beadarray: An R Package to Analyse Illumina BeadArrays

by Mark Dunning, Mike Smith*, Natalie Thorne, and Simon Tavaré*

Introduction

Illumina have created an alternative microarray technology based on randomly arranged beads, each of which carries copies of a gene-specific probe (Kuhn et al., 2004). Random sampling from an initial pool of beads produces an array containing, on average, 30 randomly positioned replicates of each probe type. A series of decoding hybridisations is used to identify each bead on an array (Gunderson et al., 2004). The high degree of replication makes the gene expression levels obtained using BeadArrays more robust. Spatial effects do not have such a detrimental effect as they do for conventional arrays that provide little or no replication of probes over an array. Illumina produce two distinct BeadArray technologies; the SAM (Sentrix Array Matrix) and the BeadChip. A SAM is a 96-well plate arrangement containing 96 uniquely prepared hexagonal BeadArrays, each containing around 1500 bead types. The BeadChip technology comprises a series of rectangular strips on a slide with each strip containing around 24,000 bead types. The most commonly used BeadChip is the HumanRef-6 BeadArray. These have 6 pairs of strips, each pair having 24,000 RefSeq genes and 24,000 supplemental genes.

Until now, analysis of BeadArray data has been carried out by using Illumina's own software package (BeadStudio) and therefore does not utilise the wide range of bioinformatic tools already available via Bioconductor (Gentleman et al., 2004), such as **limma** (Smyth, 2005) and **affy** (Gautier et al., 2004). Also, the data output from BeadStudio only gives a single average for each bead type on an array. Thus potentially interesting information about the replicates is lost. The intention of this project is to create an R package, **beadarray**, for the analysis of BeadArray data incorporating ideas from existing Bioconductor packages. We aim to provide a flexible and extendable means of analysing BeadArray data both for our own research purposes and for the benefit of other users of BeadArrays. The **beadarray** package is able to read the full bead level data for both SAM and BeadChip technologies as well as bead summary data processed by BeadStudio. The package includes an implementation of Illumina's algorithms for image processing, although local background correction and sharpening are optional.

At present, **beadarray** is a package for the analysis of Illumina expression data only. For the analysis of Illumina SNP data, see the **beadarraySNP** package in Bioconductor.

[1]These authors contributed equally to this work.

Data Formats

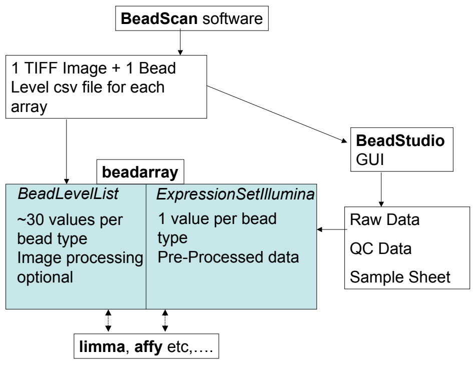

Figure 1: Diagram showing the interaction between **beadarray** and Illumina software. Various functions from Bioconductor packages (eg., **limma** and **affy**) may be used for the pre-processing or downstream analysis of `BeadLevelList` or `ExpressionSetIllumina` objects.

After hybridisation and washing, each SAM or BeadChip is scanned by the BeadScan software to produce a TIFF image for each array in an experiment. The size of each TIFF image is around 6Mb and 80Mb for SAM and BeadChip arrays respectively. BeadScan will also provide a text file describing the identity and position of each bead on the array if requested (version 3.1 and above of BeadScan only). We call this the bead level data for an array. Note that the bead level text file has to be produced when the arrays are scanned and cannot be generated later on. BeadStudio is able to read these raw data and produce a single intensity value for each bead type after outliers are excluded. These values are referred to as bead summary data. The images are processed using a local background correction and sharpening transformation (Dunning et al., 2005). These steps are not optional within the BeadStudio software. The bead summary values calculated by BeadStudio can be output directly with or without normalisation applied or used for further analysis within the application. The output from BeadStudio also contains information about the standard error of each bead type, the number of beads and a detection score that measures the probablity that the bead type is expressed; this uses a model assumed by Illumina. All analysis within BeadStudio is done on the unlogged scale and using the bead summary values rather than full replicate information for each bead type. Figure 1 gives an overview of the interaction between the file formats and software involved in the analysis of BeadArray data.

