
 [5] affy_1.15.7
 [6] preprocessCore_0.99.12
 [7] affyio_1.3.2
 [8] geneplotter_1.15.6
 [9] lattice_0.14-16
[10] annotate_1.13.4
[11] Biobase_1.15.26
[12] limma_2.11.9

loaded via a namespace (and not attached):
[1] grid_2.6.0
[2] KernSmooth_2.22-19
[3] RColorBrewer_0.2-3
```

Acknowledgements

We are grateful to Inma Spiteri for providing the BeadChip data set distributed with the beadarray package. We also thank Julie Addison, Tom Hardcastle, John Marioni, Inma Spiteri and Anna Git for their helpful feedback on the exercises in this tutorial.