

# Analysis of bead-summary data using `beadarray`

Mark Dunning and Matt Ritchie

April 21, 2008

## 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.

An example data set is included with the `beadarray` package and can be found as a zip file (`BeadSummaryExample.zip`) which can be located with the following command

```
> system.file("demodata", "BeadSummaryExample.zip",
+           package = "beadarray")
```

The example data set consists of 3 Human-6 version 1 BeadChips. These arrays were part of a pilot study into BeadArray technology and 7 different samples (IC, IH, MC, MD, MT, P and Norm) were hybridised to the arrays. MC, MD, MT, P and Norm are all samples from different cell lines and IH and IC are samples provided by Illumina. The non-normalised data and control information were produced using BeadStudio version 3.2.3.

## 1.1 Description of files

Each file contained in `BeadSummaryExample.zip` is briefly described below.

- `SampleProbeProfile.txt` (*required*) - This file contains the raw, non-normalised bead-summary values as output by BeadStudio. The file begins with several lines of header information followed by a data matrix with around 48,000 rows. Each row gives expression measures for a different probe, while the columns give various measurements from different arrays. For each array, we record the summarised expression level (`AVG_Signal`), standard error of the bead replicates (`BEAD_STDERR`), number of beads used (`Avg_NBEADS`) and detection scores (`Detection Pval` - which estimates the probability of a probe being detected above the background level). When exporting data from BeadStudio, the user is able to choose which columns to save. The columns `AVG_Signal`, `BEAD_STDERR`, `Avg_NBEADS` and `Detection Pval`, along with the `ProbeID` column, which contains a unique identifier for each probe, should be selected at a minimum.

It is also 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 at a later stage from other Bioconductor packages, such as `illuminaHumanv1` or from the custom annotations available from <http://www.compbio.group.cam.ac.uk/Resources/Annotation/>.

- `SampleSheet.csv` (*optional*) - This is a file format that Illumina recommend for users of BeadStudio to specify the array IDs and samples hybridised to each array. This file is created by the user.
- `ControlGeneProfile.txt` (*optional*) - Gives the summarised data for each control type on each array as output by BeadStudio, which can be used for quality assessment purposes. The format of the control file differs slightly between BeadStudio version 1 and later versions of the software. The current format is similar to that of `SampleProbeProfile.txt` with columns for the intensities, standard errors, number of beads and detection scores (`AVG_Signal`, `BEAD_STDERR`, `Avg_NBEADS` and `Detection Pval` respectively) for each array. Each row contains data for a different control type, whose identifier is given in the `TargetID` column. Refer to the Illumina documentation for information on each control.

The following code can be used to read the example data into R (provided that the contents of `BeadSummaryExample.zip` have been extracted to the current working directory).

```
> library(beadarray)
> dataFile = "SampleProbeProfile.txt"
> sampleSheet = "SampleSheet.csv"
> qcFile = "ControlGeneProfile.txt"
> BSData = readBeadSummaryData(dataFile = dataFile,
+   qcFile = qcFile, sampleSheet = sampleSheet,
+   controlID = "TargetID")
```

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: 47312 features, 18 samples
  element names: exprs, se.exprs, NoBeads, Detection
phenoData
  rowNames: IH-1, IH-2, ..., Norm-2 (18 total)
  varLabels and varMetadata description:
    Sample_Name: Sample_Name
    Sample_Well: Sample_Well
    ...: ...
    Sentrix_Position: Sentrix_Position
    (7 total)
featureData
  featureNames: 360450, 1690139, ..., 103060372 (47312 total)
  fvarLabels and fvarMetadata description:
    ProbeID: NA
experimentData: use 'experimentData(object)'
Annotation:
QC Information
  Available Slots:  exprs se.exprs Detection NoBeads
  featureNames:  biotin, cy3_hyb, ..., low_stringency_hyb, negative
  sampleNames:  IH-1, IH-2, ..., Norm-1, Norm-2

> dim(BSData)

Features  Samples
  47312      18

> slotNames(BSData)

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

```

> names(assayData(BSData))

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

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

      IH-1      IH-2
360450  87.8092 231.9214
1690139 161.7640 258.5725
5420594 481.1803 499.3547
3060411 633.6545 581.3866
450341 1535.5670 1302.9530

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

      IH-1      IH-2
360450  5.071144 15.53351
1690139 12.018600 18.97802
5420594 21.652110 34.37503
3060411 21.579310 25.81822
450341  42.663560 80.21368

> pData(BSData)[1:2, ]

      Sample_Name Sample_Well Sample_Plate
IH-1      IH-1      NA      NA
IH-2      IH-2      NA      NA
      Sample_Group Pool_ID Satrix_ID
IH-1      IH      NA 1475542114
IH-2      IH      NA 1475542114
      Satrix_Position
IH-1      A
IH-2      C

```

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 `NA`s.