Bead Level Analysis

The `readBeadLevelData` function is used to read bead level data. A targets object is used in a similar way to **limma** to define the location of the TIFF images and text files. Both SAM and BeadChip data can be read by `readBeadLevelData`. The default options for the function use the sharpening transformation recommended by Illumina and calculate a local background value for each bead. The usage of `readBeadLevelData` is:

```
> BLData = readBeadLevelData(targets,
+ imageManipulation="sharpen")

> slotNames(BLData)

[1] "G"        "R"        "Rb"       "Gb"
[5] "GrnY"     "GrnX"     "ProbeID"  "targets"
```

Even though the size of the TIFF images is rather large, `readBeadLevelData` uses C code to read these images quickly, taking around 2 seconds to process each SAM array and 1 minute for each BeadChip array on a 3Ghz Pentium©IV machine with 3Gb of RAM. An object of type `BeadLevelList` is returned by `readBeadLevelData`. The design of this class is similar to the `RGList` object used in **limma**. Note that we store the foreground (`G`) and background (`Gb`) intensities of each bead separately so that local background correction is optional using the `backgroundCorrect` function. Each bead on the array has a ProbeID that defines the bead type for that bead. We order the rows in the matrix according to the ProbeID of beads, making searching for beads with a particular ProbeID more efficient. The `GrnX` and `GrnY` matrices give the coordinates of each bead centre on the original TIFF image. Note that the random nature of BeadArrays means that the number of replicates of a particular bead type will vary between arrays and rows in the matrices will not always correspond to the same bead type, unlike data objects for other microarray technologies.

Typical quality control steps for conventional microarrays involve looking for systematic differences between arrays in the same experiment and also for large spatial effects on each array. The `boxplot` function can be easily applied to the foreground or background values to reveal possible defective arrays. We also provide the function `imageplot` for investigating the variation in foreground and background intensities over an array. This function is different from the functionality available in **limma** because BeadArrays do not have print-tip groups as found on conventional spotted microarrays. Therefore we define a grid of M × N rectangles, average over the $\log_2$ bead intensities found inside each rectangle and call the `image` function. In Figure 3 we can clearly see that the top-right corner of an array has a marked difference in intensity compared to the rest of the array, where the variation in intensity appears to be random. Such a spatial effect might be a cause for concern in conventional microarrays where probes representing the same gene appear in the same location on all arrays. In BeadArrays however, the arrangement of beads is random and hence spatial trends tend to have less impact. Whilst BeadStudio provides functionality to view the TIFF images, the resolution of the images is too high to be able to identify overall trends across an array. Also, the intensity levels between arrays cannot be easily compared. The `displayTIFFImage` function in **beadarray** allows users to view sections of TIFF images in a similar manner to BeadStudio but with the advantage of being able to see which beads are outliers. See Figure 3 of Dunning et al. (2005).

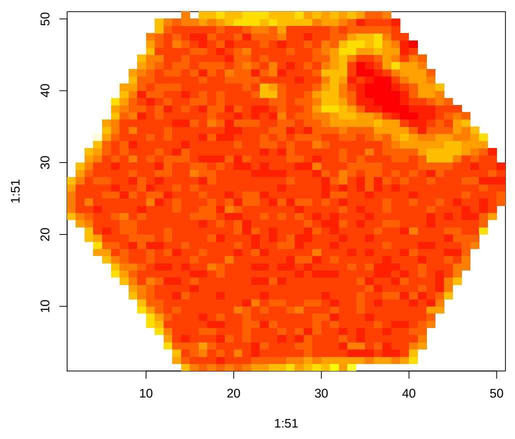

Figure 3: Plot showing the variation in $\log_2$ foreground intensity for a BeadArray from a SAM.

