

# Analysis of Bead-summary Data using beadarray

*Mark Dunning*

April 26, 2023

## Contents

### 1 Introduction

---

The BeadArray technology involves randomly arranged arrays of beads, with beads having the same probe sequence attached colloquially known as a bead-type. BeadArrays are combined in parallel on either a rectangular chip (Bead-Chip) or a matrix of 8 by 12 hexagonal arrays (Sentrix Array Matrix or SAM). The BeadChip is further divided into strips on the surface known as sections, with each section giving rise to a different image when scanned by BeadScan. These images, and associated text files, comprise the raw data for a beadarray analysis. However, for BeadChips, the number of sections assigned to each biological sample may vary from 1 on HumanHT12 chips, 2 on HumanWG6 chips or sometimes ten or more for SNP chips with large numbers of SNPs being investigated.

This vignette demonstrates the analysis of bead summary data using beadarray. The recommended approach to obtain these data is to start with bead-level data and follow the steps illustrated in the vignette [beadlevel.pdf](#) distributed with [beadarray](#). If bead-level data are not available, the output of Illumina's BeadStudio or GenomeStudio can be read by [beadarray](#). Example code to do this is provided at the end of this vignette. However, the same object types are produced from either of these routes and the same functionality is available.

To make the most use of the code in this vignette, you will need to install the [beadarrayExampleData](#) and [illuminaHumanv3.db](#) packages from Bioconductor. We use the [BiocManager](#) package to install these:

```
install.packages("BiocManager")
BiocManager::install(c("beadarrayExampleData", "illuminaHumanv3.db"))
```

## Analysis of Bead-summary Data using beadarray

The code used to produce these example data is given in the vignette of [beadarrayExampleData](#), which follow similar steps to those described in the [beadlevel.pdf](#) vignette of [beadarray](#). The following commands give a basic description of the data.

```
library("beadarray")

require(beadarrayExampleData)

data(exampleSummaryData)

exampleSummaryData
## ExpressionSetIllumina (storageMode: list)
## assayData: 49576 features, 12 samples
##   element names: exprs, se.exprs, nObservations
## protocolData: none
## phenoData
##   rowNames: 4613710017_B 4613710052_B ... 4616494005_A (12 total)
##   varLabels: sampleID SampleFac
##   varMetadata: labelDescription
## featureData
##   featureNames: ILMN_1802380 ILMN_1893287 ... ILMN_1846115 (49576
##     total)
##   fvarLabels: ArrayAddressID IlluminaID Status
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
## Annotation: Humanv3
## QC Information
## Available Slots:
##   QC Items: Date, Matrix, ..., SampleGroup, numBeads
##   sampleNames: 4613710017_B, 4613710052_B, ..., 4616443136_A, 4616494005_A
```

Summarized data are stored in an object of type *ExpressionSetIllumina* which is an extension of the *ExpressionSet* class developed by the Bioconductor team as a container for data from high-throughput assays. Objects of this type use a series of slots to store the data. For consistency with the definition of other *ExpressionSet* objects, we refer to the expression values as the `exprs` matrix (this stores the probe-specific average intensities) which can be accessed using `exprs` and subset in the usual manner. The `se.exprs` matrix, which stores the probe-specific variability can be accessed using `se.exprs`. You may notice that the expression values have already been transformed to the  $\log_2$  scale,

## Analysis of Bead-summary Data using beadarray

which is an option in the `summarize` function in *beadarray*. Data exported from BeadStudio or GenomeStudio will usually be un-transformed and on the scale 0 to  $2^{16}$ .

```
exprs(exampleSummaryData)[1:5,1:5]

##           G:4613710017_B G:4613710052_B G:4613710054_B G:4616443079_B
## ILMN_1802380      8.454468      8.616796      8.523001      8.420796
## ILMN_1893287      5.388161      5.419345      5.162849      5.133287
## ILMN_1736104      5.268626      5.457679      5.012766      4.988511
## ILMN_1792389      6.767519      7.183788      6.947624      7.168571
## ILMN_1854015      5.556947      5.721614      5.595413      5.520391
##           G:4616443093_B
## ILMN_1802380      8.527748
## ILMN_1893287      5.221987
## ILMN_1736104      5.284026
## ILMN_1792389      7.386435
## ILMN_1854015      5.558717

se.exprs(exampleSummaryData)[1:5,1:5]

##           G:4613710017_B G:4613710052_B G:4613710054_B G:4616443079_B
## ILMN_1802380      0.2833023      0.3367157      0.2750020      0.4141796
## ILMN_1893287      0.3963681      0.3882834      0.5516421      0.6761106
## ILMN_1736104      0.4704854      0.4951260      0.4031143      0.5276266
## ILMN_1792389      0.4038533      0.4728013      0.5032908      0.3447242
## ILMN_1854015      0.5663066      0.3783570      0.5511991      0.5358812
##           G:4616443093_B
## ILMN_1802380      0.3581862
## ILMN_1893287      0.4448673
## ILMN_1736104      0.4864355
## ILMN_1792389      0.3951935
## ILMN_1854015      0.6748219
```

## 2 feature and pheno data

The `fData` and `pData` functions are useful shortcuts to find more information about the features (rows) and samples (columns) in the summary object. These annotations are created automatically whenever a bead-level data is summarized (see `beadlevel.pdf`) or read from a BeadStudio file. The `fData` will be added to later, but initially contains information on whether each probe is a control or not. In this example the `phenoData` denotes the sample group for each array; either Brain or UHRR (Universal Human Reference RNA).

## Analysis of Bead-summary Data using beadarray

```
head(fData(exampleSummaryData))

##           ArrayAddressID  IlluminaID  Status
## ILMN_1802380           10008 ILMN_1802380 regular
## ILMN_1893287           10010 ILMN_1893287 regular
## ILMN_1736104           10017 ILMN_1736104 regular
## ILMN_1792389           10019 ILMN_1792389 regular
## ILMN_1854015           10020 ILMN_1854015 regular
## ILMN_1904757           10021 ILMN_1904757 regular

table(fData(exampleSummaryData)[, "Status"])

##
##           biotin           cy3_hyb
##           2           2
## cy3_hyb, low_stringency_hyb housekeeping
##           4           7
##           labeling low_stringency_hyb
##           2           4
##           negative           regular
##           759           48796

pData(exampleSummaryData)

##           sampleID SampleFac
## 4613710017_B 4613710017_B  UHRR
## 4613710052_B 4613710052_B  UHRR
## 4613710054_B 4613710054_B  UHRR
## 4616443079_B 4616443079_B  UHRR
## 4616443093_B 4616443093_B  UHRR
## 4616443115_B 4616443115_B  UHRR
## 4616443081_B 4616443081_B  Brain
## 4616443081_H 4616443081_H  Brain
## 4616443092_B 4616443092_B  Brain
## 4616443107_A 4616443107_A  Brain
## 4616443136_A 4616443136_A  Brain
## 4616494005_A 4616494005_A  Brain
```

### 3 Subsetting the data

There are various way to subset an *ExpressionSetIllumina* object, each of which returns an *ExpressionSetIllumina* with the same slots, but different dimensions. When bead-level data are summarized by *beadarray* there is an option to apply

## Analysis of Bead-summary Data using beadarray

different transformation options, and save the results as different channels in the resultant object. For instance, if summarizing two-colour data one might be interested in summarizing the red and green channels, or some combination of the two, separately. Both  $\log_2$  and un-logged data are stored in the `exampleSummaryData` object and can be accessed by using the `channel` function. Both the rows and columns in the resultant `ExpressionSetIllumina` object are kept in the same order.

```
channelNames(exampleSummaryData)

## [1] "G"      "G.ul"

exampleSummaryData.log2 <- channel(exampleSummaryData, "G")
exampleSummaryData.unlogged <- channel(exampleSummaryData, "G.ul")

sampleNames(exampleSummaryData.log2)

## [1] "4613710017_B" "4613710052_B" "4613710054_B" "4616443079_B"
## [5] "4616443093_B" "4616443115_B" "4616443081_B" "4616443081_H"
## [9] "4616443092_B" "4616443107_A" "4616443136_A" "4616494005_A"

sampleNames(exampleSummaryData.unlogged)

## [1] "4613710017_B" "4613710052_B" "4613710054_B" "4616443079_B"
## [5] "4616443093_B" "4616443115_B" "4616443081_B" "4616443081_H"
## [9] "4616443092_B" "4616443107_A" "4616443136_A" "4616494005_A"

exprs(exampleSummaryData.log2)[1:10,1:3]

##           4613710017_B 4613710052_B 4613710054_B
## ILMN_1802380      8.454468      8.616796      8.523001
## ILMN_1893287      5.388161      5.419345      5.162849
## ILMN_1736104      5.268626      5.457679      5.012766
## ILMN_1792389      6.767519      7.183788      6.947624
## ILMN_1854015      5.556947      5.721614      5.595413
## ILMN_1904757      5.421553      5.320500      5.522316
## ILMN_1740305      5.417821      5.623998      5.720007
## ILMN_1665168      5.321087      5.155455      4.967601
## ILMN_2375156      5.894207      6.076418      5.638877
## ILMN_1705423      5.426463      4.806624      5.357688

exprs(exampleSummaryData.unlogged)[1:10,1:3]

##           4613710017_B 4613710052_B 4613710054_B
## ILMN_1802380     356.88235     396.46875     367.81481
## ILMN_1893287      40.85000      44.29167      38.42105
```

## Analysis of Bead-summary Data using beadarray

## ILMN_1736104	40.53333	46.50000	33.46154
## ILMN_1792389	112.90909	153.17647	122.65000
## ILMN_1854015	50.47059	53.26087	51.57143
## ILMN_1904757	41.45833	42.10000	49.92593
## ILMN_1740305	38.45455	51.50000	46.21429
## ILMN_1665168	42.38889	37.95000	30.46154
## ILMN_2375156	61.47368	72.73913	52.46154
## ILMN_1705423	42.38889	28.14286	38.62500