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.

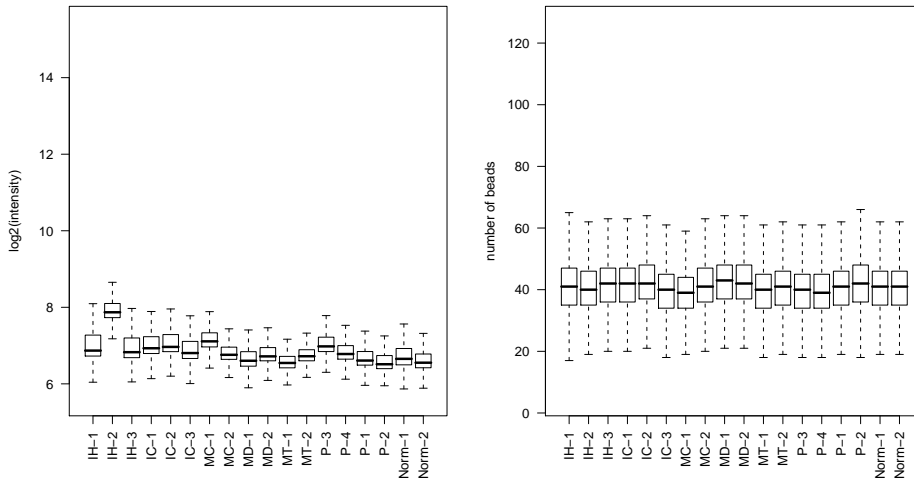
### 3 Quality assessment and normalisation

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")

```

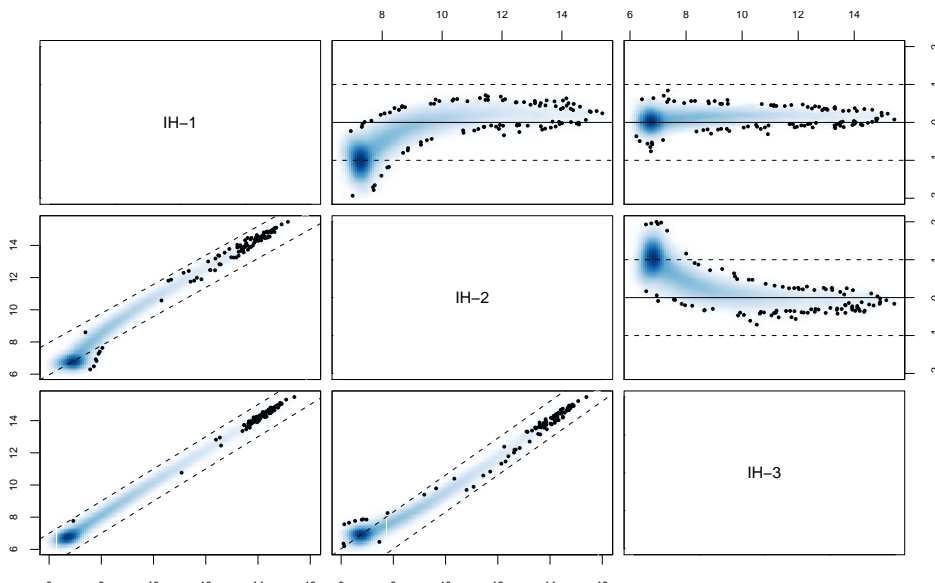


Recall that there are 6 arrays per BeadChip and differences between chips hybridised on different days can be expected (NOTE: in this example, the samples are not ordered by BeadChip. Instead replicate samples appear consecutively). In this experiment, differences in intensity level between arrays are generally small, with the exception of array 2, which has much higher signal than the other arrays. The sample on this array is replicated three times in the experiment and comparing the MA and XY plots for the replicates of this sample using the `plotMAXY` function can be informative.

```

> 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`.