The random nature of BeadArrays and high replication rates allow for low level analysis that is not possible for other microarray technologies. For example, we can see where the replicates of a particular bead type are located on an array (`plotBeadLocations`) and view the intensities of the replicates (`plotBeadIntensities`). Due to the large number of bead types, in-depth analysis of each bead type is impractical. However, we might be interested in a subset of bead types, such as the controls used in the experiment. The `plotBeadIntensities` function produces a series of boxplots of $\log_2$ bead intensity for a supplied vector of ProbeIDs and can therefore

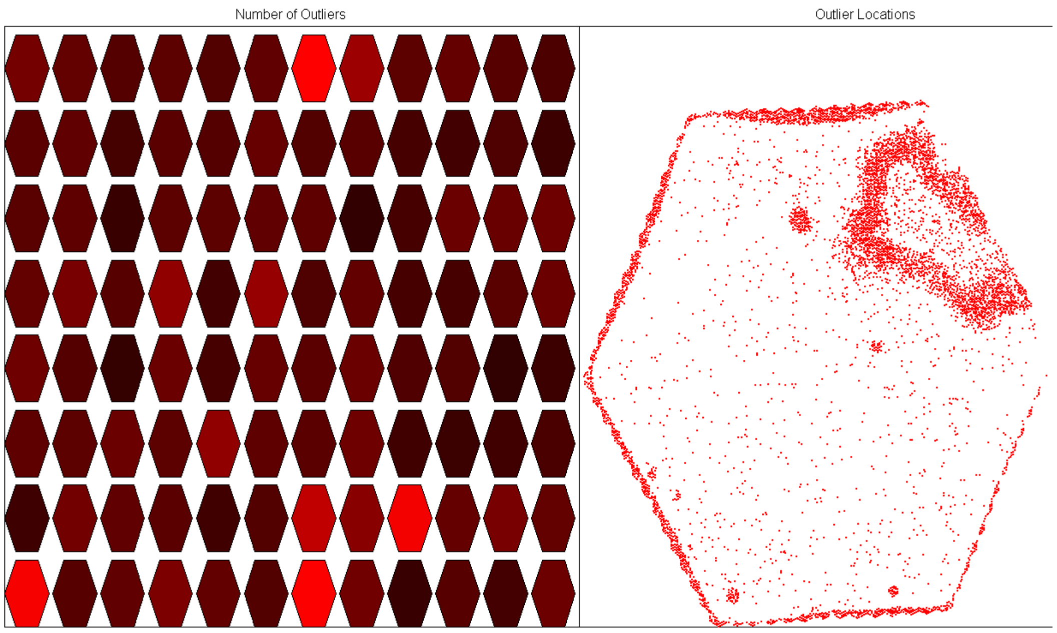

Figure 2: **beadarray** can be used to find spatial effects on arrays. On the left is a representation of the number of outliers for each array (bright red indicates more outliers) and on the right is the location of outliers for a particular array. Clicking on a hexagon on the left will change which array is displayed on the right. For this figure, the array in the 7th column of the first row of the SAM was chosen.

be a useful diagnostic tool.

Another set of beads that are of interest are outliers. Illumina exclude outliers for a particular bead type using a cut-off of 3 MADs from the unlogged mean. This analysis can be repeated for all bead types on an array using the `findAllOutliers` function. Users may specify a different cut-off as a multiple of the MAD or use the $\log_2$ intensity of beads in this function. Note that the outliers are not removed from the analysis at this point. To find all the outliers for every bead type on the first array we would use:

```
> o = findAllOutliers(BLData, array=1)
```

The result is a vector that indexes the rows of the first array in `BLData`. The `plotBeadLocations` function can then be used to plot the location of the outliers on the array.

```
> plotBeadLocations(BLData, array=1,
                    BeadIDs=o)
```

Typically, we find that 5% of beads on arrays are outliers and this can be used as an ad-hoc criterion for quality control. Figure 2 shows one of the interactive features available within **beadarray**. The left side of the screen gives an overview of all arrays on a SAM or BeadChip. In this example, each array is coloured according to the number of outliers and immediately we can see which arrays on the SAM have a larger number of outliers (shown in bright red). By clicking on a particular array in the graphic display, the location of all outliers on that array will be displayed on the right of the screen. This example shows the same array seen in Figure 3. As one might expect, many of the outliers in Figure 2 correspond to areas of higher foreground intensity visible in Figure 3.