As we have seen, the expression matrix of the `ExpressionSetIllumina` object can be subset by column or row. In fact, the same subset operations can be performed on the `ExpressionSetIllumina` object itself. In the following code, notice how the number of samples and features changes in the output.

```
exampleSummaryData.log2[,1:4]

## ExpressionSetIllumina (storageMode: list)
## assayData: 49576 features, 4 samples
##   element names: exprs, se.exprs, nObservations
## protocolData: none
## phenoData
##   rowNames: 4613710017_B 4613710052_B 4613710054_B 4616443079_B
##   varLabels: sampleID SampleFac
##   varMetadata: labelDescription
## featureData
##   featureNames: ILMN_1802380 ILMN_1893287 ... ILMN_1846115 (49576
##     total)
##   fvarLabels: ArrayAddressID IlluminaID Status
##   fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
## Annotation: Humanv3
## QC Information
## Available Slots:
##   QC Items: Date, Matrix, ..., SampleGroup, numBeads
##   sampleNames: 4613710017_B, 4613710052_B, 4613710054_B, 4616443079_B

exampleSummaryData.log2[1:10,]

## ExpressionSetIllumina (storageMode: list)
## assayData: 10 features, 12 samples
##   element names: exprs, se.exprs, nObservations
## protocolData: none
## phenoData
##   rowNames: 4613710017_B 4613710052_B ... 4616494005_A (12 total)
```

## Analysis of Bead-summary Data using beadarray

```
## varLabels: sampleID SampleFac
## varMetadata: labelDescription
## featureData
## featureNames: ILMN_1802380 ILMN_1893287 ... ILMN_1705423 (10
## total)
## fvarLabels: ArrayAddressID IlluminaID Status
## fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
## Annotation: Humanv3
## QC Information
## Available Slots:
## QC Items: Date, Matrix, ..., SampleGroup, numBeads
## sampleNames: 4613710017_B, 4613710052_B, ..., 4616443136_A, 4616494005_A
```

The object can also be subset by a vector of characters which must correspond to the names of features (i.e. row names). Currently, no analogous functions is available to subset by sample.

```
randIDs <- sample(featureNames(exampleSummaryData), 1000)

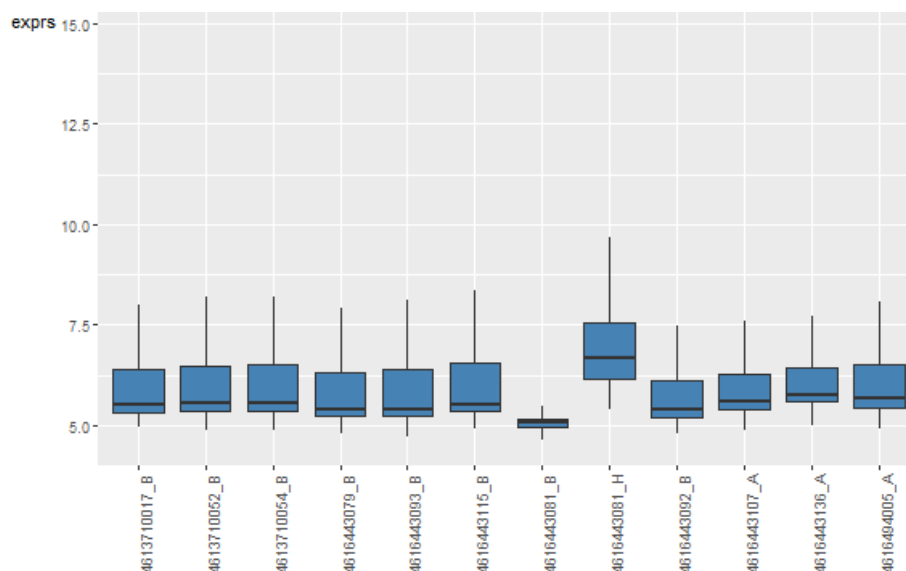
exampleSummaryData[randIDs,]

## ExpressionSetIllumina (storageMode: list)
## assayData: 1000 features, 12 samples
## element names: exprs, se.exprs, nObservations
## protocolData: none
## phenoData
## rowNames: 4613710017_B 4613710052_B ... 4616494005_A (12 total)
## varLabels: sampleID SampleFac
## varMetadata: labelDescription
## featureData
## featureNames: ILMN_1768975 ILMN_1721170 ... ILMN_1789641 (1000
## total)
## fvarLabels: ArrayAddressID IlluminaID Status
## fvarMetadata: labelDescription
## experimentData: use 'experimentData(object)'
## Annotation: Humanv3
## QC Information
## Available Slots:
## QC Items: Date, Matrix, ..., SampleGroup, numBeads
## sampleNames: 4613710017_B, 4613710052_B, ..., 4616443136_A, 4616494005_A
```

### 4 Exploratory analysis using boxplots

Boxplots of intensity levels and the number of beads are useful for quality assessment purposes. `beadarray` includes a modified version of the `boxplot` function that can take any valid *ExpressionSetIllumina* object and plot the expression matrix by default. For these examples we plot just a subset of the original `exampleSummaryData` object using random row IDs.

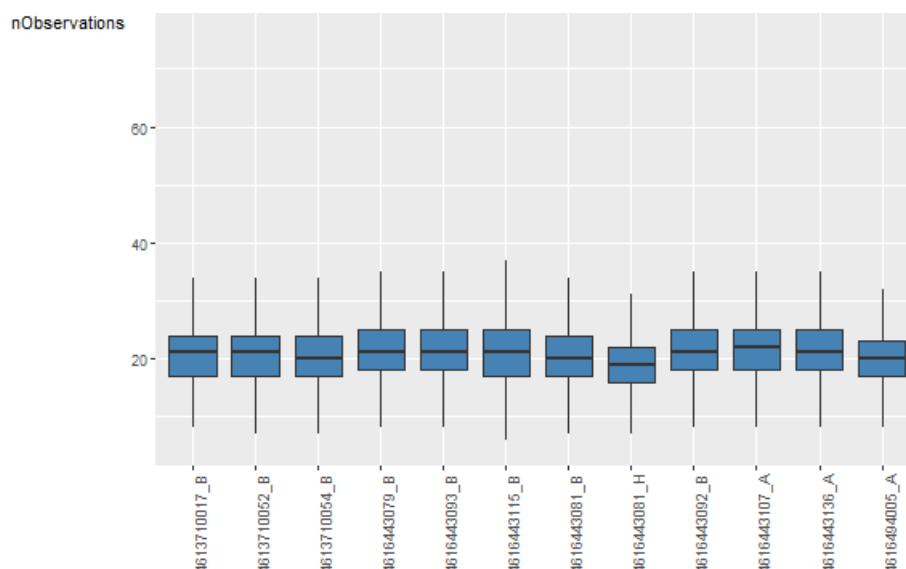
```
boxplot(exampleSummaryData.log2[randIDs,])
```



The function can also plot other `assayData` items, such as the number of observations.

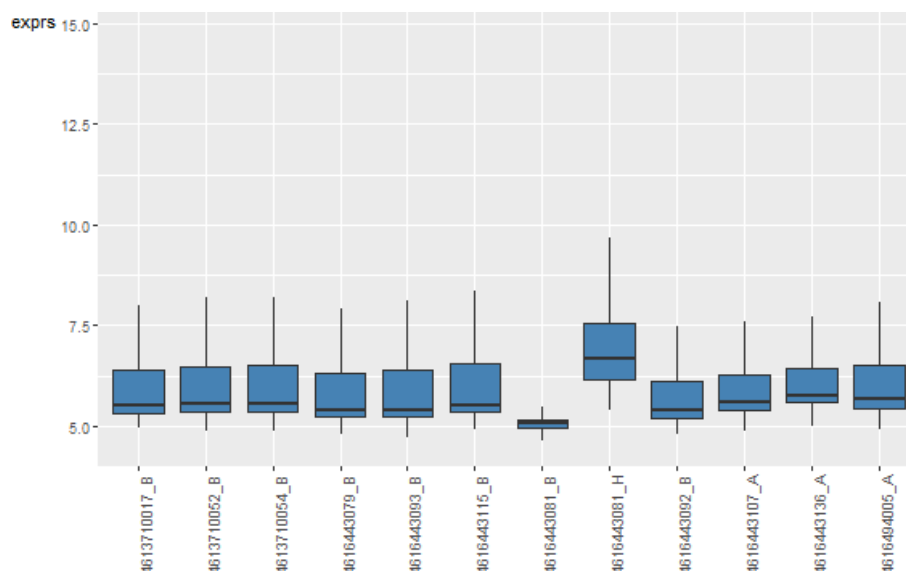
```
boxplot(exampleSummaryData.log2[randIDs,], what="nObservations")
```

## Analysis of Bead-summary Data using beadarray



The default boxplot plots a separate box for each array, but often it is beneficial for compare expression levels between different sample groups. If this information is stored in the *phenoData* slot it can be incorporated into the plot. The following compares the overall expression level between UHRR and Brain samples.

