Description of BeadExplorer

Gareth Elvidge

01 May 2006

1. Introduction

BeadExplorer if a R software package for quality control, exploration and normalisation of Illumina BeadChip expression data. The software is especially useful for lab-bench biologists due to its ease of use however the software provides easy data import/export analysis methods for more experienced bioinformaticians which can be used as a basis for more in-depth analyses.
The current standard BeadChip analysis workflow involves the importation of scanned .tiff image files into the Illumina BeadStudio software tool which displays basic quality control parameters and enables normalisation of the data. It is this summarised data that is used by BeadExplorer.
BeadExplorer uses the normalisation functions within the affy package and also requires the R2HTML and widgetTools packages. The affy and widgetTools packages are available from the bioconductor repository (www.bioconductor.org) and the R2HTML package is available from the R-project website (www.R-project.org). The most useful chapters for the inexperienced user are Chapter 3 (Quick start) and Chapter 5 (Additional data exploration methods).

1.1 Background and terminology

The Illumina BeadChip platform allows the simultaneous hybridisation of 6-16 samples (depending on the chip type) onto a single chip. Within this vignette we use ‘chip’ to describe the hybridisation and data of all samples and ‘array’ to describe a single sample within the entire chip.

2. R basics

Download the latest version of R from www.R-project.org. For windows users, download the pre-compiled binary version. Ensure that the required libraries (R2HTML, widgetTools and affy) have been installed. Windows users can download these automatically from within the R GUI. Set the appropriate repository (bioconductor or CRAN) by selecting ‘Select repositories…’ from the ‘Packages’ dropdown menu. The required packages can then be installed by selecting ‘Install Package(s)…’ from the

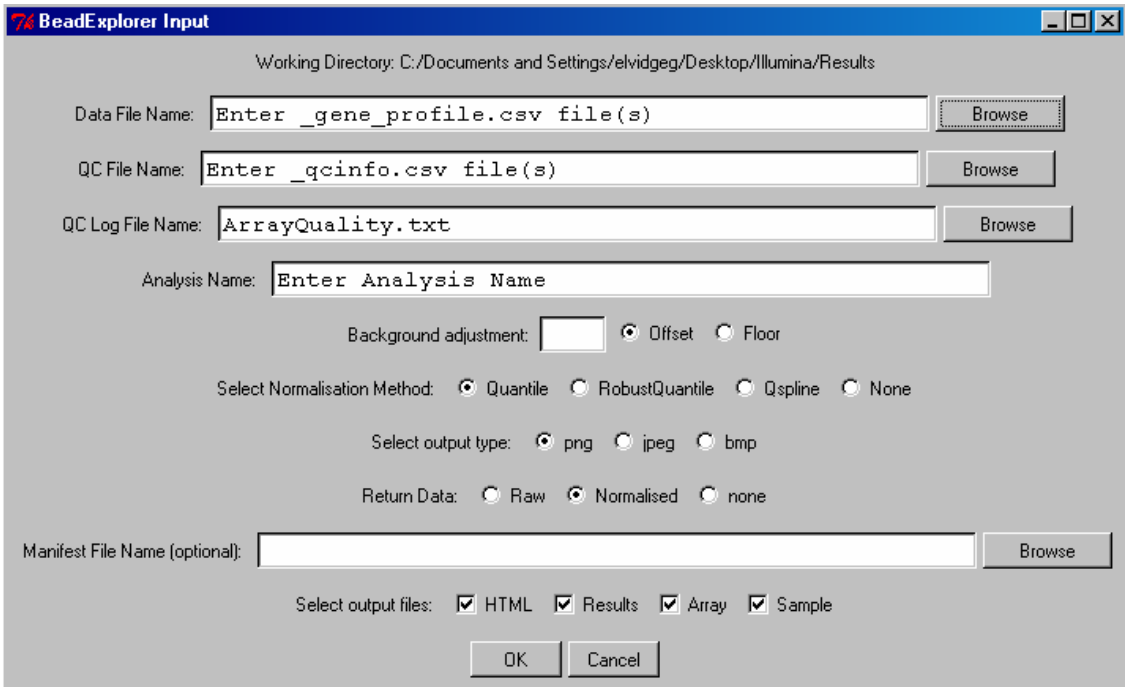
'Packages' drop down menu.
In addition to installing the required packages, we recommend creating a new folder where all the input files (and subsequent output files) will be stored. This folder then needs to be selected as the working directory by selecting the 'File' dropdown and 'Change dir…'.
In addition to this package vignette, further information on individual functions and general R usage can be gained by typing- *?function name* or selecting the help dropdown menu.