Alternatively, the foreground or background intensities of arrays may be used to colour the arrays in the left screen and imageplots such as Figure 3 can be displayed when individual arrays are clicked. This interactive functionality is available for both SAM and BeadChip bead level data and can be started by the `SAMSummary` and `BeadChipSummary` functions respectively with the `BeadLevelList` created by `readBeadLevelData` passed as a parameter.

The `createBeadSummaryData` function creates bead summary data for each array by removing outliers for a particular bead type and then taking an average of the remaining beads. The `findAllOutliers` function is used by default and the result is an `ExpressionSetIllumina` object which can be analysed using different functionality within **beadarray**.

Reading Pre-Processed Bead Summary Data

Bead Summary data processed by BeadStudio can be read into **beadarray** using the function `readBeadSummaryData`. Three separate input files are required by the function, the location of which can be specified by a targets text file. The first two files are automatically created by BeadStudio by selecting the Gene Analysis option whereas the third file must be created by the user.

- Raw Data file - This contains the raw, non-normalised bead summary values as output by BeadStudio. Inside the file are several lines of header information followed by a data matrix. Each row is a different probe in the experiment and the columns give different measurements for the probes. For each array, we record the summarised expression level (AVG_Signal), standard error of the bead replicates (BEAD_ST_DEV), Number of beads used (Avg_NBEADS) and a Detection score which estimates the probability of a gene being detected above the background. Note that whilst this data has not been normalised, it has been subjected to local background correction at the bead level prior to summarising.
- QC Info - Gives the summarised expression values for each of the controls that Illumina place on arrays and hence is useful for diagnostic purposes. Each row in the file is a different array and the columns give average expression, standard error and detection for various controls on the array. See Illumina documentation for descriptions of control types.
- Sample Sheet - Gives a unique identifier to each array and defines which samples were hybridised to each array.

An example Bead Summary data set is included with the **beadarray** package and can be found as a zipped folder in the **beadarray** download. These data were obtained as part of a pilot study into BeadArray technology and comprises 3 HumanRef-6 BeadChips with various tumour and normal samples hybridised. The following code can be used to read the example data into R.

```
> targets <-
    readBeadSummaryTargets("targets.txt")
> BSData <- readBeadSummaryData(targets)

> BSData

Instance of ExpressionSetIllumina

assayData
  Storage mode: list
  Dimensions:
         BeadStDev Detection exprs NoBeads
Features     47293     47293 47293   47293
Samples         18        18    18      18

phenoData
  rowNames: I.1, IC.1,..., Norm.2, (18 total)
  varLabels and descriptions:

featureData
  featureNames: GI_10047089-S,...(47293 total)
  varLabels and descriptions:

Experiment data
  Experimenter name:
  Laboratory:
  Contact information:
  Title:
  URL:
  PMIDs:
  No abstract available.

Annotation [1] "Illumina"
QC Information
 Available Slots:  Signal StDev Detection
  featureNames: 1475542110_F,...1475542113_F
  sampleNames: Biotin, ..negative
```

The output of `readBeadSumamryData` is an object of type `ExpressionSetIllumina` which is an extension of the ExpressionSet class developed by the Biocore team used as a container for high-throughput assays. The data from the raw data file has been written to the assayData slot of the object, whereas the phenoData slot contains information from sample sheet and the QC slot contains the quality control information. For consistency with the definition of other `ExpressionSet` objects, we use `exprs` and `se.exprs` to access the expression and standard error slots.

