Analysis of bead-summary data using beadarray

Mark Dunning and Matt Ritchie

October 11, 2010

Introduction

Illumina have created an alternative microarray technology (BeadArray) based on randomly arranged beads. A specific oligonucleotide sequence is assigned to each *bead type*, which is replicated about 30 times on an array. A series of decoding hybridisations is used to identify each bead on the array. The high degree of replication makes robust measurements for each bead type possible.

BeadArrays are used in many applications, including gene expression studies, SNP genotyping and methylation profiling and are processed in parallel as a Sentrix Array Matrix (SAM) or BeadChip. A SAM is a plate of 96 uniquely prepared hexagonal BeadArrays, each of which contains around 1,500 bead types. The BeadChip technology comprises a series of rectangular strips on a slide with each strip containing about 24,000 bead types. For example, there are six pairs of strips on each Human-6 BeadChip. Depending on the particular assay used, the data from a BeadArray may be single channel or two-colour.

The data from Illumina BeadArrays is available in different formats. We refer to the raw TIFF images and text files output by the BeadScan software as *bead-level data*. For details on how to use the beadarray package to read in and process this kind of data, refer to the bead-level user's guide which can be launched with the following command

```
> library(beadarray)
> beadarrayUsersGuide(topic = "beadlevel")
```

The second format is produced by Illumina's BeadStudio software. We refer to this output as *bead-summary data* as these files contain summary intensities for each bead type on each array. In this user guide we describe how to process summarised gene expression data from Illumina BeadArrays using the beadarray package. Most of the analysis outlined in this guide can equally be applied to the summary values produced by reading and processing the bead-level data using beadarray.

1 Importing bead-summary data

BeadStudio is Illumina's proprietary software for analysing raw bead-level data from BeadScan. It contains different modules for analysing data from different platforms. For further information on the software and how to export summarised data, refer to the user's manual. In this section we consider how to read in and analyse BeadStudio output from the gene expression module.

We will demonstrate the functionality of beadarray using example data available from Illuminas website

`http://www.switchtoi.com/datasets/asuragenmadqc/AsuragenMAQC_BeadStudioOutput.zip`

This dataset, provided courtesy of Asuragen, Inc., contains three labeling replicates each of the "A" and "B" MAQC samples (6 samples total) hybridized on HumanWG-6 v2 arrays. The following code can be used to read the example data into `R` (provided that the contents of `AsuragenMAQC_BeadStudioOutput.zip` have been extracted to the current working directory).

```
> library(beadarray)
> dataFile = "AsuragenMAQC-probe-raw.txt"
> qcFile = "AsuragenMAQC-controls.txt"
> BSData = readBeadSummaryData(dataFile = dataFile,
+     qcFile = qcFile, controlID = "TargetID",
+     skip = 0, qc.skip = 0, qc.columns = list(exprs = "AVG_Signal"))
```

The arguments of `readBeadSummaryData` can be modified to suit data from versions 1, 2 or 3 of BeadStudio. The current default settings should work for version 3 output. Users may need to change the argument `sep`, which specifies if the `dataFile` is comma or tab delimited and the `skip` argument which specifies the number of lines of header information at the top of the file. Possible `skip` arguments of 0, 7 and 8 have been observed, depending on the version of BeadStudio or way in which the data was exported. The `columns` argument is used to specify which column headings to read from `dataFile` and store in various matrices. Note that the naming of the columns containing the standard errors changed between versions of BeadStudio (earlier versions used BEAD_STDEV in place of BEAD_STDERR - be sure to check that the `columns` argument is appropriate for your data). Equivalent arguments (`qc.sep`, `qc.skip` and `qc.columns`) are used to read the data from `qcFile`. See the help page (`?readBeadSummaryData`) for a complete description of each argument to the function. Control information from Illumina experiments can also be read into beadarray independently using the `readQC` function.

2 The BSData object

`BSData` is an object of type *ExpressionSetIllumina* which is an extension of the *ExpressionSet* class from the `Biobase` package. Objects of this type use a series of slots to store the data.

```
> BSData

ExpressionSetIllumina (storageMode: list)
assayData: 48708 features, 6 samples
  element names: exprs, se.exprs, NoBeads, Detection
protocolData: none
phenoData
  rowNames: SUHRR-1 SUHRR-2 ... Brain-3 (6
    total)
  varLabels: sampleID
  varMetadata: labelDescription
featureData
  featureNames: 20605 3450747 ... NEGATIVE
    (48708 total)
  fvarLabels: ProbeID TargetID ... Status (5
    total)
  fvarMetadata: labelDescription
experimentData: use 'experimentData(object)'
Annotation:
QC Information
 Available Slots:  exprs se.exprs NoBeads controlType
  featureNames:
  sampleNames:

> dim(BSData)
```

```
Features  Samples
   48708        6
```