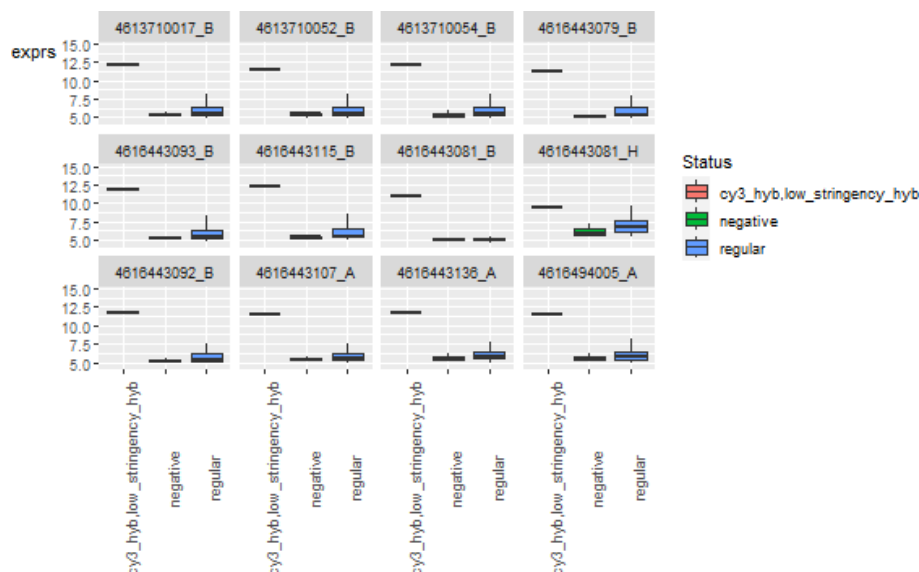
```
boxplot(exampleSummaryData.log2[randIDs,], SampleGroup="SampleFac")
```



In a similar manner, we may wish to visualize the differences between sample groups for particular probe groups. As a simple example, we look at the difference between negative controls and regular probes for each array. You should notice that the negative controls are consistently lower (as expected) with the exception of array 4616443081\_B.

## Analysis of Bead-summary Data using beadarray

```
boxplot(exampleSummaryData.log2[randIDs,], probeFactor = "Status")
```



Extra feature annotation is available from annotation packages in Bioconductor, and [beadarray](#) includes functionality to extract these data from the annotation packages. The annotation of the object must be set in order that the correct annotation package can be loaded. For example, the `exampleSummaryData` object was generated from Humanv3 data so the [illuminaHumanv3.db](#) package must be present. The `addFeatureData` function annotates all features of an `ExpressionSetIllumina` object using particular mappings from the [illuminaHumanv3.db](#) package. To see which mappings are available you can use the `illuminaHumanv3()` function, or equivalent from other packages.

```
annotation(exampleSummaryData)

## [1] "Humanv3"

exampleSummaryData.log2 <- addFeatureData(exampleSummaryData.log2,
toAdd = c("SYMBOL", "PROBEQUALITY", "CODINGZONE", "PROBESEQUENCE", "GENOMICLOCATION"))

head(fData(exampleSummaryData.log2))

##           Row.names ArrayAddressID  IlluminaID  Status SYMBOL
## ILMN_1802380 ILMN_1802380      10008 ILMN_1802380 regular  RERE
## ILMN_1893287 ILMN_1893287      10010 ILMN_1893287 regular  <NA>
## ILMN_1736104 ILMN_1736104      10017 ILMN_1736104 regular  <NA>
## ILMN_1792389 ILMN_1792389      10019 ILMN_1792389 regular  ARK2C
## ILMN_1854015 ILMN_1854015      10020 ILMN_1854015 regular  <NA>
## ILMN_1904757 ILMN_1904757      10021 ILMN_1904757 regular  <NA>
```

## Analysis of Bead-summary Data using beadarray

```
##          PROBEQUALITY      CODINGZONE
## ILMN_1802380      Perfect Transcriptomic
## ILMN_1893287        Bad Transcriptomic?
## ILMN_1736104        Bad      Intergenic
## ILMN_1792389      Perfect Transcriptomic
## ILMN_1854015        Bad      Intergenic
## ILMN_1904757 Perfect*** Transcriptomic?
##
##                                     PROBESEQUENCE
## ILMN_1802380 GCCCTGACCTTCATGGTGTCTTTGAAGCCCAACCACTCGGTTTCCTTCGG
## ILMN_1893287 GGATTTCTTACACTCTCCACTTCTGAATGCTTGAAACACTTGCCATGCT
## ILMN_1736104 TGCCATCTTTGCTCCACTGTGAGAGGCTGCTCACACCACCCCCTACATGC
## ILMN_1792389 CTGTAGCAACGTCTGTGAGGCCCCCTTGTTTCATCTCCTGCGCGCGTA
## ILMN_1854015 GCAGAAAACCATGAGCTGAAATCTCTACAGGAACCACTGCTGGGGTAGGG
## ILMN_1904757 AGCTGTACCGTGCGGAGGCTTGGTCCTCTTGCCCCATTTGTGTGATGTCT
##
##                      GENOMICLOCATION
## ILMN_1802380      chr1:8412758:8412807:-
## ILMN_1893287      chr9:42489407:42489456:+
## ILMN_1736104      chr3:134572184:134572223:-
## ILMN_1792389      chr18:44040244:44040293:+
## ILMN_1854015      chr3:160827837:160827885:+
## ILMN_1904757      chr3:197872267:197872316:+

illuminaHumanv3()

## #####Mappings based on RefSeqID####
## Quality control information for illuminaHumanv3:
##
##
## This package has the following mappings:
##
## illuminaHumanv3ACCNUM has 31857 mapped keys (of 49576 keys)
## illuminaHumanv3ALIAS2PROBE has 63289 mapped keys (of 208700 keys)
## illuminaHumanv3CHR has 29550 mapped keys (of 49576 keys)
## illuminaHumanv3CHRLNGTHS has 93 mapped keys (of 711 keys)
## illuminaHumanv3CHRLLOC has 29354 mapped keys (of 49576 keys)
## illuminaHumanv3CHRLLOCEND has 29354 mapped keys (of 49576 keys)
## illuminaHumanv3ENSEMBL has 29154 mapped keys (of 49576 keys)
## illuminaHumanv3ENSEMBL2PROBE has 20997 mapped keys (of 39839 keys)
## illuminaHumanv3ENTREZID has 29551 mapped keys (of 49576 keys)
## illuminaHumanv3ENZYME has 3526 mapped keys (of 49576 keys)
## illuminaHumanv3ENZYME2PROBE has 967 mapped keys (of 975 keys)
## illuminaHumanv3GENENAME has 29551 mapped keys (of 49576 keys)
## illuminaHumanv3GO has 26989 mapped keys (of 49576 keys)
```

## Analysis of Bead-summary Data using beadarray

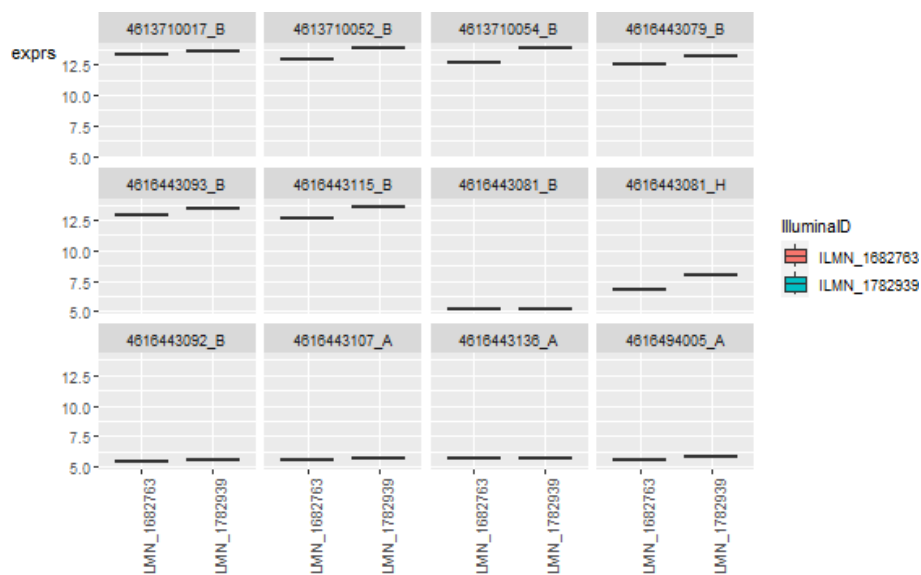
```
## illuminaHumanv3G02ALLPROBES has 19471 mapped keys (of 22934 keys)
## illuminaHumanv3G02PROBE has 15200 mapped keys (of 18944 keys)
## illuminaHumanv3MAP has 29402 mapped keys (of 49576 keys)
## illuminaHumanv3OMIM has 21943 mapped keys (of 49576 keys)
## illuminaHumanv3PATH has 9180 mapped keys (of 49576 keys)
## illuminaHumanv3PATH2PROBE has 229 mapped keys (of 229 keys)
## illuminaHumanv3PMID has 29329 mapped keys (of 49576 keys)
## illuminaHumanv3PMID2PROBE has 439054 mapped keys (of 766627 keys)
## illuminaHumanv3REFSEQ has 29551 mapped keys (of 49576 keys)
## illuminaHumanv3SYMBOL has 29551 mapped keys (of 49576 keys)
## illuminaHumanv3UNIPROT has 27704 mapped keys (of 49576 keys)
##
##
## Additional Information about this package:
##
## DB schema: HUMANCHIP_DB
## DB schema version: 2.1
## Organism: Homo sapiens
## Date for NCBI data: 2015-Mar17
## Date for GO data: 20150314
## Date for KEGG data: 2011-Mar15
## Date for Golden Path data: 2010-Mar22
## Date for Ensembl data: 2015-Mar13
## #####Custom Mappings based on probe sequence####
## illuminaHumanv3ARRAYADDRESS()
## illuminaHumanv3NUID()
## illuminaHumanv3PROBEQUALITY()
## illuminaHumanv3CODINGZONE()
## illuminaHumanv3PROBESEQUENCE()
## illuminaHumanv3SECONDMATCHES()
## illuminaHumanv3OTHERGENOMICMATCHES()
## illuminaHumanv3REPEATMASK()
## illuminaHumanv3OVERLAPPINGSNP()
## illuminaHumanv3ENTREZREANNOTATED()
## illuminaHumanv3GENOMICLOCATION()
## illuminaHumanv3SYMBOLREANNOTATED()
## illuminaHumanv3REPORTERGROUPNAME()
## illuminaHumanv3REPORTERGROUPID()
## illuminaHumanv3ENSEMBLREANNOTATED()
```

## Analysis of Bead-summary Data using beadarray

If we suspect that a particular gene may be differentially expressed between conditions, we can subset the *ExpressionSetIllumina* object to just include probes that target the gene, and plot the response of these probes against the sample groups. Furthermore, the different probes can be distinguished using the `probeFactor` parameter.

```
ids <- which(fData(exampleSummaryData.log2)[, "SYMBOL"] == "ALB")

boxplot(exampleSummaryData.log2[ids, ],
        SampleGroup = "SampleFac", probeFactor = "IlluminaID")
```



### 4.1 A note about ggplot2

The `boxplot` function in *beadarray* creates graphics using the *ggplot2* package rather than the R base graphics system. Therefore, the standard way of manipulating graphics using `par` and `mflow` etc will not work with the output of `boxplot`. However, the *ggplot2* package has equivalent functionality and is a more powerful and flexible system. There are numerous tutorials on how to use the *ggplot2* package, which is beyond the scope of this vignette. In the below code, we assign the results of `boxplot` to objects that we combine using the *gridExtra* package. The code also demonstrates how aspects of the plot can be altered programatically.