Analysis of Bead Summary Data

The quality control information read as part of `readBeadSummaryData` can be retrieved using `QCInfo` and plotted using `plotQC`, which gives an overview of each control type. Plots of particular controls (e.g., negative controls) can be produced using `singleQCPlot` with the usual R plotting arguments available.

Scatter and M (difference) vs. A (average) plots can be generated for multiple arrays using the `plotMAXY` function (Figure 4). These can give a visual indicator of the variability between arrays in an experiment. For replicate arrays, we would expect to see the majority of points lying on the horizontal for the MA plots and along the diagonal for the scatter plots. Systematic deviation from these lines may indicate a bias between the arrays which requires normalising. An MA or scatter plot can also be produced for just two arrays at a time (`plotMA` and `plotXY` respectively). An attractive feature of these plots is that the location of particular genes (e.g., controls) can be highlighted using the `genesToLabel` argument.

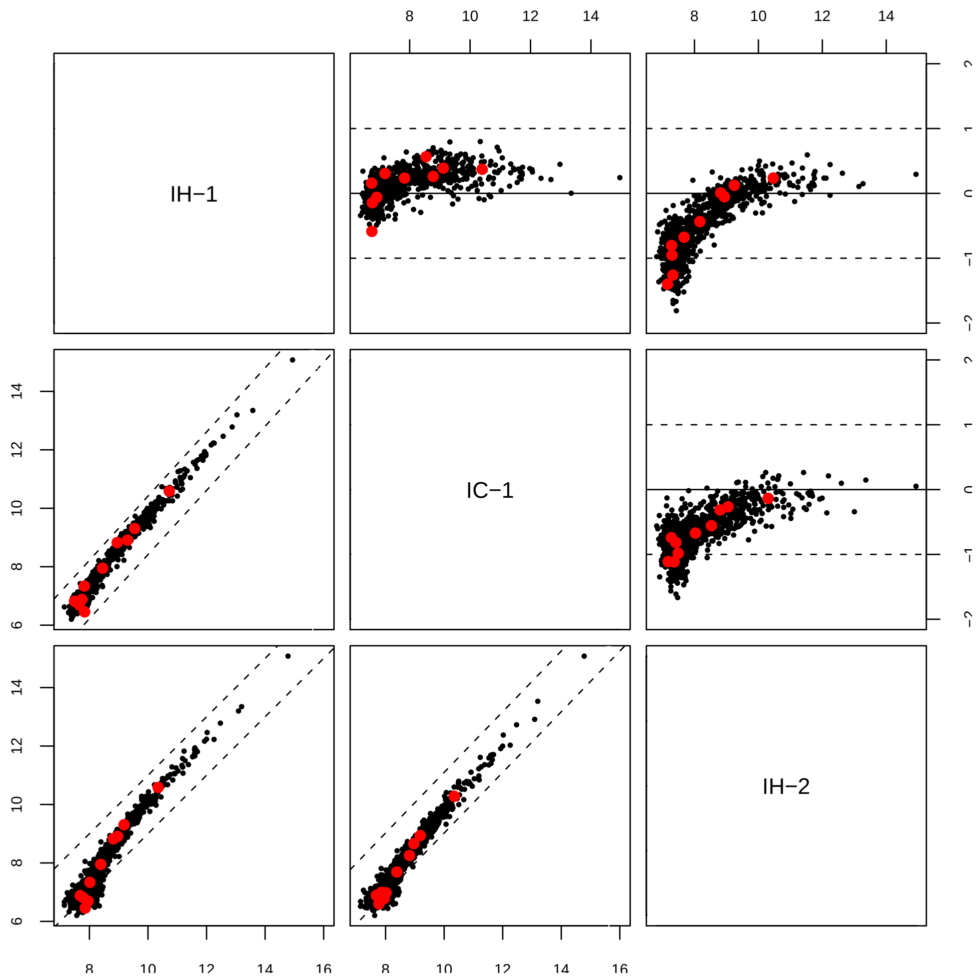

Figure 4: The `plotMAXY` function can be used to compare bead summary data from multiple arrays in the same experiment. In this figure we compare three replicates from the example bead summary data provided with **beadarray**