> slotNames(BSData)

```
 [1] "QC"                 "BeadLevelQC"
 [3] "channelData"        "assayData"
 [5] "phenoData"          "featureData"
 [7] "experimentData"     "annotation"
 [9] "protocolData"       ".__classVersion__"
```

> names(assayData(BSData))

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

> exprs(BSData)[1:5, 1:2]

```
          SUHRR-1   SUHRR-2
20605   141.84740 136.70080
3450747 192.98100 193.43690
3060450 178.69660 176.77010
870131   83.29353  84.60507
5310368  88.46398  96.38007
```

> se.exprs(BSData)[1:5, 1:2]

```
         SUHRR-1  SUHRR-2
20605   3.343350 3.793267
3450747 5.626391 7.036246
3060450 4.347808 4.339916
870131  2.742022 3.125126
5310368 2.777152 3.225832
```

> fData(BSData)[1:10,]

```
        ProbeID TargetID     PROBE_ID SYMBOL
20605     20605   15E1.2 ILMN_1809034 15E1.2
3450747 3450747   2'-PDE ILMN_1660305 2'-PDE
3060450 3060450      76P ILMN_1792173    76P
870131   870131      7A5 ILMN_1762337    7A5
5310368 5310368     A1BG ILMN_1736007   A1BG
770300   770300    A2BP1 ILMN_1787689  A2BP1
3290546 3290546    A2BP1 ILMN_1731507  A2BP1
3420601 3420601    A2BP1 ILMN_1814316  A2BP1
7400044 7400044      A2M ILMN_1745607    A2M
2100711 2100711    A2ML1 ILMN_1757454  A2ML1
        Status
20605     Gene
3450747   Gene
3060450   Gene
870131    Gene
5310368   Gene
```

```
770300    Gene
3290546   Gene
3420601   Gene
7400044   Gene
2100711   Gene
```

The data from the file `SampleProbeProfile.txt` is stored in 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 specified by the `columns` parameter in `readBeadSummaryData`. If the character strings specified in `columns` cannot be matched in the file, the matrix will be filled with `NAs`.

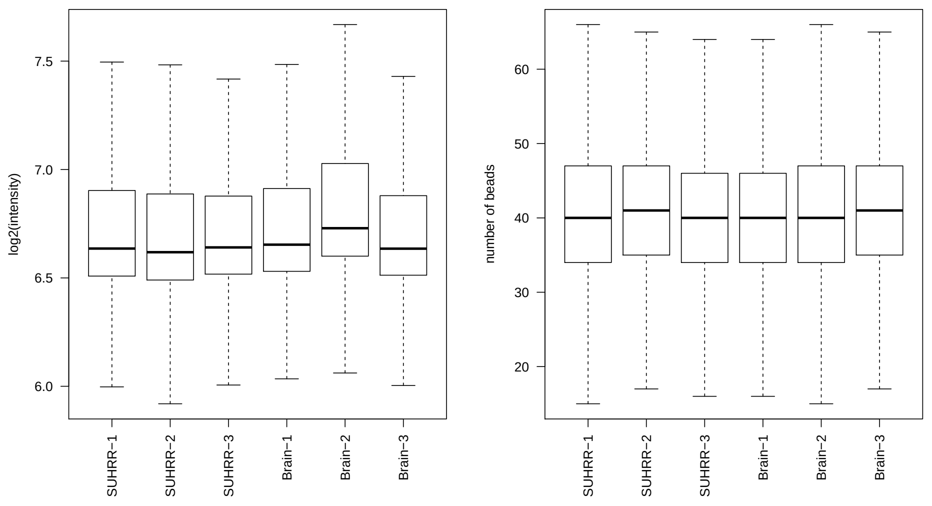
For consistency with the definition of other *ExpressionSet* objects, we now refer to the expression values as the *exprs* matrix which can be accessed using `exprs` and subsetted in the usual manner. Similarly, the standard errors for each bead, which are stored in the *se.exprs* matrix can be accessed using `se.exprs`. The number of beads and detection scores can be accessed using the functions `NoBeads` and `Detection` respectively. The rows names of each of these matrices are from the column in `SampleProbeProfile.txt` that matches the `ProbeID` argument of `readBeadSummaryData`.

Sample information for the experiment can be accessed using `pData` and the `QC` slot contains the control probe intensities.