3. Quick start

BeadExplorer provides a wrapper function to automate standard analysis of BeadChip data. This can be used in conjunction with a GUI widget for easy data importation. Two BeadStudio output files are used as an input to BeadExplorer and these need to be placed in the working directory. The first contains the expression intensities of the BeadChip data and contains the suffix '_gene_profile'. The second contains quality control values that are based on the control sequences within the arrays. This file contains the suffix '_qcinfo'. Both files can be found within the BeadStudio output folder that are produced for each experimental analysis and are .csv files.

Launch the GUI by typing:

data<-beadAnalysis()



An example of the BeadExplorer input widget

A number of analysis parameters can be set from the GUI interface.

Data File Name:- Use 'Browse' to select the '_gene_profile.csv' file. Once this is selected

the QCfile name, QC Log file Name and Analysis Name are selected by default based on the name of the Data File. These however can be changed manually although this is not recommended for most purposes. Note that multiple files can be selected – in this case the data are combined into one analysis – useful if an experiment is spread over 2 or more BeadChips.

QC File Name:- The accompanying '_qcinfo.csv' file.

QC Log File Name:- A file where QC data from all previous BeadChip analyses is stored.

Analysis Name:- A unique identifier for the analysis. By default, the analysis name will be the same as the import file name, however, multiple analyses may be performed for a particular input file (e.g. using different normalisation and background correction options) and these can be saved separately by using unique analysis names.

Background adjustment:- see section 4.3 for further details. If offset is selected the default value is the absolute minimum value on the array. This can be changed by entering an alternative in the adjacent text box.

Select Normalisation Method:- see section 4.3 for further details.

Select output type:- format for graph outputs.

Return Data:- Select the type of BeadData object to return. For example, if no further analyses will be performed select 'None'. If further analyses will be performed (e.g. as described in Chapter 5) use the following command and select the option to return a normalised BeadData object:-

n<-beadAnalysis()

'n' will then hold a normalised beadData set and can be used in the further analyses.

Manifest File Name:- Optionally select an Illumina manifest (transcript annotation) .csv file to be included in the output data file. These can be found on the Illumina mapping CDs.

Select output files:- See below.

Output files

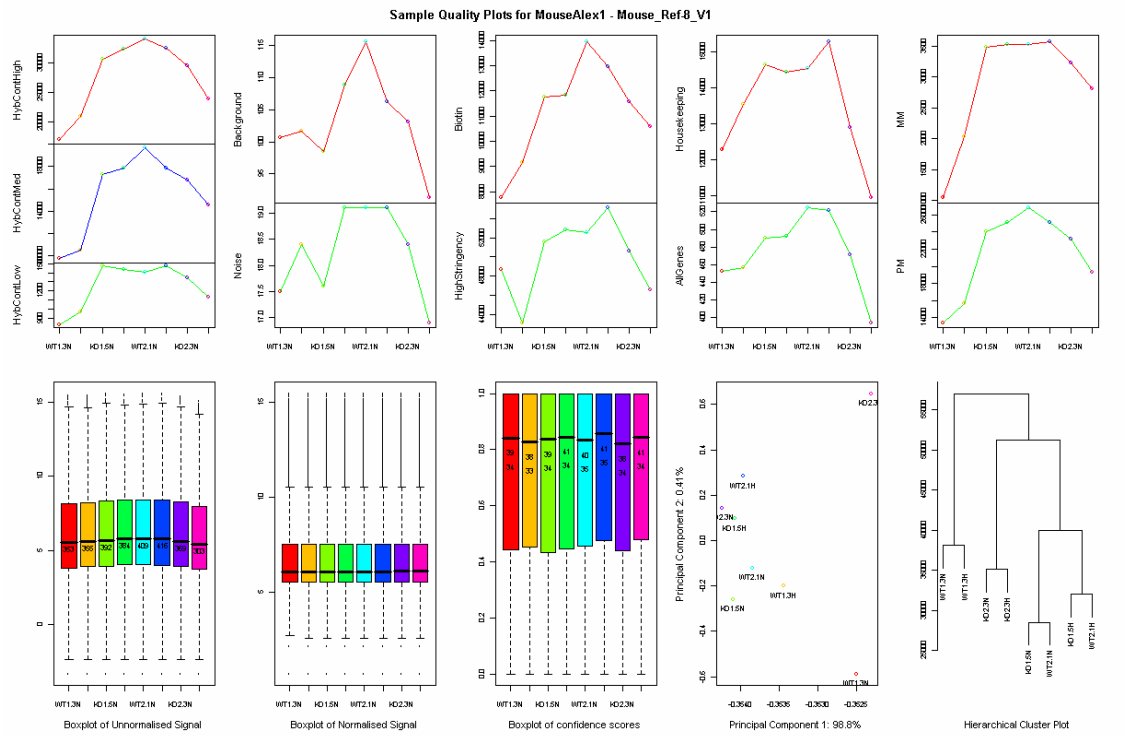
Four output files can be created in the working directory:

1. Normalised intensities

(with '_results' suffix) contains the normalised intensity values in tab delimited format.

2. Sample QC plots

A ‘jpeg, png or bmp’ image file with a ‘_samples’ suffix containing quality control information for each sample on the array thus providing a method for assessing intra-array variability. The top row of images show the quality control parameters (as determined in the BeadStudio software). The bottom row (from left to right) shows a boxplot of raw expression intensities, boxplot of normalised expression intensities, a principal component plot of the first and second principal components (together with the proportion of the total variation that is captured by each principal component) and a hierarchical clustering plot using Euclidean distance of the samples on the arrays.

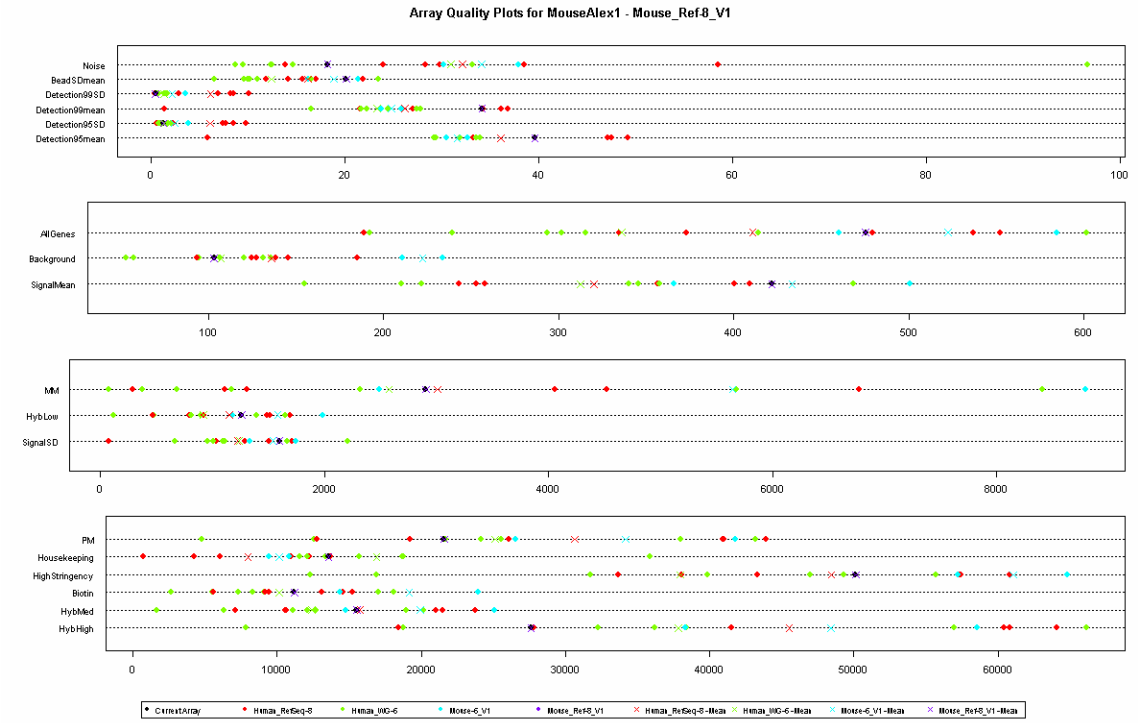


Example of a sample QC plot

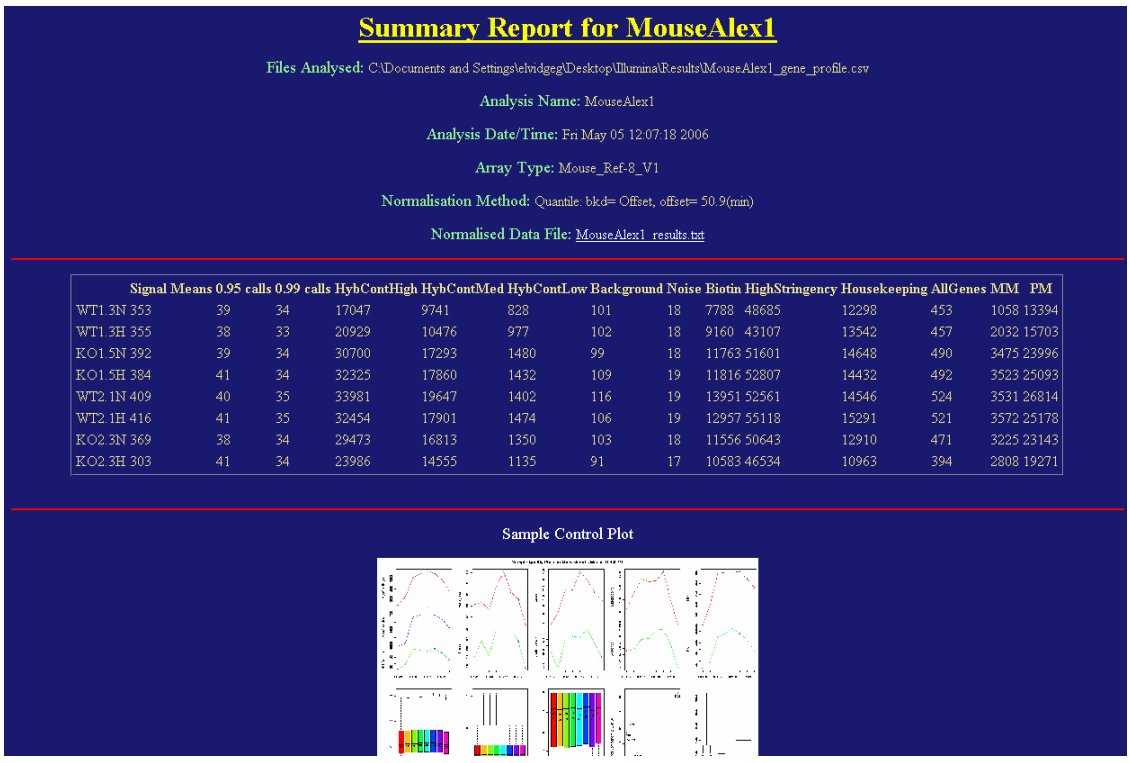
3. Array QC plot

A ‘jpeg, png or bmp’ image file with a ‘_arrays’ suffix containing quality control information for the entire chip. As each analysis is performed, quality control parameters for the entire chip are logged into a file in the working directory called ‘ArrayQuality.txt’. The images in the Array QC plot uses this file to compare the current analysis QC values against previous values and provides a method for assessing inter-array variability.

In addition, if the html output option is selected a html report file will be produced and displayed which show the array quality control values, analysis parameters and provide links to the files described above.