Since the `ExpressionSetIllumina` class includes a matrix of expression values, it can be analysed in a similar manner to data obtained from other technologies. In particular, this enables normalisation of BeadArray data to be carried out using the existing methods available in other Bioconductor packages (such as those available within the **affy** package, using `assayDataElementReplace` to replace the `exprs` slot). Illumina recommend normalising data by subtracting the average value of negative controls from each array. This method is implemented in the `backgroundNormalise` function and has quite a dramatic effect at lower intensities, often producing a lot of negative values. Typically, we find that the variability between arrays is low and a quantile or median normalisation (the function `medianNormalise` in **beadarray**) is sufficient.

The linear modelling tools within **limma** (Smyth, 2004) can be applied to the log-transformed expression `exprs` matrix in order to detect genes which are differentially expressed between arrays. The Illumina custom method is implemented by the function `DiffScore` but at present is only able to make pairwise comparisons between arrays.

Computational Issues and Future Plans

The vast amounts of data that can be generated from BeadArray experiments present a number of challenges for researchers, especially for analyses based on the bead level data. One has to consider that there are 12 80MB TIFF images for each BeadChip and 96 6MB TIFF Images for a SAM. In the case of a BeadChip experiment, simply reading the data into R for arrays from more than one BeadChip is problematic. We find that at least 1 Gb of RAM is required to run the `readBeadLevelData` and `createBeadSummaryData` functions on a `BeadLevelList` object representing one BeadChip. We hope to implement methods for normalisation which take the full bead level information into account but anticipate that this is going to be a computationally expensive task and may require the package to take advantage of parallel computing tools for R. Future versions of the package will also have better methods for creating bead averages which take information from replicate arrays into account.

Another major addition planned for **beadarray** is to allow sequence annotation information to be imported so that Illumina expression data can be combined with other microarray technologies such as arrayCGH, SNP and DNA methylation arrays. We plan to include functionality to read Illumina SNP and methylation data into the package.

Useful Illumina Resources

The vignettes supplied with the package give more detailed examples of how to analyse both bead level and bead summary data. Our previous paper (Dunning et al., 2005) provides descriptions of the image processing steps used by Illumina and some examples of bead level analysis. Readers interested in a comparison between Illumina and Affymetrix technologies are referred to Barnes et al. (2005).

We are keen to hear comments and feedback from users of **beadarray**.

Acknowledgements

We thank Brenda Kahl and Semyon Kruglyak (Illumina), Barbara Stranger, Matthew Forrest and Manolis Dermitzakis (Wellcome Trust Sanger Institute), and Andrew Lynch and John Marioni (University of Cambridge) for many helpful discussions during the devlopment of this work. We would also like to thank Isabelle Camilier (Ecole Polytechnique) for implementing the Illumina image processing algorithms and Roman Sasik (University of California, San Diego) for providing C code for reading TIFF images. The authors were supported by grants from Cancer Research UK (MS, NT & ST) and the Medical Research Council (MD). Simon Tavaré is a Royal Society / Wolfson Research Merit Award holder.

Bibliography

MJ Dunning, NP Thorne, I Camilier, *et al.* Quality control and low-level statistical analysis of Illumina BeadArrays. *Revstat*, 4:1–30, 2006.

RC Gentleman, VJ Carey, DM Bates, *et al*. Bioconductor: open software development for computational biology and bioinformatics. *Genome Biol*, 5:R80, 2004.

KL Gunderson, S Kruglyak, MS Graige, *et al*. Decoding randomly ordered DNA arrays. *Genome Res*, 14:870–877, 2004.

K Kuhn, SC Baker, E Chudin, *et al*. A novel, high-performance random array platform for quantitative gene expression profiling. *Genome Res*, 14:2347–2356, 2004.

L Gautier, L Cope, BM Bolstad, *et al*. **affy**–analysis of Affymetrix GeneChip data at the probe level. *Bioinformatics*, 20(3):307–15, 2004.