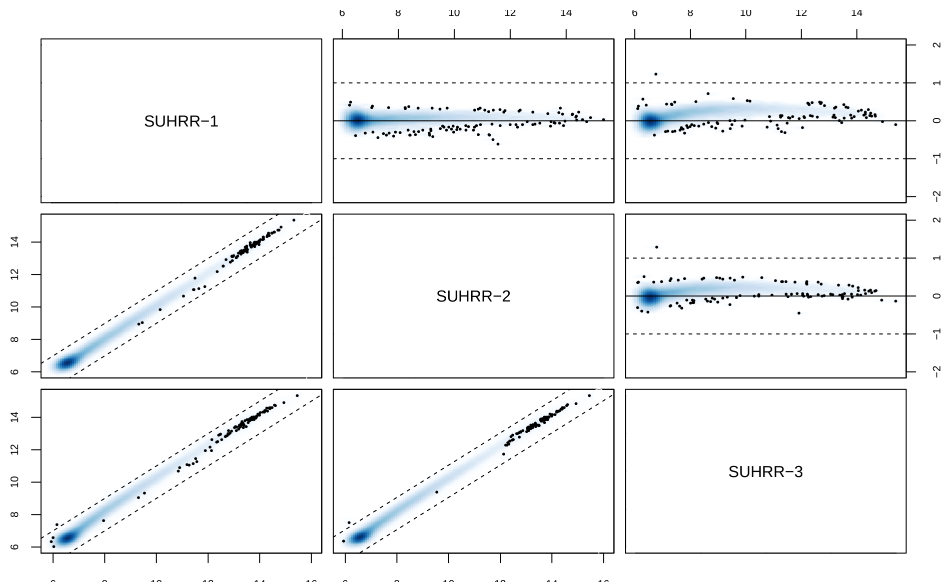
3 Quality assessment and normalisation

Boxplots of intensity levels and the number of beads are useful for quality assessment purposes. Below is the code to produce boxplots of these quantities for each array in the experiment.

```
> par(mfrow = c(1, 2))
> boxplot(as.data.frame(log2(exprs(BSData))),
+     las = 2, outline = FALSE, ylab = "log2(intensity)")
> boxplot(as.data.frame(NoBeads(BSData)),
+     las = 2, outline = FALSE, ylab = "number of beads")
```



```
> plotMAXY(exprs(BSData), arrays = 1:3,
+     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 probe we plot the average of the log_2-intensities from the two arrays on the x-axis and the difference in intensities (log_2-ratios) on the y-axis. For replicate arrays we would expect all probes to be unchanged between the two samples and hence most points on the plot should 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 second array is systematically different to the other replicates and may benefit from normalisation.

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

To correct for differences in expression level across a chip and between chips we need to normalise the signal to make the arrays comparable. The normalisation methods available in the affy package, or variance-stabilising transformation from the lumi package may be applied using the `normaliseIllumina` function. Below we quantile normalise the log_2 transformed data.

```
> BSData.quantile = normaliseIllumina(BSData,
+     method = "quantile", transform = "log2")
> plotMAXY(exprs(BSData.quantile), arrays = 1:3,
+     log = FALSE, pch = 16)
```

4 Differential expression

The differential expression methods available in the limma package can be used to identify differentially expressed genes. The functions `lmFit` and `eBayes` can be applied to the normalised data.

In the example below, we set up a design matrix for the example experiment and fit a linear model to summarise the data from the IC, IH, MC, MD, MT, P and Norm replicates to give one value per condition. We then define contrasts comparing the IH sample to the P sample, IH to Norm and P to Norm and calculate moderated t-statistics with empirical Bayes' shrinkage of the sample variances. In this particular experiment, the IH and P samples are very different and we would expect to see many differentially expressed genes.

```
> library(limma)
> samples = c(rep("UHRR", 3), rep("Brain",
+     3))
> samples

[1] "UHRR"  "UHRR"  "UHRR"  "Brain" "Brain"
[6] "Brain"

> samples = as.factor(samples)
> design = model.matrix(~0 + samples)
> colnames(design) = levels(samples)
> fit = lmFit(exprs(BSData.quantile), design)
> cont.matrix = makeContrasts(BrainDiff = Brain -
+     UHRR, levels = design)
> fit = contrasts.fit(fit, cont.matrix)
> fit$genes = fData(BSData)
> ebFit = eBayes(fit)
> topTable(ebFit, coef = 1, number = 5)

