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.

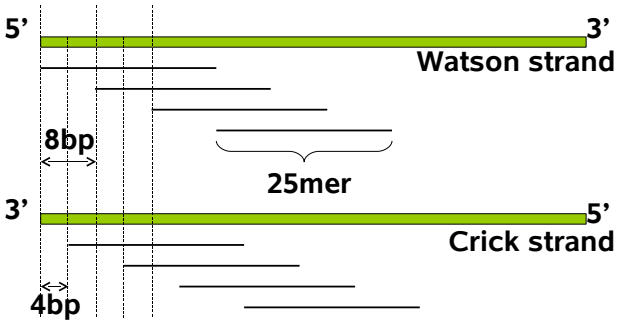

Figure 1: Probe spacing for *Saccharomyces cerevisiae* tiling arrays. The 25 mer probes are offset by 8 bases and tile each strand of DNA.

Reading data

We assume that the .CEL files from the **davidTiling** package are unzipped and available in the R working directory. To read in these data, the following commands can be used.

```
> library("tilingArray")
> cels = dir(pattern = ".cel")
> e = readCel2eSet(cels, rotated = TRUE)
```

The `readCel2eSet` function is is a wrapper for `ReadAffy` from the **affy** package. Rotating the .CEL file data through 90 degrees clock-wise by setting `rotated=TRUE` was necessary for these arrays due to old scanner settings. The data can also be loaded with

```
> library("davidTiling")
> data("davidTiling")
> dim(davidTiling)

Features  Samples
 6553600        8
```

The 8 arrays each contain 6,553,600 probes arranged in a grid with 2560 rows x 2560 columns. A special data structure is necessary for the mapping between the probes on the array and their target regions in the genome. For the yeast tiling array, we use an environment, named `probeAnno`, which organizes this information in two ways. Firstly, for each chromosomal strand, a vector of probe identifiers, in linear genomic order, and the coordinates and type of match is stored. In addition, the type of match and identifier of its hit region (e.g. YBR275C) are stored in the order that the probes appear on the array. To load, and find out further details about this environment, type

```
> data("probeAnno")
> ? probeAnno
```

To create this environment, the probes were mapped to the yeast genome using the MUMmer program (Delcher et al., 2002). In order to screen out ambiguous probes from the analysis, probes with multiple matches to the yeast genome where flagged and discarded from the analysis.

Users who wish to apply the **tilingArray** package to other types of tiling arrays, for example, from other species, need to produce their own `probeAnno`-like environment. Currently, there is no support for doing this. An objective of future work is to make this process and the genome mapping data structures more generic. We aim to use the infrastructure that will be provided by the **oligo** package for this purpose.

Once the data has been imported, some simple diagnostic plots can be generated with

```
> qcPlots(davidTiling, probeAnno = probeAnno)
```

This command generates an HTML report with image plots, box plots and density plots of log base 2 intensity data in the current working directory.

Normalization

The **tilingArray** package implements a DNA based normalization strategy in the `normalizeByReference` function. Normalization of the `davidTiling` data is carried out with the following commands

```
> isDNA = davidTiling$nucleicAcid ==
+     "genomic DNA"
> isRNA = davidTiling$nucleicAcid ==
+     "poly(A) RNA"
> pm = PMindex(probeAnno)
> bg = BGindex(probeAnno)
> yn = normalizeByReference(davidTiling[,
+     isRNA], reference = davidTiling[,
+     isDNA], pm = pm, background = bg)
```

The logical vectors `isDNA` and `isRNA` indicate which arrays are DNA and RNA hybridizations respectively. The intensities measured on the DNA hybridizations are used for two purposes. First, the probes indexed by the `bg` vector are used to estimate the background signal. Second, the PM intensities (indexed by the `pm` vector) provide a reference signal which is used to correct for probe specific responses due to base content. Once the PM intensities from the RNA hybridizations have been adjusted for background signal and probe effects, variance-stabilizing normalization between arrays using the `vsn` function (Huber et al., 2002) is applied to the data. For details of the method, see Huber et al. (2006).

