Introduction to the les package: Loci of Enhanced Significance in Tiling Array Data

Julian Gehring

June 29, 2010

Abstract

In this vignette we describe using the les package for finding Loci of Enhanced Significance (LES) in tiling microarray data. With an example of a general framework we illustrate how to apply the package for exploring regions of regulation in differential expression and chip-CHIP analysis.

1 Introduction

Tiling microarrays have become an important platform for the investigation of regulation in expression and DNA-protein interaction. They provide a relatively unbiased tool covering large regions of interest in the genome.

Beside the analysis of single microarrays the investigation of differential effects between experimental conditions is critical for current research. A common approach consists in applying statistical tests on the level of single probes and thereby computing p-values for each probe independently. Since the targets of such experiments cover areas with several probes the logical next step involves combining information from neighboring probes into a reasonable statistic. In regions with differential effects the test statistics change their distribution and are referred to as Loci of Enhanced Significance (LES). The changes in the test statistics depend on the underlying test applied.

The les package provides the ability to detect such LES independent of the underlying statistical test and can therefore be used for a wide range of applications. This vignette illustrates how to LES can be found in tiling microarray data sets.

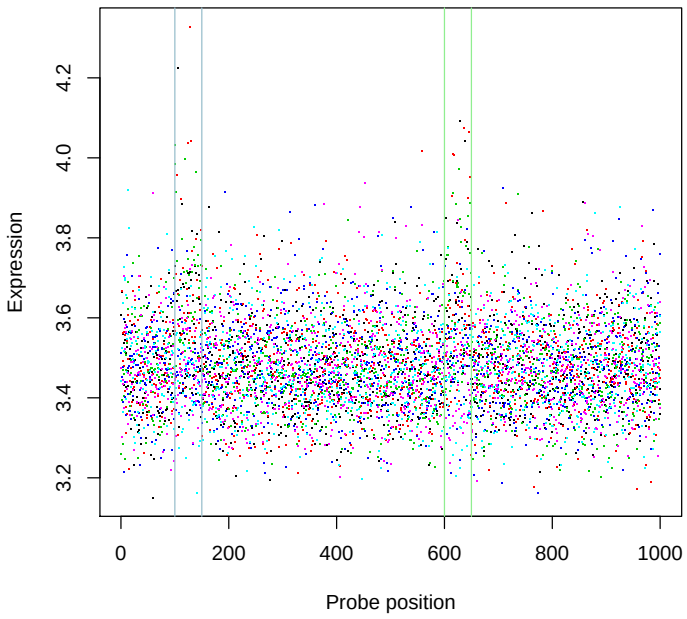
2 Data and statistics on probe level

For this analysis we will use a simulated data set describing differential expression between two conditions. It contains 1000 probes with 3 chips each for the conditions treatment and control. The expression values are stored in an expression set. We will extract the position of the probes, the conditions of the samples and the expression values. There are two regions with changes present in the data, each being 50 bp long.

```
> library(les)
> library(Biobase)
> data(simTile)
> treatment <- as.factor(phenoData(simTile)$condition ==
+     "treatment")
> pos <- featureData(simTile)$position
> exprs <- exprs(simTile)
> regions <- c(100, 150, 600, 650)
> cols <- rep(c("lightblue3", "lightgreen"), each = 2)
```

Next we have a look at the expression values.

```
> matplot(exprs, pch = ".", xlab = "Probe position", ylab = "Expression")
> abline(v = regions, col = cols)
```