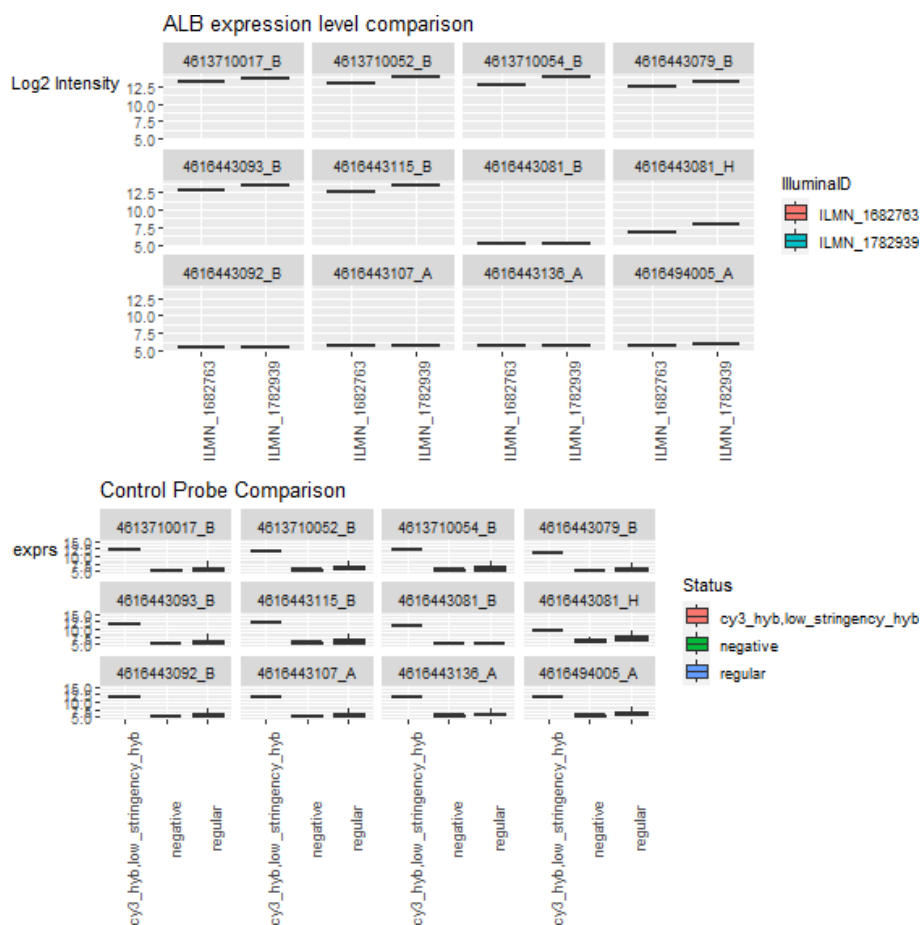
```
library(ggplot2)
library(gridExtra)
bp1 <- boxplot(exampleSummaryData.log2[ids, ],
               SampleGroup = "SampleFac", probeFactor = "IlluminaID")
```

## Analysis of Bead-summary Data using beadarray

```
bp1 <- bp1+ labs(title = "ALB expression level comparison") + xlab("Illumina Probe") + ylab("Log2 Intensity")

bp2 <- boxplot(exampleSummaryData.log2[randIDs,], probeFactor = "Status")
bp2 <- bp2 + labs(title = "Control Probe Comparison")

grid.arrange(bp1,bp2)
```



We can also extract the data that was used to construct the plot.

```
bp1$data
```

##	Var1	Var2	value	SampleGroup	probeFactor
## 1	ILMN_1782939	4613710017_B	13.528212	UHRR	ILMN_1782939
## 2	ILMN_1682763	4613710017_B	13.264742	UHRR	ILMN_1682763
## 3	ILMN_1782939	4613710052_B	13.800577	UHRR	ILMN_1782939
## 4	ILMN_1682763	4613710052_B	12.947888	UHRR	ILMN_1682763
## 5	ILMN_1782939	4613710054_B	13.841128	UHRR	ILMN_1782939
## 6	ILMN_1682763	4613710054_B	12.641636	UHRR	ILMN_1682763

## Analysis of Bead-summary Data using beadarray

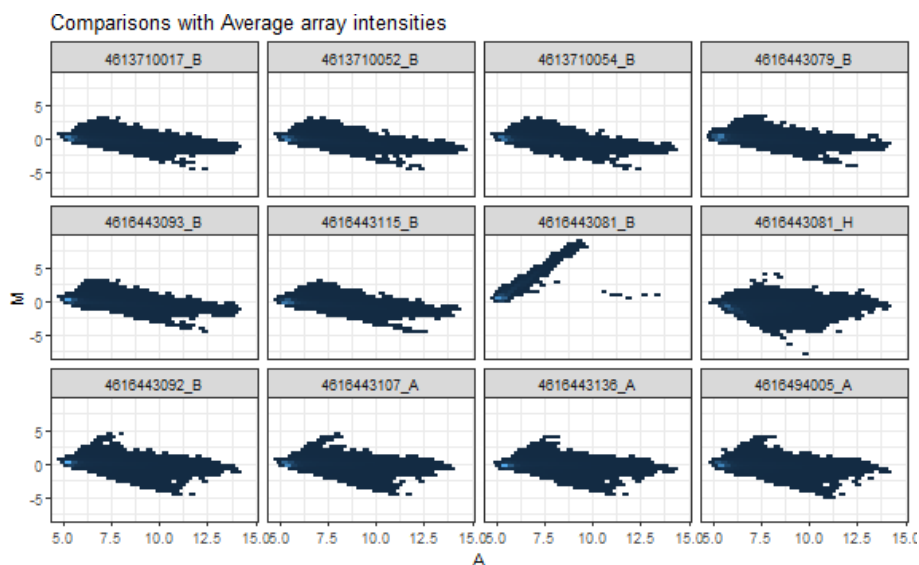
```
## 7  ILMN_1782939 4616443079_B 13.119897    UHRR ILMN_1782939
## 8  ILMN_1682763 4616443079_B 12.575922    UHRR ILMN_1682763
## 9  ILMN_1782939 4616443093_B 13.468822    UHRR ILMN_1782939
## 10 ILMN_1682763 4616443093_B 12.878392    UHRR ILMN_1682763
## 11 ILMN_1782939 4616443115_B 13.510831    UHRR ILMN_1782939
## 12 ILMN_1682763 4616443115_B 12.634381    UHRR ILMN_1682763
## 13 ILMN_1782939 4616443081_B  5.190355    Brain ILMN_1782939
## 14 ILMN_1682763 4616443081_B  5.249992    Brain ILMN_1682763
## 15 ILMN_1782939 4616443081_H  7.995407    Brain ILMN_1782939
## 16 ILMN_1682763 4616443081_H  6.788807    Brain ILMN_1682763
## 17 ILMN_1782939 4616443092_B  5.549147    Brain ILMN_1782939
## 18 ILMN_1682763 4616443092_B  5.388535    Brain ILMN_1682763
## 19 ILMN_1782939 4616443107_A  5.704762    Brain ILMN_1782939
## 20 ILMN_1682763 4616443107_A  5.617309    Brain ILMN_1682763
## 21 ILMN_1782939 4616443136_A  5.729863    Brain ILMN_1782939
## 22 ILMN_1682763 4616443136_A  5.658919    Brain ILMN_1682763
## 23 ILMN_1782939 4616494005_A  5.849509    Brain ILMN_1782939
## 24 ILMN_1682763 4616494005_A  5.598482    Brain ILMN_1682763
```

## 5 Other exploratory analysis

Replicate samples can also be compared using the `plotMA` function.

```
mas <- plotMA(exampleSummaryData.log2, do.log=FALSE)
```

```
mas
```

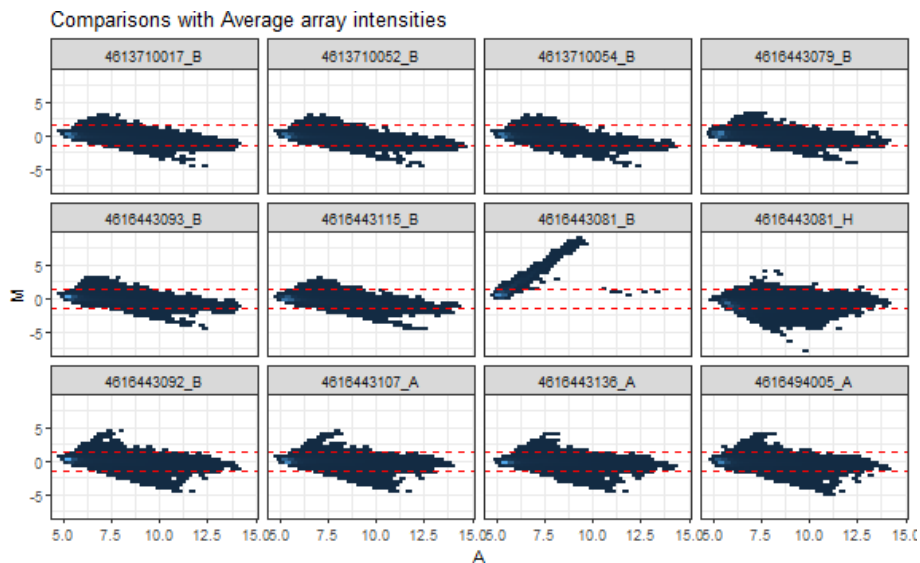


## Analysis of Bead-summary Data using beadarray

In each panel we see the MA plots for all arrays in the experiment compared to a 'reference' array composed of the average intensities of all probes. On an MA plot, for each probe we plot the average of the log<sub>2</sub> -intensities from the two arrays on the x-axis and the difference in intensities (log -ratios) on the y-axis. We would expect most probes to be and hence most points on the plot should lie along the line  $y=0$ .