Segmentation

Transcript boundaries are estimated from the data using a structural change model (SCM). Assume we have normalized intensity values z_{ki} from arrays $i = 1, \ldots, I$ and probes $k = 1, \ldots, n$ where the probe indexes (k) order the data by increasing position along the chromosome. We fit the SCM model

$$z_{ki} = \mu_s + \varepsilon_{ki} \qquad \text{for } t_s \leq k < t_{s+1} \tag{1}$$

which has mean signal μ_s for the s-th segment, and residuals ε_{ki}. The change-points, $t_1, \ldots, t_S$ are the coordinates of the segment boundaries. This model was applied to arrayCGH data in Picard et al. (2005). The model is fitted separately for each strand of each chromosome. The algorithm is implemented in the function `segment`, which can be used directly on a matrix of data ordered along the chromosome.

To standardize some of the common data preprocessing steps, such as extracting the data for the chromosome of interest from `yn`, we use the wrapper function `segChrom`. To segment the data from chromosome 1, use

```
> seg1 = segChrom(yn, probeAnno = probeAnno,
+     chr = 1)
```

The `segment` algorithm is both time and memory intensive. On the example data, it uses a maximum of around 8 GB of RAM on larger chromosomes and takes several hours to complete. The computations for different chromosomes are trivially parallelizable, and the function `segChrom` offers a primitive mechanism for parallelization by synchronization through lock files.

The key parameter which the user must specify to the `segChrom` function is the maximum number of segments (S) to fit. The dynamic programming algorithm will then fit models with $1, 2, \ldots, S$ segments. S is specified via the parameter `nrBasesPerSegment`, the average number of bases per segment, which is used to set S by dividing the chromosome length by this number. For the data in David et al. (2006), the value `nrBasesPerSegment=1500` was chosen based on biological expectations and by manually inspecting the results obtained by varying this parameter. This value is obviously specific for these particular data. For other chip types or species, the value of `nrBasesPerSegment` needs to be adapted to the data.

The change-points t_s indicate transcript boundaries. Confidence intervals for the boundary locations are calculated using the **strucchange** package (Zeileis et al., 2002) when the `confint` argument of `segChrom` is set to `TRUE`.

Following the identification of transcript boundaries and levels through the segmentation algorithm, expression level changes for a given segment can be measured between different experimental conditions. Another approach to look for expression changes is to calculate a statistic for the condition-dependence for each probe and run the segmentation on this statistic.

Chromosome plots

Raw or normalized probe intensities can be plotted in along the chromosome order using the `plotAlongChrom` function. This function assumes that data is available for both strands of DNA and requires a `probeAnno` environment. Annotated genomic features can be included in these displays where a `GFF` (General Feature Format, see `http://www.sanger.ac.uk/Software/formats/GFF`) data frame is available.

An along the chromosome plot of the **davidTiling** data can be created with

```
> data("gff")
> plotAlongChrom(segObj = seg1,
+     probeAnno = probeAnno, gff = gff,
+     what = "dots", chr = 1, coord = c(156500,
+         160000))
```

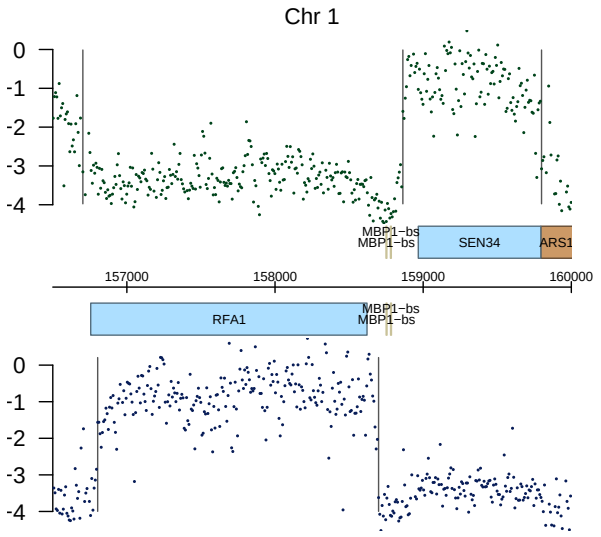

Figure 2: Plot of normalized intensity data (averaged between arrays) versus position along the chromosome (in bases) for a region of chromosome 1. The top and bottom panels display the probes from the Watson (+) and Crick (-) strands respectively. The gray lines indicate the segment boundaries (t_s) estimated from the data using `segChrom`. These boundaries correspond very closely with the known coordinates of the SEN34 and RFA1 genes, marked by blue boxes.