An example of an array QC plot



Summary Report for MouseAlex1

Files Analysed: C:\Documents and Settings\elvidgeg\Desktop\Illumina\Results\MouseAlex1_gene_profile.csv

Analysis Name: MouseAlex1

Analysis Date/Time: Fri May 05 12:07:18 2006

Array Type: Mouse_Ref-8_V1

Normalisation Method: Quantile: bkd= Offset, offset= 50.9(min)

Normalised Data File: MouseAlex1_results.txt

	Signal Means	0.95 calls	0.99 calls	HybContHigh	HybContMed	HybContLow	Background	Noise	Biotin	HighStringency	Housekeeping	AllGenes	MM	PM
WT1.3N	353	39	34	17047	9741	828	101	18	7788	48685	12298	453	1058	13394
WT1.3H	355	38	33	20929	10476	977	102	18	9160	43107	13542	457	2032	15703
KO1.5N	392	39	34	30700	17293	1480	99	18	11763	51601	14648	490	3475	23996
KO1.5H	384	41	34	32325	17860	1432	109	19	11816	52807	14432	492	3523	25093
WT2.1N	409	40	35	33981	19647	1402	116	19	13951	52561	14546	524	3531	26814
WT2.1H	416	41	35	32454	17901	1474	106	19	12957	55118	15291	521	3572	25178
KO2.3N	369	38	34	29473	16813	1350	103	18	11556	50643	12910	471	3225	23143
KO2.3H	303	41	34	23986	14555	1135	91	17	10583	46534	10963	394	2808	19271

Sample Control Plot

An example of part of the HTML summary report

4. Description of standard workflow

A typical workflow for an analysis in BeadExplorer is automated by the wrapper function described above in the quick start chapter. This provides a mechanism for the automated analysis of BeadChip data. The functions within the wrapper are now described in greater detail and can be performed separately using a command based input for greater flexibility and customisation:

4.1 Import BeadChip data file

data<-readBead("Example_gene_profile.csv")

The intensity data is read into a beadData object which inherits from an Exprs object (see chapter 7 for a description of classes used in BeadExplorer). Note that by passing multiple file names to this function, data from different chips can be combined into one analysis.

4.2 Import the QC data file

The default file to import is the same name as the data file expect that instead of a '_gene_profile.csv' suffix, the software looks for a '_qcinfo.csv suffix'.

qcdata<-getQC(data)

The QC data is read into a beadQC object (see chapter 7 for a description of classes used in BeadExplorer).

4.3 Normalise the raw data

A number of normalisation methods are provided that use algorithms within the affy package. Alternatively, if the normalisation has already been performed within BeadStudio and no further analysis is required then the method can be set to 'none'.
In addition to the normalisation options, adjustments are made to remove negative values. Negative intensities create problems for many downstream applications that require logged values or fold change calculations. They arise when the background intensity, based on the negative control values are subtracted from the raw intensities within BeadStudio. Two methods are available to remove negative values:

1. All measurements less than 0.1 are set to 0.1 (called 'Floor' in BeadExplorer).
2. A small constant, (equal to the lowest intensity on the chip by default) is added to each intensity value. This effect removes negative values and also reduces the variability of the data at low expression values (called 'Offset' in BeadExplorer).

normdata<-normalise (data)

4.4 Write normalised data

Normalised data is written to a tab delimited text results file within the working directory called analysisname_results.txt.

write.beadData(data)

4.5 Write the QC data.

The QC data for the analysis is written to the log file, 'ArrayQuality.txt'. If the log file is not found within the working directory then a new file is created. Only QC data with a unique analysis name will be written to the log file. If the analysis name is already found within the log file then no data is written.

writeQC(qcdata)

4.6 Read QC log file

The QC log file, containing QC data from previous analyses is loaded into a data frame. This data is required for plotting graphs to assess inter-chip variability.

qc<-readQC()

4.7 Generate sample (array) quality control plots

The sample quality control plots are plotted to assess sample-sample variation (or intra-chip variability). The allows the detection of problematic samples or hybridisations.

plotSamples(data, normdata, qcdata)

4.8 Generate chip quality control plots

The chip quality control plots enable the assessment of chip-chip variability (or inter-chip variability). This allows the detection of a problematic chip.