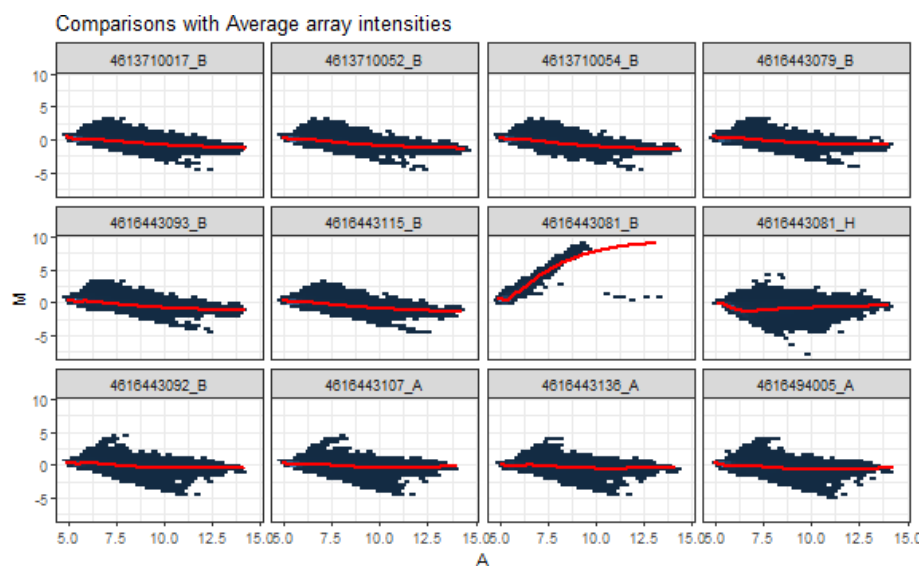
As with `boxplot`, the object returned is a *ggplot2* object that can be modified by the end-user.

```
##Added lines on the y axis  
mas + geom_hline(yintercept=c(-1.5,1.5),col="red",lty=2)
```



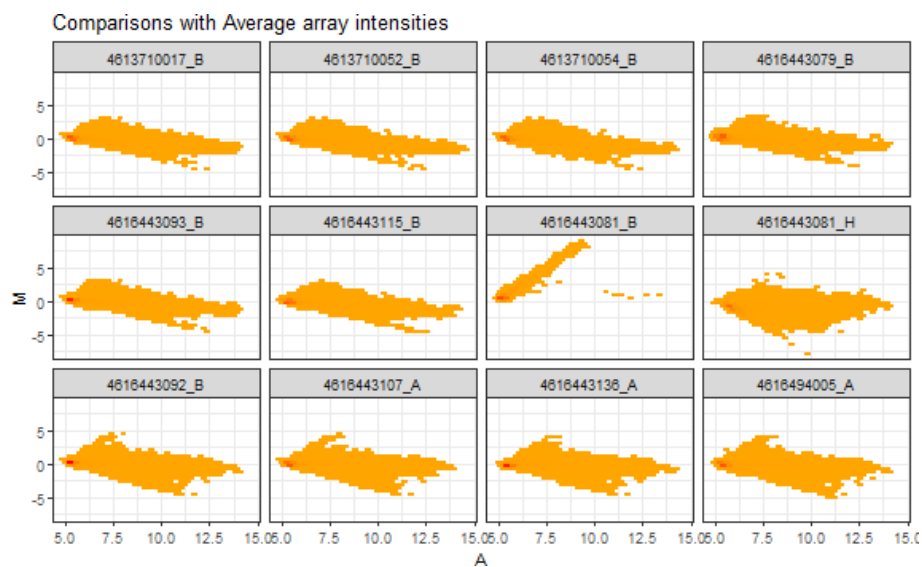
```
##Added a smoothed line to each plot  
mas+ geom_smooth(col="red")
```

## Analysis of Bead-summary Data using beadarray



*##Changing the color scale*

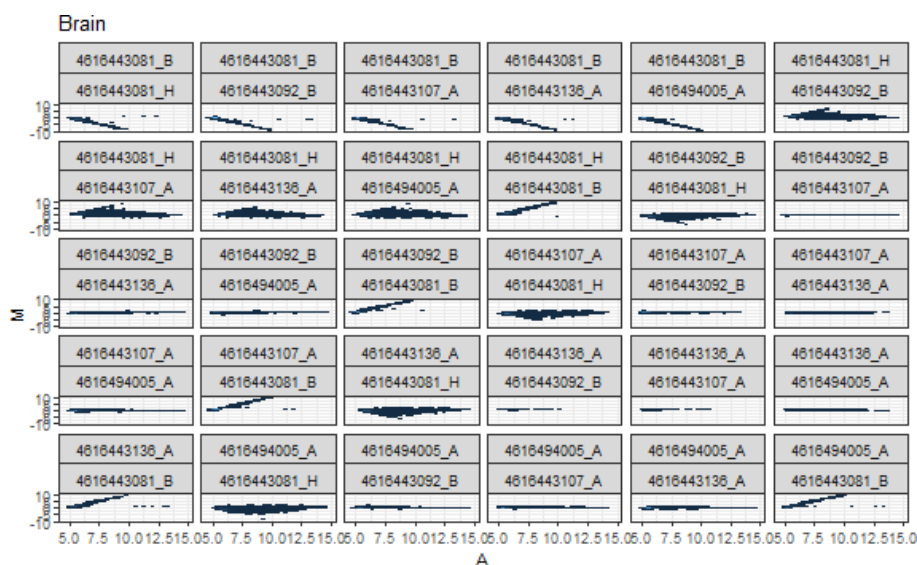
```
mas + scale_fill_gradient2(low="yellow",mid="orange",high="red")
```



We can also specify a sample grouping, which will make all pairwise comparisons

```
mas <- plotMA(exampleSummaryData.log2,do.log=FALSE,SampleGroup="SampleFac")
mas[[1]]
```

## Analysis of Bead-summary Data using beadarray



## 6 Normalisation

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.

```
exampleSummaryData.norm <- normaliseIllumina(exampleSummaryData.log2,  
method="quantile", transform="none")
```

An alternative approach is to combine normal-exponential background correction with quantile normalisation as suggested in the *limma* package. However, this requires data that have not been log-transformed. Note that the control probes are removed from the output object

```
exampleSummaryData.norm2 <- normaliseIllumina(channel(exampleSummaryData, "G.u1"),  
method="neqc", transform="none")
```

## 7 Filtering

Filtering non-responding probes from further analysis can improve the power to detect differential expression. One way of achieving this is to remove probes whose probe sequence has undesirable properties. Four basic annotation quality categories ('Perfect', 'Good', 'Bad' and 'No match') are defined and have been shown to correlate with expression level and measures of differential expression.

## Analysis of Bead-summary Data using beadarray

We recommend removing probes assigned a 'Bad' or 'No match' quality score after normalization. This approach is similar to the common practice of removing lowly-expressed probes, but with the additional benefit of discarding probes with a high expression level caused by non-specific hybridization.

```
library(illuminaHumanv3.db)

ids <- as.character(featureNames(exampleSummaryData.norm))

qual <- unlist(mget(ids, illuminaHumanv3PROBEQUALITY, ifnotfound=NA))

table(qual)

## qual
##      Bad      Good   Good***   Good****   No match   Perfect
##    13475     925     148       358     1739     24687
## Perfect*** Perfect****
##     6269     1975

rem <- qual == "No match" | qual == "Bad" | is.na(qual)

exampleSummaryData.filt <- exampleSummaryData.norm[!rem,]

dim(exampleSummaryData.filt)

## Features  Samples Channels
##    34362     12         1
```

## 8 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 UHRR and Brain replicates to give one value per condition. We then define contrasts comparing the Brain sample to the UHRR and calculate moderated t-statistics with empirical Bayes shrinkage of the sample variances. In this particular experiment, the Brain and UHRR samples are very different and we would expect to see many differentially expressed genes.

Empirical array quality weights can be used to measure the relative reliability of each array. A variance is estimated for each array by the `arrayWeights` function which measures how well the expression values from each array follow the linear

## Analysis of Bead-summary Data using beadarray

model. These variances are converted to relative weights which can then be used in the linear model to down-weight observations from less reliable arrays which improves power to detect differential expression. You should notice that some arrays have very low weight consistent with their poor QC.