GK Smyth. Linear models and empirical Bayes methods for assessing differential expression in microarray experiments. *Statistical Applications in Genetics and Molecular Biology*, 3:113–136, 2004.

GK Smyth. Limma: linear models for microarray data. In R Gentleman, V Carey, W Huber, *et al.* *Bioinformatics and Computational Biology Solutions using R and Bioconductor*, pages 397–420. Springer, New York, 2005.

M Barnes, J Freudenberg, S Thompson, *et al.* Experimental comparison and cross-validation of the Affymetrix and Illumina gene expression analysis platforms. *Nucleic Acids Res*, 33:5914–5923, 2005.

Mark Dunning, Mike Smith, Natalie Thorne and Simon Tavaré
Computational Biology Group
Hutchison / MRC Research Centre
Department of Oncology
University of Cambridge
Hills Rd, Cambridge CB2 2XZ
United Kingdom
md392@cam.ac.uk
mls40@cam.ac.uk
npt22@cam.ac.uk
s.tavare@damtp.cam.ac.uk

Transcript Mapping with High-Density Tiling Arrays

by Matthew Ritchie and Wolfgang Huber

Introduction

Oligonucleotide tiling arrays allow the measurement of transcriptional activity and DNA binding events at a much higher resolution than traditional microarrays. Compared to the spotted technology, tiling arrays typically contain between 10 and 1000 times as many probes, which may be ordered or 'tiled' along entire chromosomes, or within specific regions of interest, such as promoters.

For RNA analysis, tiling arrays can be used to identify novel transcripts, splice variants, and antisense transcription (Bertone et al., 2004; Stolc et al., 2005). In DNA analysis, this technology can identify DNA binding sites through chromatin immunoprecipitation (ChIP) on chip analysis (Sun et al., 2003; Carroll et al., 2005) or genetic polymorphisms and chromosomal rearrangements via comparative genome hybridization (arrayCGH).

Due to the wide range of applications of this technology and the custom nature of the probe layout, the analysis of these data is different to that of regular microarrays. In this article, the **tilingArray** package, which extends the existing Bioconductor toolset to the problem of measuring transcriptional activity using Affymetrix high-density tiling arrays, is presented.

Background

The initial processing steps of quality assessment and normalization which are routinely applied to lower density arrays are also important when analyzing tiling array data. Diagnostic plots of the raw probe intensity data can highlight systematic biases or artefacts which may warrant the need for individual arrays or batches of arrays to be repeated. Normalization between arrays is necessary when data from multiple hybridizations is to be combined in an analysis. In the **tilingArray** package, a normalization method which uses the probe intensities from a DNA hybridization as a reference is implemented (Huber et al., 2006). The next step in the analysis is to detect the transcript boundaries. A simple change-point model, which segments the ordered chromosomal intensity data into discrete units has proven quite useful for whole genome tiling array data (David et al., 2006). Other approaches which use Hidden Markov Models (Toyoda and Shinozaki, 2005; Munch et al., 2006) or moving averages (Schadt et al., 2004) have also been proposed. Displaying the data with reference to the position along the chromosome allows visualization of the segmentation results. These capabilities will be demonstrated in the following sections.

The custom Affymetrix arrays used in this article were produced for the Stanford Genome Technology Center and tile the complete genome of *Saccharomyces cerevisiae* with 25mer probes arranged in steps of 8 bases along both strands of each chromosome. The two tiles per chromosome are offset by 4 bases (see Figure 1). Both perfect match (PM) and mismatch (MM) probes were available. The experimental data we analyze comes from David et al. (2006), and includes 5 RNA hybridizations from yeast cells undergoing exponential growth and 3 DNA hybridizations of labelled genomic fragments. This data is publicly available in the **davidTiling** package or from ArrayExpress (accession number E-TABM-14). A cell cycle experiment made up of RNA hybridizations from 24 time-points sampled at 10 minute intervals and 3 DNA hybridizations will also be used.