dotPlotArrays(qcdata,qc)

If a large number of chips are present within the log file that a dot plot can become cluttered. Another option is to plot the data as a boxplot instead:

boxPlotArrays(qcdata,qc)

4.9 Generate an HTML report

The generation of an HTML report can be useful to keep track of analysis parameters used

within a particular analysis. Links are also provided to all output files that are produced. We routinely use the file as a starting point for the exploration of all other output files.

htmlOutput (normdata, data, qcdata)

5. Additional data exploration methods

Most of the subsequent methods require a normalised beadData object (called 'data' in the examples below). This can be obtained by using the wrapper function:

data<-beadAnalysis()

(and selecting Normalised data as the return type)

5.1 Filter data

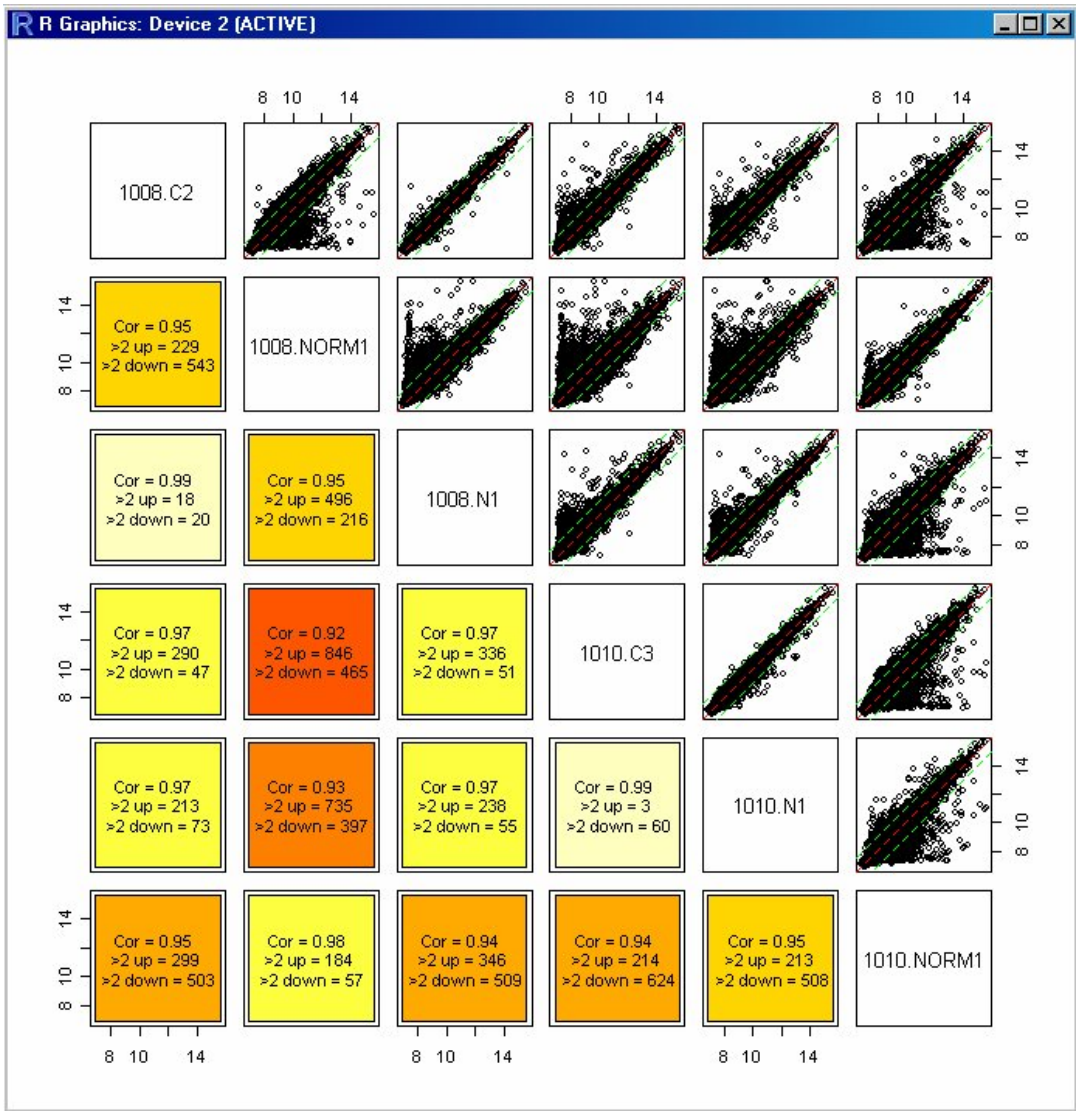
Removes data that are not expressed (using a given detection threshold) in any of the samples on the chip