This plot (Figure 2) shows the result of the segmentation from the previous section; gray vertical lines indicate the transcript boundaries (t_s) and each point represents a probe. The `chr` and `coord` arguments specify the region plotted. The light blue boxes show the open reading frames of protein coding genes. Binding sites and other features from the

`gff` data frame are also marked on the figure in the relevant locations. The absolute level of expression for different transcripts can be visualized in this display.

The following commands can be used to generate a heatmap display of the normalized intensities from the cell cycle data set (available in the object `cycle`), for the same region on chromosome 1 as in Figure 2.

```
> plotAlongChrom(y=cycle,
 + probeAnno=probeAnno, gff=gff,
 + what="heatmap", chr=1,
 + coord=c(156500,160000))
```

Figure 3 shows the resulting output. The periodic expression patterns for the cell cycle regulated SEN34 and RFA1 transcripts are clearly visible in this figure.

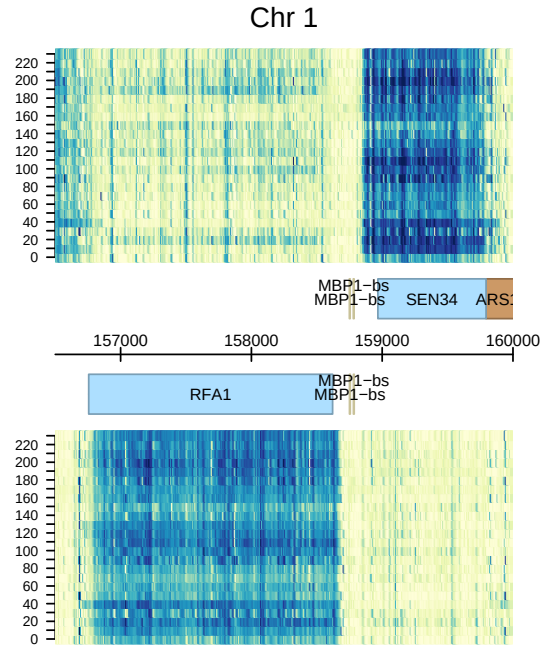

Figure 3: Heatmap plot of probe intensities from a cell cycle experiment for a region of chromosome 1. Each row in the y-direction displays the normalized intensities from a different time-point (0 minutes through to 230 minutes). The x-axis shows the position along the chromosome (in bases). The colorscheme indicates the intensity level, ranging from light yellow for lower intensities, through to dark blue for higher intensities.

Summary

The **tilingArray** package provides tools for reading, normalizing, segmenting and visualizing Affymetrix tiling array data. The core functions are `normalizeByReference` and `segment`, and their underlying methodology is described in Huber et al. (2006). These functions should be widely applicable to tiling array data.

There are also some functions and objects that have been customized to our specific *Saccharomyces cerevisiae* data sets, such as `segChrom`, `plotAlongChrom` and `probeAnno`. These functions can be used as templates for transferring the methods to other types of arrays and species, but this will require some work by the user. In future releases, we hope to use the infrastructure for tiling array data that will be offered by the **oligo** package to make these tools more generic.

Many exciting, open research questions still remain including a data-driven approach for selecting an optimal number of segments (S) for each chromosome (Huber et al., 2006), applying SCMs beyond piece-wise constant curves and the segmentation of condition-dependent transcription patterns.

Acknowledgements

We thank Jörn Tödling and Zhenyu Xu for contributions to the development of the **tilingArray** package, Lior David, Marina Granovskaia, Sandra Clauder-Münster and Lars Steinmetz for providing the data used in this article, Eugenio Mancera for Figure 1 and Rachel Uren for proof-reading the article.

Bibliography

P. Bertone, V. Stolc, T. E. Royce *et al.* Global identification of human transcribed sequences with genome tiling arrays. *Science*, 306(5705):2242–2246, 2004.

J. S. Carroll, X. S. Liu, A. S. Brodsky *et al.* Chromosome-wide mapping of estrogen receptor binding reveals long-range regulation requiring the forkhead protein FoxA1. *Cell*, 122:33–43, 2005.

L. David, W. Huber, M. Granovskaia *et al.* A high-resolution map of transcription in the yeast genome. *Proc Natl Acad Sci USA*, 103(14):5320–5325, 2006.