We then define a contrast comparing UHRR to Brain Reference and calculate moderated  $t$ -statistics with empirical Bayes' shrinkage of the sample variances.

```
rna <- factor(pData(exampleSummaryData)[, "SampleFac"])

design <- model.matrix(~0+rna)
colnames(design) <- levels(rna)
aw <- arrayWeights(exprs(exampleSummaryData.filt), design)
aw
fit <- lmFit(exprs(exampleSummaryData.filt), design, weights=aw)
contrasts <- makeContrasts(UHRR-Brain, levels=design)
contr.fit <- eBayes(contrasts.fit(fit, contrasts))
topTable(contr.fit, coef=1)
```

### 8.1 Automating the DE analysis

A convenience function has been created to automate the differential expression analysis and repeat the above steps. The requirements to the function are a normalised object and a SampleGroup. By default, a design matrix and contrast matrix are derived from the SampleGroup by considering all pairwise contrasts. The matrices used, along with the array weights are saved in the output and can be retrieved later.

```
limmaRes <- limmaDE(exampleSummaryData.filt, SampleGroup="SampleFac")
limmaRes

## Results of limma analysis
## Design Matrix used...
##   Brain UHRR
## 1      0    1
## 2      0    1
## 3      0    1
## 4      0    1
## 5      0    1
## 6      0    1
## .....
##
## Array Weights....
```

## Analysis of Bead-summary Data using beadarray

```
## Contrast Matrix used...
##           Contrasts
## Levels   Brain-UHRR
##   Brain           1
##   UHRR            -1
## Array Weights....
## 2.098 2.544 ... 2.07 1.282
## Top Table
## Top 10 probes for contrast Brain-UHRR
##           Row.names ArrayAddressID   IlluminaID   Status SYMBOL
## ILMN_1651358 ILMN_1651358       4830541 ILMN_1651358 regular   HBE1
## ILMN_1796678 ILMN_1796678       450537 ILMN_1796678 regular   HBG1
## ILMN_1713458 ILMN_1713458       6980192 ILMN_1713458 regular   HBZ
## ILMN_1783832 ILMN_1783832       7570189 ILMN_1783832 regular   GAGE6
##           PROBEQUALITY      CODINGZONE
## ILMN_1651358      Perfect Transcriptomic
## ILMN_1796678      Perfect Transcriptomic
## ILMN_1713458      Perfect Transcriptomic
## ILMN_1783832      Good**** Transcriptomic
##
##                                     PROBESEQUENCE
## ILMN_1651358 ATTCTGGCTACTCACTTTGGCAAGGAGTTCACCCCTGAAGTGCAGGCTGC
## ILMN_1796678 AGAATTCACCCCTGAGGTGCAGGCTTCCTGGCAGAAGATGGTGAAGTGCAG
## ILMN_1713458 GTCCTGGAGGTTCCCCAGCCCCACTTACCGCGTAATGCGCCAATAAACCA
## ILMN_1783832 CCACAGACTGGGTGTGAGTGTGAAGATGGTCCTGATGGGCAGGAGGTGGA
##           GENOMICLOCATION      LogFC   LogOdds      pvalue
## ILMN_1651358 chr11:5289754:5289803:- -7.344650 67.23219 4.287612e-34
## ILMN_1796678 chr11:5269621:5269670:- -7.320087 66.69410 7.898248e-34
## ILMN_1713458 chr16:2044444:204493:+ -6.419908 64.52756 8.879586e-33
## ILMN_1783832 chrX:49330136:49330185:+ -5.973710 64.27316 1.175228e-32
##
##
## Significant probes with adjusted p-value < 0.05
## Direction
##   -1    0    1
## 4720 25619 4023

DesignMatrix(limmaRes)

##   Brain UHRR
## 1     0    1
## 2     0    1
## 3     0    1
## 4     0    1
```

## Analysis of Bead-summary Data using beadarray

```
## 5      0      1
## 6      0      1
## 7      1      0
## 8      1      0
## 9      1      0
## 10     1      0
## 11     1      0
## 12     1      0
## attr("assign")
## [1] 1 1
## attr("contrasts")
## attr("contrasts")$`as.factor(SampleGroup)`
## [1] "contr.treatment"

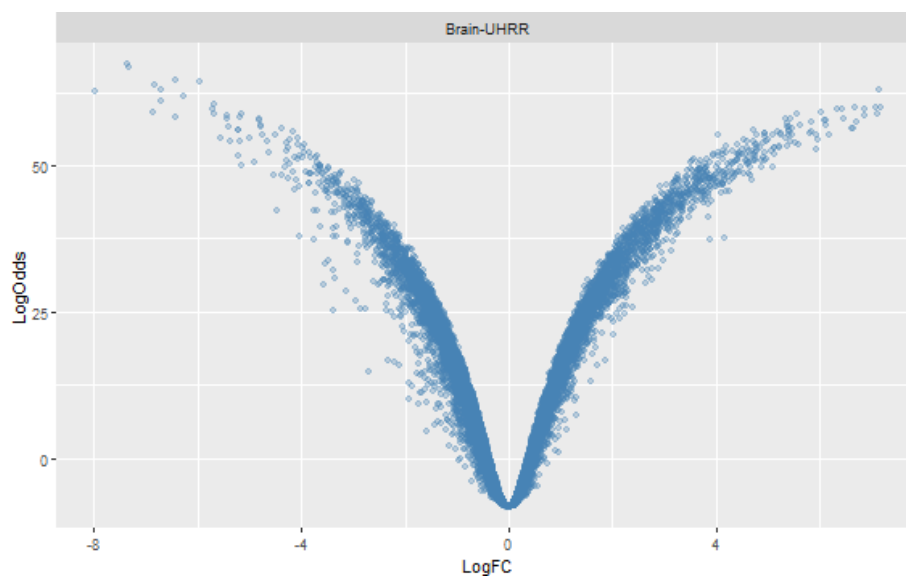
ContrastMatrix(limmaRes)

##           Contrasts
## Levels Brain-UHRR
## Brain      1
## UHRR      -1

ArrayWeights(limmaRes)

##           1           2           3           4           5           6           7
## 2.09833261 2.54416003 1.46675905 1.76081750 2.15657466 1.85199681 0.01204889
##           8           9          10          11          12
## 0.11056049 2.50695037 2.04941743 2.06972327 1.28194050

plot(limmaRes)
```



## 9 Output as GRanges

To assist with meta-analysis, and integration with other genomic data-types, it is possible to export the normalised values as a *GRanges* object. Therefore it is easy to perform overlaps, counts etc with other data using the [GenomicRanges](#) and [GenomicFeatures](#) packages. In order for the ranges to be constructed, the genomic locations of each probe have to be obtained from the appropriate annotation package (*illuminaHumanv3.db* in this example). Provided that this package has been installed, the mapping should occur automatically. The expression values are stored in the *GRanges* object along with the `featureData`.

```
gr <- as(exampleSummaryData.filt[,1:5], "GRanges")
gr
```

```
## GRanges object with 37013 ranges and 14 metadata columns:
```

##		seqnames	ranges	strand	Row.names	
##		<Rle>	<IRanges>	<Rle>	<AsIs>	
##	ILMN_1776601	chr1	69476-69525	+	ILMN_1776601	
##	ILMN_1665540	chr1	324468-324517	+	ILMN_1665540	
##	ILMN_1776483	chr1	324469-324518	+	ILMN_1776483	
##	ILMN_1682912	chr1	324673-324722	+	ILMN_1682912	
##	ILMN_1889155	chr1	759949-759998	+	ILMN_1889155	
##	...	...	...	...	...	
##	ILMN_1691189	chrUn_gl000211	23460-23509	-	ILMN_1691189	
##	ILMN_1722620	chr4_gl000193_random	75089-75138	-	ILMN_1722620	
##	ILMN_1821517	chrM	8249-8287	+	ILMN_1821517	
##	ILMN_1660133	chr7_gl000195_random	165531-165580	+	ILMN_1660133	
##	ILMN_1684166	chr4_gl000194_random	55310-55359	-	ILMN_1684166	
##		ArrayAddressID	IlluminaID	Status	SYMBOL	PROBEQUALITY
##		<numeric>	<factor>	<factor>	<character>	<character>
##	ILMN_1776601	3610128	ILMN_1776601	regular	OR4F5	Perfect
##	ILMN_1665540	2570482	ILMN_1665540	regular	<NA>	Perfect****
##	ILMN_1776483	6290672	ILMN_1776483	regular	<NA>	Perfect****
##	ILMN_1682912	4060014	ILMN_1682912	regular	<NA>	Perfect****
##	ILMN_1889155	1780768	ILMN_1889155	regular	<NA>	Perfect***
##	...	...	...	...	...	...
##	ILMN_1691189	5310750	ILMN_1691189	regular	<NA>	Perfect****
##	ILMN_1722620	5560181	ILMN_1722620	regular	LINC01667	Perfect****
##	ILMN_1821517	6550386	ILMN_1821517	regular	<NA>	Good
##	ILMN_1660133	7510136	ILMN_1660133	regular	<NA>	Perfect***
##	ILMN_1684166	7650241	ILMN_1684166	regular	MAFIP	Perfect
##		CODINGZONE	PROBESEQUENCE	GENOMICLOCATION		
##		<character>	<character>			<character>