data<-filter(data, 0.99)

5.2 Pair wise scatter plots

Explore the overall differences in expression between different samples on the chip. This can be useful to gain an overall impression of the variability in the data.

pairs(data)



An example of the output from pairs()

5.3 Identify up/down transcripts

Reports the pairwise number of transcripts up and down regulated using a predetermined fold cutoff.

cutoff (data,2)

5.4 Plot a two sample scatterplot comparison

Produces a scatterplot of two samples on the chip. Specific points (determined by a vector of transcript IDs) can be coloured differently or the plot can be coloured by detection value (see function description for more details). In addition, if identify is TRUE (default) then individual points on the graph can be clicked. When all points have been identified, values are printed to the console that identify each point and give fold change and intensity values. To include annotation data in the output to the console specify a manifest data.frame in the

input. This can be produced by *read Manifest*().

man<-readManifest("d:\\Bead_Set_Manifest\\Human_WG-6.csv")
scatterPlot(data, 1, 2, man)

5.5 Plot multiple density distributions of data
This is a similar function to boxplots expect that density distributions are plotted instead.

e.g. to plot a distribution of expression values:

multihists(exprs(data))

bead standard deviations:

multihists(se.exprs(data))

detection scores:

multihists(detExprs(data))

5.6 Plot individual QC plots
Many of the separate plots that are produced using *plotSamples()* and *dotplotArrays()* can be produced individually. These functions allow for greater customisation of plotting parameters (e.g. colours, plotted data) and also analysis parameters.
Functions that can be used include:-

pc()
boxplot()
dotplotQC()
boxplotQC()
clusterplot()

5.7 Further data exploration and manipulation

Obtain sample names: *sampleNames (data)*
Obtain summary of the data: *summary (data)*
Obtain the percentage of probes that pass a predefined detection cutoff:
detectionCalls (data,0.99)
Obtain the number of transcripts: *ngenes (data)*
Obtain analysis parameters: *param (data)*
Obtain QC parameters: *getArraysStats (data)*
Create a beadData object using a subset of the samples on the chip: *data<-data[,c(1,2,3)]*

6. Batch Analysis

Multiple files can be analysed in an automated fashion. Using the batchanalysis () function all files within the working directory that possess the suffix '_gene_profile.csv' will be analysed by the wrapper function, beadAnalysis(). Note that the corresponding '_qcinfo.csv' files will also be required within the working directory.

7. Description of classes

Two custom classes, 'beadData' and 'beadQC' are used within BeadExplorer to hold the expression data and qc data respectively. The classes are described below with more specific information within the corresponding html help file for the class.

7.1 BeadData

The BeadData class is an extension of the Exprs class and thus contains all the slots and methods of the Exprs class. Additional slots are :

fileOrigin- the filenames of the data used to create the object
detExprs- a matrix of detection scores for each transcript and sample
param – various analysis parameters
normmethod- the normalisation method used (if any)

The class holds the analysis name within the annotation slot, intensity data (raw or normalised) within the exprs slot, bead standard deviations (within the se.exprs slot) and detection scores (within the detExprs) slot.
Most of the slots can be accessed (and replaced) using the name of the slot e.g.

exprs(data)

- to access the intensity values

7.2 BeadQC

The beadQC class holds the quality control information for each chip, mainly determined from the corresponding '_qc_info.csv' file.
The slots are:

fileOrigin – as above
annotation – analysis name (as above)
sampleQC – various QC parameters for each sample on the chip.