The control probe intensities can be plotted as a further quality diagnostic. To retrieve this control data, which is stored in the `QC` slot, use the `QCInfo` function. `QCInfo` returns a list object containing several matrices which can be accessed using the `$` operator. The row names of each matrix indicate the control type (as specified by the column in `qcFile` matching the `controlID` argument of `readBeadSummaryData`). In the following example we plot summary intensities for each control type on each array.

```
> QC = QCInfo(BSData)
> names(QC)

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

> rownames(QC$exprs)

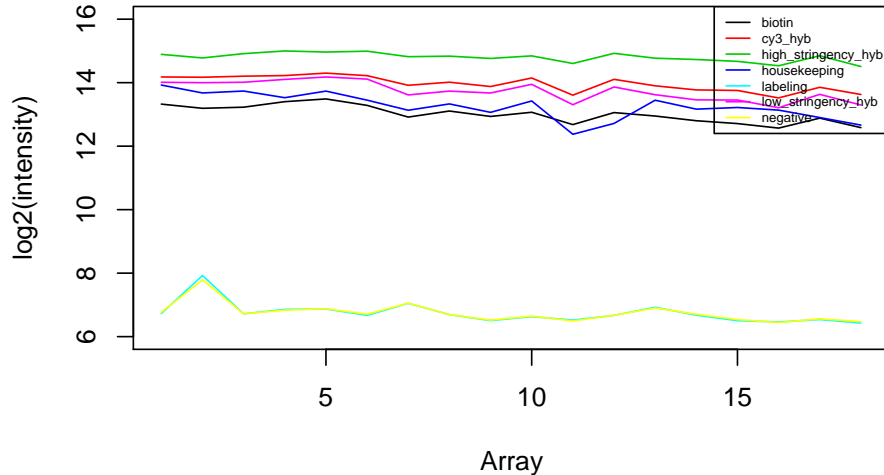
[1] "biotin"           "cy3_hyb"
[3] "high_stringency_hyb" "housekeeping"
[5] "labeling"         "low_stringency_hyb"
[7] "negative"

> QC$exprs[, 1:2]

              IH-1      IH-2
biotin          10255.0200  9358.6680
cy3_hyb          18556.4400 18442.0200
high_stringency_hyb 30390.5100 28120.1700
housekeeping     15551.8900 13086.8300
labeling         105.9794   243.0619
low_stringency_hyb 16526.4700 16378.6700
negative         108.7406   221.2646

> numContr = nrow(QC$exprs)
> plot(log2(QC$exprs[1, ]), type = "l",
+      xlab = "Array", ylab = "log2(intensity)",
+      main = "Control probes", cex = 0.5,
+      ylim = c(6, 16))
> for (i in 2:numContr) {
+   points(log2(QC$exprs[i, ]), type = "l",
+         col = i)
+ }
> legend("topright", legend = rownames(QC$exprs),
+       col = 1:numContr, lty = 1, cex = 0.5)
```

## Control probes



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.

```
> samples = pData(BSData.quantile)$Sample_Group
> samples

 [1] "IH"  "IH"  "IH"  "IC"  "IC"  "IC"
 [7] "MC"  "MC"  "MD"  "MD"  "MT"  "MT"
[13] "P"   "P"   "P"   "P"   "Norm" "Norm"

> samples = as.factor(samples)
> design = model.matrix(~0 + samples)
> colnames(design) = levels(samples)
```

```

> fit = lmFit(exprs(BSDData.quantile), design)
> cont.matrix = makeContrasts(IHvsP = IH -
+   P, IHvsNorm = IH - Norm, PvsNorm = P -
+   Norm, levels = design)
> fit = contrasts.fit(fit, cont.matrix)
> ebFit = eBayes(fit)
> topTable(ebFit, coef = 1, number = 5)

```

	ID	logFC	AveExpr	t	
	13746	2030280	-6.400270	8.646376	-88.39647
	9262	4570301	7.085818	9.047241	86.78303
	21843	3800358	5.497822	8.475639	74.16995
	22146	3800435	5.162826	9.459757	72.14053
	9261	6900332	6.805046	9.031485	69.66278
	P.Value	adj.P.Val	B		
	13746	1.351034e-33	5.135877e-29	64.73139	
	9262	2.171067e-33	5.135877e-29	64.36352	
	21843	1.235732e-31	1.948832e-27	61.09813	
	22146	2.522020e-31	2.983045e-27	60.49917	
	9261	6.193555e-31	5.860590e-27	59.73601	

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(illuminaHumanv1)
> illuminaHumanv1()

```

```

Quality control information for illuminaHumanv1
Date built: Created: Mon Sep 24 15:18:40 2007

```

```

Number of probes: 47297

```

```

Probe number mismatch: illuminaHumanv1ACCNUM; illuminaHumanv1CHRLLOC; illuminaHumanv1CHR; illuminaHumanv1ENTREZID; illuminaHumanv1ENZYME; illuminaHumanv1GENENAME; illuminaHumanv1GO; illuminaHumanv1MAP; illuminaHumanv1OMIM; illuminaHumanv1PATH; illuminaHumanv1PMID; illuminaHumanv1REFSEQ

```