## Analysis of Bead-summary Data using beadarray

```
## ILMN_1776601 Transcriptomic TGTGTGGCAACGCATGTGTC.. chr1:69476:69525:+
## ILMN_1665540 Transcriptomic CAGAACTTTCTCCAGTCAGC.. chr1:324468:324517:+
## ILMN_1776483 Transcriptomic AGAACTTTCTCCAGTCAGCC.. chr1:324469:324518:+
## ILMN_1682912 Transcriptomic GTCGACCTCACCAGGCCAG.. chr1:324673:324722:+
## ILMN_1889155 Transcriptomic? GCCCCAAGTGGAGGAACCT.. chr1:759949:759998:+
## ... ..
## ILMN_1691189 Transcriptomic GCCTGTCTTCAAACTAAGA.. chrUn_gl000211:23460..
## ILMN_1722620 Transcriptomic CCAGCATCTCCTGGACAGTC.. chr4_gl000193_random..
## ILMN_1821517 Transcriptomic GCAGGGCCCGTATTTACCCT.. chrM:8249:8287:+
## ILMN_1660133 Transcriptomic? GCAGACAGCCTGAGGAAGAT.. chr7_gl000195_random..
## ILMN_1684166 Transcriptomic TTTCTCCTCTGTCCCACTTA.. chr4_gl000194_random..
## X4613710017_B X4613710052_B X4613710054_B X4616443079_B
## <numeric> <numeric> <numeric> <numeric>
## ILMN_1776601 5.65762 5.32280 5.42786 5.23788
## ILMN_1665540 6.41036 6.19793 6.27768 6.22283
## ILMN_1776483 6.65937 6.23047 6.31748 6.08682
## ILMN_1682912 6.05543 6.03521 5.99879 6.01619
## ILMN_1889155 5.51711 5.53557 5.51818 5.47225
## ... ..
## ILMN_1691189 5.26922 5.11971 5.30667 5.33993
## ILMN_1722620 5.86809 5.88124 5.69247 5.74034
## ILMN_1821517 13.23931 13.53013 13.60952 13.09584
## ILMN_1660133 5.75937 5.93725 5.89674 5.52979
## ILMN_1684166 5.44291 5.34176 5.25382 5.78671
## X4616443093_B
## <numeric>
## ILMN_1776601 5.36293
## ILMN_1665540 6.30330
## ILMN_1776483 6.26472
## ILMN_1682912 6.01619
## ILMN_1889155 5.61177
## ... ..
## ILMN_1691189 5.46230
## ILMN_1722620 5.95092
## ILMN_1821517 13.19210
## ILMN_1660133 5.91776
## ILMN_1684166 5.71984
## -----
## seqinfo: 48 sequences from an unspecified genome; no seqlengths
```

## Analysis of Bead-summary Data using beadarray

The limma analysis results can also be exported as a GRanges object for downstream analysis. The elementMetadata of the output object is set to the statistics from the limma analysis.

```
lgr <- as(limmaRes, "GRanges")
lgr

## GRangesList object of length 1:
## $`Brain-UHRR`
## GRanges object with 37013 ranges and 3 metadata columns:
##           seqnames      ranges strand |      LogFC
##           <Rle>      <IRanges> <Rle> | <numeric>
## ILMN_1776601      chr1    69476-69525   + | -0.0429096
## ILMN_1665540      chr1  324468-324517   + | -0.3102748
## ILMN_1776483      chr1  324469-324518   + | -0.5445102
## ILMN_1682912      chr1  324673-324722   + | -0.2790607
## ILMN_1889155      chr1  759949-759998   + |  0.0095533
##           ...           ...           ...   ... .
## ILMN_1691189      chrUn_gl000211  23460-23509   - |  0.1446237
## ILMN_1722620 chr4_gl000193_random  75089-75138   - | -0.4055233
## ILMN_1821517           chrM      8249-8287   + |  0.0149122
## ILMN_1660133 chr7_gl000195_random 165531-165580   + | -0.3728807
## ILMN_1684166 chr4_gl000194_random  55310-55359   - |  0.0003597
##           LogOdds      PValue
##           <numeric> <numeric>
## ILMN_1776601  -8.17005 6.34482e-01
## ILMN_1665540  -3.16422 1.83906e-03
## ILMN_1776483   2.91450 4.06969e-06
## ILMN_1682912  -3.04031 1.61866e-03
## ILMN_1889155  -8.28264 9.09142e-01
##           ...           ...           ...
## ILMN_1691189  -7.682845 2.83997e-01
## ILMN_1722620   0.282373 5.58833e-05
## ILMN_1821517  -8.276389 8.74748e-01
## ILMN_1660133  -1.675010 4.01359e-04
## ILMN_1684166  -8.289516 9.97108e-01
## -----
## seqinfo: 48 sequences from an unspecified genome; no seqlengths
```

The data can be manipulated according to the DE stats

```
lgr <- lgr[[1]]
lgr[order(lgr$LogOdds,decreasing=T)]
```

## Analysis of Bead-summary Data using beadarray

```
## GRanges object with 37013 ranges and 3 metadata columns:
##           seqnames           ranges strand |           LogFC      LogOdds
##           <Rle>           <IRanges> <Rle> | <numeric> <numeric>
## ILMN_1651358 chr11 5289754-5289803   - | -7.34465 67.2322
## ILMN_1796678 chr11 5269621-5269670   - | -7.32009 66.6941
## ILMN_1713458 chr16 204444-204493     + | -6.41991 64.5276
## ILMN_1783832 chrX 49330136-49330185   + | -5.97371 64.2732
## ILMN_1782939 chr4 74285311-74285356   + | -6.82192 63.9110
##           ...           ...           ... .           ...           ...
## ILMN_1700728 chr2 27666372-27666399   + | 1.76813e-05 -8.28952
## ILMN_1700728 chr2 27666816-27666835   + | 1.76813e-05 -8.28952
## ILMN_1774781 chr19 45015129-45015178   - | -1.25897e-05 -8.28952
## ILMN_2152095 chr5 31401519-31401568   - | -6.37711e-06 -8.28952
## ILMN_1654074 chr17 48165652-48165701   + | 4.70621e-08 -8.28952
##           PValue
##           <numeric>
## ILMN_1651358 4.28761e-34
## ILMN_1796678 7.89825e-34
## ILMN_1713458 8.87959e-33
## ILMN_1783832 1.17523e-32
## ILMN_1782939 1.74926e-32
##           ...           ...
## ILMN_1700728 0.999847
## ILMN_1700728 0.999847
## ILMN_1774781 0.999883
## ILMN_2152095 0.999957
## ILMN_1654074 1.000000
## -----
## seqinfo: 48 sequences from an unspecified genome; no seqlengths

lgr[p.adjust(lgr$PValue)<0.05]

## GRanges object with 9452 ranges and 3 metadata columns:
##           seqnames           ranges strand |           LogFC      LogOdds
##           <Rle>           <IRanges> <Rle> | <numeric> <numeric>
## ILMN_1709067 chr1 879456-879505     + | -0.627372 7.43633
## ILMN_1705602 chr1 900738-900787     + | -0.611749 5.61562
## ILMN_1770454 chr1 991196-991245     + | -1.014300 18.40797
## ILMN_1780315 chr1 1246734-1246783   + | -0.720793 8.34688
## ILMN_1773026 chr1 1372636-1372685   + | 0.669350 10.36332
##           ...           ...           ... .           ...           ...
## ILMN_2398587 chr6_apd_hap1 1269579-1269628   + | -1.25312 19.7559
## ILMN_1692486 chr6_apd_hap1 1270027-1270031   + | -1.58605 26.9093
```

## Analysis of Bead-summary Data using beadarray

```
##      ILMN_1692486 chr6_apd_hap1 1270238-1270282      + | -1.58605 26.9093
##      ILMN_1708006 chr6_apd_hap1 2793475-2793524      + | -2.07723 33.1812
##      ILMN_2070300 chr6_apd_hap1 3079946-3079995      - | -1.43871 25.3520
##                                     PValue
##                                     <numeric>
##      ILMN_1709067 4.73562e-08
##      ILMN_1705602 2.83116e-07
##      ILMN_1770454 1.06095e-12
##      ILMN_1780315 1.94013e-08
##      ILMN_1773026 2.69858e-09
##      ...
##      ILMN_2398587 2.85759e-13
##      ILMN_1692486 2.70501e-16
##      ILMN_1692486 2.70501e-16
##      ILMN_1708006 5.95089e-19
##      ILMN_2070300 1.23259e-15
##      -----
##      seqinfo: 48 sequences from an unspecified genome; no seqlengths
```

We can do overlaps with other *GRanges* objects

```
library(GenomicRanges)
HBE1 <- GRanges("chr11", IRanges(5289580,5291373),strand="-")

lgr[lgr %over% HBE1]

## GRanges object with 1 range and 3 metadata columns:
##           seqnames      ranges strand |      LogFC  LogOdds
##           <Rle>        <IRanges> <Rle> | <numeric> <numeric>
##      ILMN_1651358 chr11 5289754-5289803 - | -7.34465 67.2322
##           PValue
##           <numeric>
##      ILMN_1651358 4.28761e-34
##      -----
##      seqinfo: 48 sequences from an unspecified genome; no seqlengths
```

## 9.1 Visualisation options

Having converted the DE results into a common format such as *GRanges* allows access to common routines, such as those provided by *ggbio*. For example, it is often useful to know where exactly the illumina probes are located with respect to the gene.

## Analysis of Bead-summary Data using beadarray

```
library(ggbio)
library(TxDb.Hsapiens.UCSC.hg19.knownGene)
tx <- TxDb.Hsapiens.UCSC.hg19.knownGene
p1 <- autoplot(tx, which=HBE1)
p2 <- autoplot(lgr[lgr %over% HBE1])
tracks(p1,p2)
id <- plotIdeogram(genome="hg19", subchr="chr11")
tracks(id,p1,p2)
```

Genome-wide plots are also available

```
plotGrandLinear(lgr, aes(y = LogFC))
```

## 10 Creating a GEO submission file

Most journals are now requiring that data are deposited in a public repository prior to publication of a manuscript. Formatting the microarray and associated metadata can be time-consuming, so we have provided a function to create a template for a GEO submission. GEO require particular meta data to be recorded regarding the experimental protocols. The output of the `makeGEOSubmissionFiles` includes a spreadsheet with the relevant fields that can be filled in manually. The normalised and raw data are written to tab-delimited files. By default, the annotation package associated with the data is consulted to determine which probes are exported. Any probes that are present in the data, but not in the annotation package are excluded from the submission file.

```
rawdata <- channel(exampleSummaryData, "G")
normdata <- normaliseIllumina(rawdata)

makeGEOSubmissionFiles(normdata, rawdata)
```

Alternatively, GEO's official probe annotation files can be used to decide which probes to include in the submission. You will first have to download the appropriate file from the GEO website.

```
download.file(
  "ftp://ftp.ncbi.nlm.nih.gov/geo/platforms/GPL6nnn/GPL6947/annot/GPL6947.annot.gz",
  destfile="GPL6947.annot.gz"
)

makeGEOSubmissionFiles(normdata, rawdata, softTemplate="GPL6947.annot.gz")
```