      ProbeID TargetID     PROBE_ID SYMBOL Status
9538  6400079     HBG2 ILMN_1758159   HBG2   Gene
9537  4150187     HBG1 ILMN_1796678   HBG1   Gene
44422 4480474   SNAP91 ILMN_1733648 SNAP91   Gene
38349 1110528      MT3 ILMN_1675947    MT3   Gene
9532  5340674      HBB ILMN_1769753    HBB   Gene
          logFC   AveExpr         t      P.Value
9538  -7.250927 10.374215 -136.8181 2.861178e-38
9537  -7.215677 10.429002 -129.4977 1.170628e-37
44422  6.591989  9.708642  121.3301 6.211713e-37
38349  6.569537 10.021746  120.2875 7.748290e-37
9532   6.041089  9.463751  115.3805 2.251520e-36
         adj.P.Val        B
9538  1.393622e-33 76.63963
9537  2.850949e-33 75.42287
44422 9.435093e-33 73.95359
38349 9.435093e-33 73.75685
9532  2.193341e-32 72.80070
```

For more information about `lmFit` and `eBayes`, refer to the limma documentation.

Annotation

Within Bioconductor, annotation packages are available for most types of Illumina BeadChips. For this experiment, the illuminaHumanv1 package can be used to provide further information on each probe. Custom annotations available from `http://www.compbio.group.cam.ac.uk/Resources/Annotation/` can also be used.

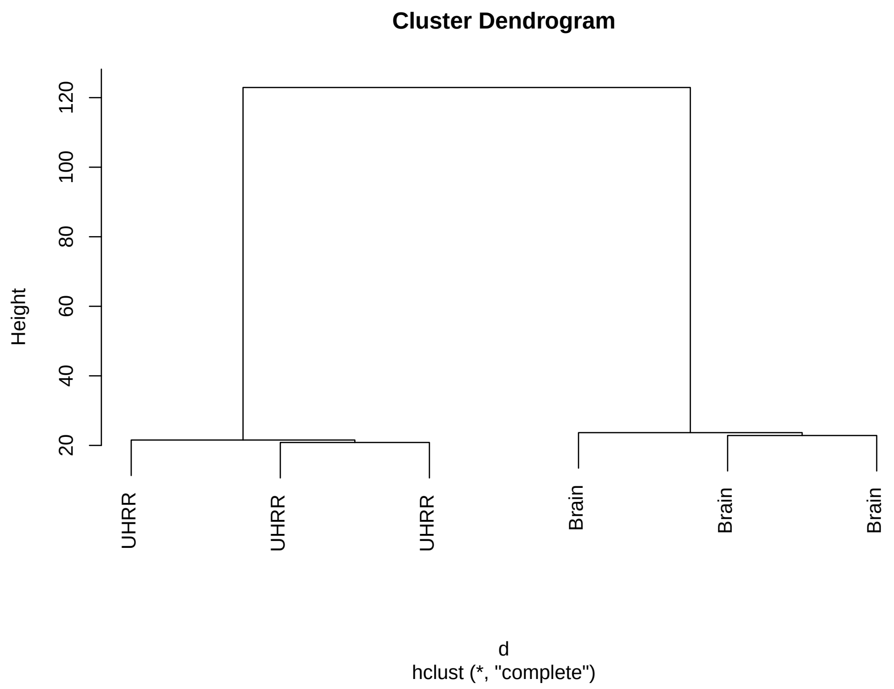
```
> library(illuminaHumanv2.db)
> illuminaHumanv2()
> ids = fData(BSData)[, 3]
> ids = ids[-which(is.na(ids))]
> chr = mget(ids, illuminaHumanv2CHR, ifnotfound = NA)
> chrloc = mget(ids, illuminaHumanv2CHRLOC,
+     ifnotfound = NA)
```

```
> refseq = mget(ids, illuminaHumanv2REFSEQ,
+     ifnotfound = NA)
> genename = mget(ids, illuminaHumanv2GENENAME,
+     ifnotfound = NA)
> symbol = mget(ids, illuminaHumanv2SYMBOL,
+     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),
+     Symbol = as.character(symbol))
> ebFit$genes = anno
> topTable(ebFit)
> write.fit(ebFit, file = "results.txt")
```

5 Further analysis

The clustering functionality available in BeadStudio can be performed in `R` using the `hclust` function once a distance matrix has been defined. The `heatmap` function could also be used.

```
> d = dist(t(exprs(BSData.quantile)))
> plot(hclust(d), labels = samples)
```



This user guide was built using the following packages:

```
> sessionInfo()
```

```
R version 2.13.0 Under development (unstable) (2010-10-05 r53184)
Platform: x86_64-unknown-linux-gnu (64-bit)
```

```
locale:
 [1] LC_CTYPE=en_GB.utf8
 [2] LC_NUMERIC=C
 [3] LC_TIME=en_GB.utf8
 [4] LC_COLLATE=en_GB.utf8
 [5] LC_MONETARY=C
 [6] LC_MESSAGES=en_GB.utf8
 [7] LC_PAPER=en_GB.utf8
 [8] LC_NAME=C
 [9] LC_ADDRESS=C
[10] LC_TELEPHONE=C
[11] LC_MEASUREMENT=en_GB.utf8
[12] LC_IDENTIFICATION=C

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

other attached packages:
[1] limma_3.5.20     beadarray_1.99.1
[3] hwriter_1.2      Biobase_2.9.2

loaded via a namespace (and not attached):
[1] KernSmooth_2.23-3 tools_2.13.0
```