```

Probe mismatch: None

```

```

Mappings found for probe based rda files:

```

```

illuminaHumanv1ACCNUM found 47290 of 47297
illuminaHumanv1CHRLLOC found 18734 of 47297
illuminaHumanv1CHR found 19274 of 47297
illuminaHumanv1ENTREZID found 19276 of 47297
illuminaHumanv1ENZYME found 2217 of 47297
illuminaHumanv1GENENAME found 19276 of 47297
illuminaHumanv1GO found 16931 of 47297
illuminaHumanv1MAP found 19186 of 47297
illuminaHumanv1OMIM found 13004 of 47297
illuminaHumanv1PATH found 4751 of 47297
illuminaHumanv1PMID found 19060 of 47297
illuminaHumanv1REFSEQ found 19276 of 47297

```



```

illuminaHumanv1SUMFUNC found 0 of 47297
illuminaHumanv1SYMBOL found 19276 of 47297
illuminaHumanv1UNIGENE found 19178 of 47297
Mappings found for non-probe based rda files:
  illuminaHumanv1CHRENGTHS found 25
  illuminaHumanv1ENZYME2PROBE found 817
  illuminaHumanv1GO2ALLPROBES found 8459
  illuminaHumanv1GO2PROBE found 6128
  illuminaHumanv1ORGANISM found 1
  illuminaHumanv1PATH2PROBE found 199
  illuminaHumanv1PFAM found 15332
  illuminaHumanv1PMID2PROBE found 158933
  illuminaHumanv1PROSITE found 10858

> 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
> write.fit(ebFit, file = "results.txt")

```

## 5 Further analysis

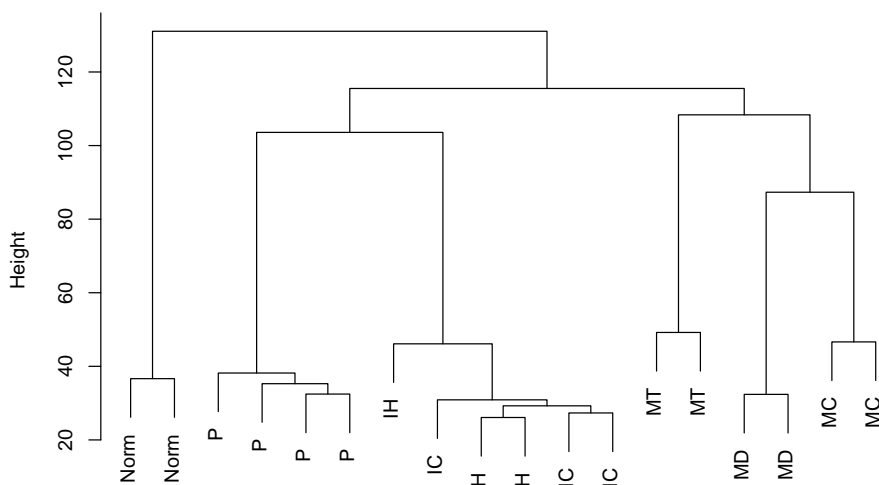
The clustering functionality available in BeadStudio can be performed in R using the `hclust` function once a distance matrix has been defined. The `heatmap` function could also be used.

```

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

```

### Cluster Dendrogram



d  
hclust(\*, "complete")

This user guide was built using the following packages:

```
> sessionInfo()
```

```
R version 2.7.0 Under development (unstable) (2008-02-28 r44635)  
x86_64-unknown-linux-gnu
```

```
locale:
```

```
LC_CTYPE=en_GB.UTF-8;LC_NUMERIC=C;LC_TIME=C;LC_COLLATE=C;LC_MONETARY=C;LC_MESSAGES=C;LC_PAPER=C;LC_NAME
```

```
attached base packages:
```

```
[1] tools      stats      graphics  grDevices  
[5] utils      datasets  methods   base
```

```
other attached packages:
```

```
[1] weaver_1.5.0          codetools_0.1-3  
[3] digest_0.3.1         illuminaHumanv1_1.4.0  
[5] beadarray_1.7.11     affy_1.17.8  
[7] preprocessCore_1.1.5 affyio_1.7.13  
[9] geneplotter_1.17.7   lattice_0.17-6  
[11] annotate_1.17.11     xtable_1.5-2  
[13] AnnotationDbi_1.1.24 RSQLite_0.6-8  
[15] DBI_0.2-4            Biobase_1.99.1  
[17] limma_2.13.5
```

```
loaded via a namespace (and not attached):
```

```
[1] KernSmooth_2.22-22 RColorBrewer_1.0-2  
[3] grid_2.7.0         tcltk_2.7.0
```

## Acknowledgements

We are grateful to Inma Spiteri for providing the example data set distributed with the `beadarray` package.