# 11 Analysing data from GEO

beadarray now contains functionality that can assist in the analysis of data available in the GEO (Gene Expression Omnibus) repository. We can download such data using *GEOquery*:

```
library(GEOquery)
url <- "ftp://ftp.ncbi.nih.gov/pub/geo/DATA/SeriesMatrix/GSE33126/"
filenm <- "GSE33126_series_matrix.txt.gz"
if(!file.exists("GSE33126_series_matrix.txt.gz")) download.file(paste(url, filenm, sep=""))
gse <- getGEO(filename=filenm)
head(exprs(gse))
```

Now we convert this to an *ExpressionSetIllumina*; *beadarray*'s native class for dealing with summarised data. The *annotation* slot stored in the *ExpressionSet* is converted from a GEO identifier (e.g. GPL10558) to one recognised by *beadarray* (e.g. Humanv4). If no conversion is possible, the resulting object will have `NULL` for the annotation slot. If successful, you should notice that the object is automatically annotated against the latest available annotation package.

```
summaryData <- as(gse, "ExpressionSetIllumina")
summaryData
head(fData(summaryData))
```

As we have annotated using the latest packages, we have imported the probe quality scores. We can calculate Detection scores by using the 'No match' probes as a reference; useful as data in repositories rarely export these data

```
fData(summaryData)$Status <-
  ifelse(fData(summaryData)$PROBEQUALITY=="No match","negative","regular" )

Detection(summaryData) <- calculateDetection(summaryData,
  status=fData(summaryData)$Status)
```

The 'neqc' normalisation method from limma can also be used now.

```
summaryData.norm <- normaliseIllumina(summaryData,method="neqc",
  status=fData(summaryData)$Status)
boxplot(summaryData.norm)
```

We can do differential expression if we know the column in the `phenoData` that contains sample group information

```
limmaResults <- limmaDE(summaryData.norm, "source_name_ch1")
limmaResults
```

## 12 Reading bead summary data into beadarray

---

BeadStudio/GenomeStudio is Illumina's proprietary software for analyzing data output by the scanning system (BeadScan/iScan). It contains different modules for analyzing data from different platforms. For further information on the software and how to export summarized data, refer to the user's manual. In this section we consider how to read in and analyze output from the gene expression module of BeadStudio/GenomeStudio.

The example dataset used in this section consists of an experiment with one Human WG-6 version 2 BeadChip. These arrays were hybridized with the control RNA samples used in the MAQC project (3 replicates of UHRR and 3 replicates of Brain Reference RNA).

The non-normalized data for regular and control probes was output by BeadStudio/GenomeStudio.

The example BeadStudio output used in this section is available as a zip file that can be downloaded from [http://compbio.works/data/BeadStudioExample/AsuragenMAQC\\_BeadStudioOutput.zip](http://compbio.works/data/BeadStudioExample/AsuragenMAQC_BeadStudioOutput.zip).

You will need to download and unzip the contents of this file to the current R working directory. Inside this zip file you will find several files including summarized, non-normalized data and a file containing control information. We give a more detailed description of each of the particular files we will make use of below.

- Sample probe profile (AsuragenMAQC-probe-raw.txt) (*required*) - text file which contains the non-normalized summary values as output by BeadStudio. Inside the file is a data matrix with some 48,000 rows. In newer versions of the software, these data are preceded by several lines of header information. Each row is a different probe in the experiment and the columns give different measurements for the gene. For each array, we record the summarized expression level (AVG\_Signal), standard error of the bead replicates (BEAD\_STDERR), number of beads (Avg\_NBEADS) and a detection  $p$ -value (Detection Pval) which estimates the probability

## Analysis of Bead-summary Data using beadarray

of a gene being detected above the background level. When exporting this file from BeadStudio, the user is able to choose which columns to export.

- Control probe profile (AsuragenMAQC-controls.txt) (*recommended*) - text file which contains the summarized data for each of the controls on each array, which may be useful for diagnostic and calibration purposes. Refer to the Illumina documentation for information on what each control measures.
- targets file (*optional*) - text file created by the user specifying which sample is hybridized to each array. No such file is provided for this dataset, however we can extract sample annotation information from the column headings in the sample probe profile.

Files with normalized intensities (those with avg in the name), as well as files with one intensity value per gene (files with gene in the name) instead of separate intensities for different probes targeting the same transcript, are also available in this download. We recommend users work with the non-normalized probe-specific data in their analysis where possible. Illumina's background correction step, which subtracts the intensities of the negative control probes from the intensities of the regular probes, should also be avoided.

```
library(beadarray)
dataFile = "AsuragenMAQC-probe-raw.txt"
qcFile = "AsuragenMAQC-controls.txt"
BSDData = readBeadSummaryData(dataFile = dataFile,
qcFile = qcFile, controlID = "ProbeID",
skip = 0, qc.skip = 0, qc.columns = list(exprs = "AVG_Signal",
Detection = "Detection Pval"))
```

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

## Analysis of Bead-summary Data using beadarray

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.

### 12.1 Reading IDAT files

We can also read `BeadArray` data in the format produced directly by the scanner, the IDAT file. The example below uses the *GEOquery* to obtain the four IDAT files stored as supplementary information for GEO series GSE27073. In this case the stored files have been compressed using gzip and need to be decompressed before *beadarray* can read them. If you are using IDAT files as they come of the scanner this step will not be necessary.

```
library(beadarray)
library(GEOquery)
downloadDir <- tempdir()
getGEOSuppFiles("GSE27073", makeDirectory = FALSE, baseDir = downloadDir)
idatFiles <- list.files(path = downloadDir, pattern = ".idat.gz", full.names=TRUE)
sapply(idatFiles, gunzip)
idatFiles <- list.files(path = downloadDir, pattern = ".idat", full.names=TRUE)
BSDData <- readIdatFiles(idatFiles)
```

The output from `readIdatFiles()` is an object of class *ExpressionSetIllumina*, as described earlier.

## 13 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

## 14 Asking for help on beadarray

---

Wherever possible, questions about *beadarray* should be sent to the Bioconductor mailing list<sup>1</sup>. This way, all problems and solutions will be kept in a searchable archive. When posting to this mailing list, please first consult the

---

<sup>1</sup><http://www.bioconductor.org>

## Analysis of Bead-summary Data using beadarray

*posting guide*. In particular, state the version of *beadarray* and R that you are using<sup>2</sup>, and try to provide a reproducible example of your problem. This will help us to diagnose the problem.

This vignette was built with the following versions of R and

```
sessionInfo()

## R version 4.3.0 RC (2023-04-18 r84287 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows Server 2022 x64 (build 20348)
##
## Matrix products: default
##
## locale:
##  [1] LC_COLLATE=C
##  [2] LC_CTYPE=English_United States.utf8
##  [3] LC_MONETARY=English_United States.utf8
##  [4] LC_NUMERIC=C
##  [5] LC_TIME=English_United States.utf8
##
## time zone: America/New_York
## tzcode source: internal
##
## attached base packages:
## [1] stats4      stats      graphics  grDevices  utils      datasets  methods
## [8] base
##
## other attached packages:
##  [1] GenomicRanges_1.53.0      GenomeInfoDb_1.37.0
##  [3] gridExtra_2.3             ggplot2_3.4.2
##  [5] illuminaHumanv3.db_1.26.0 org.Hs.eg.db_3.17.0
##  [7] AnnotationDbi_1.63.0      IRanges_2.35.0
##  [9] S4Vectors_0.39.0         beadarrayExampleData_1.37.0
## [11] beadarray_2.51.0          hexbin_1.28.3
## [13] Biobase_2.61.0            BiocGenerics_0.47.0
## [15] knitr_1.42
##
## loaded via a namespace (and not attached):
##  [1] KEGGREST_1.41.0          gtable_0.3.3             xfun_0.39
##  [4] lattice_0.21-8           vctrs_0.6.2              tools_4.3.0
```

---

<sup>2</sup>This can be done by pasting the output of running the function `sessionInfo()`.

## Analysis of Bead-summary Data using beadarray

```
## [7] bitops_1.0-7      generics_0.1.3     tibble_3.2.1
## [10] fansi_1.0.4       RSQLite_2.3.1      highr_0.10
## [13] BeadDataPackR_1.53.0 blob_1.2.4         pkgconfig_2.0.3
## [16] Matrix_1.5-4      lifecycle_1.0.3    GenomeInfoDbData_1.2.10
## [19] farver_2.1.1      compiler_4.3.0     stringr_1.5.0
## [22] Biostrings_2.69.0 munsell_0.5.0      BiocStyle_2.29.0
## [25] htmltools_0.5.5   RCurl_1.98-1.12    yaml_2.3.7
## [28] pillar_1.9.0      crayon_1.5.2       openssl_2.0.6
## [31] cachem_1.0.7      limma_3.57.0       magick_2.7.4
## [34] nlme_3.1-162      tidyselect_1.2.0   digest_0.6.31
## [37] stringi_1.7.12    dplyr_1.1.2        reshape2_1.4.4
## [40] splines_4.3.0     labeling_0.4.2     fastmap_1.1.1
## [43] grid_4.3.0        colorspace_2.1-0   cli_3.6.1
## [46] magrittr_2.0.3    utf8_1.2.3         withr_2.5.0
## [49] scales_1.2.1      bit64_4.0.5        rmarkdown_2.21
## [52] XVector_0.41.0    httr_1.4.5         bit_4.0.5
## [55] askpass_1.1       png_0.1-8          memoise_2.0.1
## [58] evaluate_0.20     mgcv_1.8-42        rlang_1.1.0
## [61] illuminaio_0.43.0 Rcpp_1.0.10        glue_1.6.2
## [64] DBI_1.1.3         BiocManager_1.30.20 base64_2.0.1
## [67] R6_2.5.1          plyr_1.8.8         zlibbioc_1.47.0
```