A. L. Delcher, A. Phillippy, J. Carlton and S. L. Salzberg. Fast algorithms for large-scale genome alignment and comparison. *Nucleic Acids Res*, 30 (11):2478–2483, 2002.

W. Huber, A. V. Heydebreck, H. Sültmann *et al.* Variance stabilization applied to microarray data calibration and to the quantification of differential expression. *Bioinformatics*, 18(Suppl 1):S96–S104, 2002.

W. Huber, J. Toedling and L. M. Steinmetz. Transcript mapping with oligonucleotide high-density tiling arrays. *Bioinformatics*, 22(16):1963–1970, 2006.

K. Munch, P. Gardner, P. Arctander and A. Krogh. A hidden Markov model approach for determining expression from genomic tiling micro arrays. *BMC Bioinformatics*, 7:239, 2006.

F. Picard, S. Robin, M. Lavielle *et al.* A statistical approach for array CGH data analysis. *BMC Bioinformatics*, 6(1):27, 2005.

E. E. Schadt, S. W. Edwards, D. GuhaThakurta *et al.* A comprehensive transcript index of the human genome generated using microarrays and computational approaches. *Genome Biol*, 5(10):R73, 2004.

V. Stolc, M. Samanta, W. Tongprasit *et al.* Identification of transcribed sequences in Arabidopsis thaliana by using high-resolution genome tiling arrays. *Proc Natl Acad Sci USA*, 102(12):4453–4458, 2005.

L. V. Sun, L. Chen, F. Greil *et al.* Protein-DNA interaction mapping using genomic tiling path microarrays in Drosophila. *Proc Natl Acad Sci USA*, 100:9428–9433, 2003.

T. Toyoda and K. Shinozaki. Tiling array-driven elucidation of transcriptional structures based on maximum-likelihood and Markov models. *Plant J*, 43(4):611–621, 2005.

A. Zeileis, F. Leisch, K. Hornik and C. Kleiber. strucchange: An R package for testing for structural change in linear regression models. *Journal of Statistical Software*, 7:1–38, 2002.

Matt Ritchie
European Bioinformatics Insitute (EBI)
European Molecular Biology Laboratory (EMBL) Cambridge, UK
ritchie@ebi.ac.uk

Wolfgang Huber
European Bioinformatics Insitute (EBI)
European Molecular Biology Laboratory (EMBL) Cambridge, UK
huber@ebi.ac.uk

Analyzing Flow Cytometry Data with Bioconductor

by Nolwenn Le Meur and Florian Hahne

Introduction

In the recent past, flow cytometry (FCM) has become a high-throughput technique used in both basic and clinical research. Applications range from studies focusing on the immunological status of patients, therapeutic approaches involving stem cells up to functional screens used to identify specific phenotypes. The technology is capable of measuring multiple fluorescence as well as some morphological properties of individual cells in a cell population on the basis of light emission. FCM experiments can be extremely complex to analyze due to the large volume of data that is typically created in several processing steps. As an example, flow cytometry high content screening (FC-HCS) can process at a single workstation up to a thousand samples per day each containing thousands of cells, monitoring up to eighteen parameters per sample. Thus, the amount of information generated by these technologies must be stored and managed and finally needs to be summarized in order to make it accessible to the researcher.

Instrument manufacturers have developed software to drive the data acquisition process of their cytometers, but these tools are primarily designed for their proprietary instrument interface and offer few or no high level data processing functions. The packages **rflowcyt** and **prada** provide facilities for importing, storing, assessing and preprocessing data from FCM experiments. In this article we demonstrate the use of these packages for some common tasks in flow cytometry data analysis.

FCS format

In order to facilitate data exchange across different platforms, a data standard has been developed which is now widely accepted by the flow cytometry community and also by most instrument manufacturers. Flow Cytometry Standard (FCS) binary files contain both raw data and accompanying meta data of individual cytometry measurements and optionally the results of prior analyses carried out on the raw data (Seamer et al., 1997). The current version of the FCS standard is 3.0, but both packages can also deal with the old 2.0 standard which is still widely used. We can import FCS files into R using the function `read.fcs`.

Data models

Both **rflowcyt** and **prada** use their own object models to deal with FCM data. While the focus of **rflowcyt** is more on single cytometry measurements, **prada** offers the possibility to combine several individual measurements in the confines of a single experiment