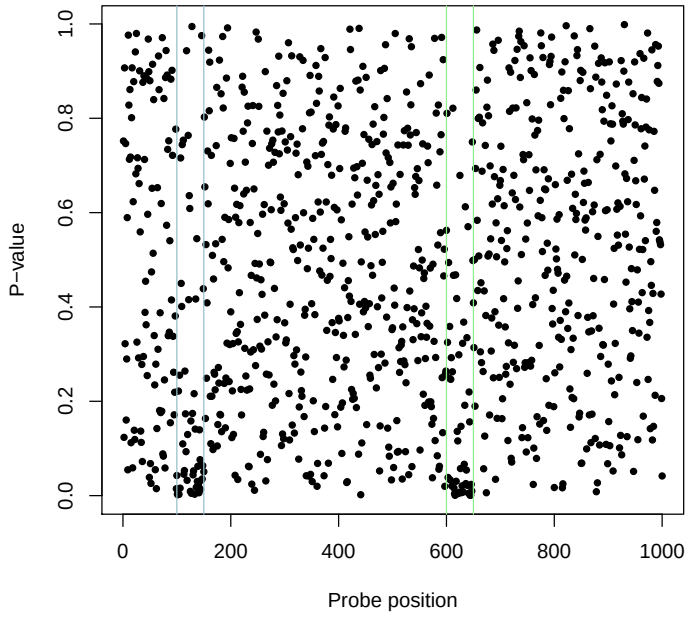


In the next step we will compute the statistics of changes between the two conditions for each probe. Since the sample size as for most tiling microarray experiments is small we will use a modified t-test provided by the limma package.

We will also plot the p-values against the probe positions. By looking at the raw p-values it may be hard to detect the LES.

```
> library(limma)
> design <- cbind(mean = 1, diff = treatment)
> fit <- lmFit(exprs, design)
> fit <- eBayes(fit)
> pval <- as.numeric(fit$p.value[, "diff"])

> plot(pos, pval, pch = 20, xlab = "Probe position", ylab = "P-value")
> abline(v = regions, col = cols)
```



3 Incorporation information from neighboring p-values to find LES

In any well designed tiling array experiments the potential targets will cover several neighboring probes. Thereby neighboring p-values should contain mutual information and incorporation of such will be beneficial.

In the les package this in done in the following manner: For each probe i the the surrounding p-values get weights assigned by some windowing function. A weighted cumulative density is then computed and the fraction of significant p-values is estimated by iterative linear fitting. The method is based on the fact that p-values under the null hypothesis H_0 are from a uniform distribution whereas p-values violating H_0 are shifted towards smaller values. This results in the index Λ_i which measures the fraction of p-values violating H_0 and therefore the degree of regulation in the local surrounding. It should be noted that this approach is closely related to the estimation of a false discovery rate and Λ_i can be interpreted as locally weighted version of such.

For the analysis we will first pass our data to an object of class Les. The only data required are the position of the probes, the corresponding p-values from the statistical test and optionally their chromosomal location.

Then we can compute our first estimate of Λ_i for which we have to specify a window size. The power of the detection will be maximal if the size of the window matches the size of the target. In many experiments one has a rough prior knowledge on the target size which will be sufficient of the first step. Later we will discuss an approach the estimate the optimal window size from the data itself. Here we will chose a small window size to start with.

```
> res <- create(pos, pval)
```

By default a triangular weighting function will be chosen. We can also take a rectangular window or write our own function and pass it via the weighting argument. We can further on specify whether we want to include the Grenander correction for the cumulative density or use multicore processing on some platforms.

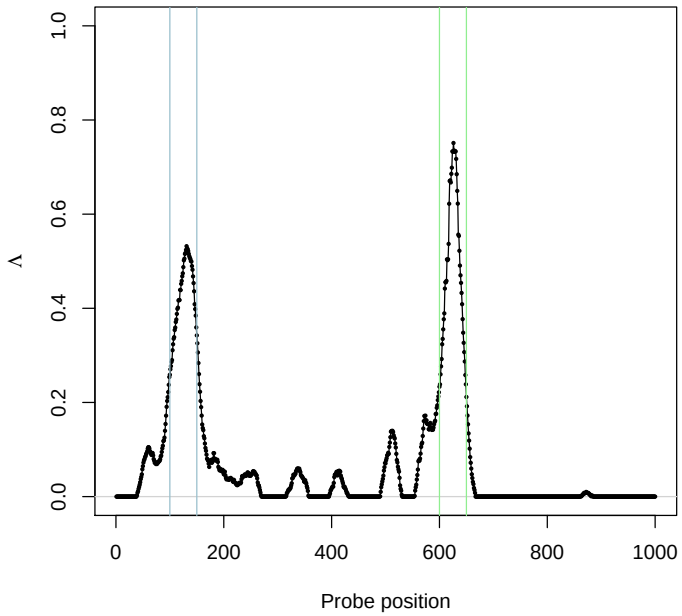
```
> win <- 30
> res <- estimate(res, win)
```

All data, results and parameters are stored in the object which is in our case called res. We can get a short summary on the results by calling res or by plotting it.

```
> res
```

```
** Object of class 'Les' **
* 1000 probes on 1 chromosomes
* Lambda in range [0, 0.751172] with kernel size 30
```

```
> plot(res)
> abline(v = regions, col = cols)
```



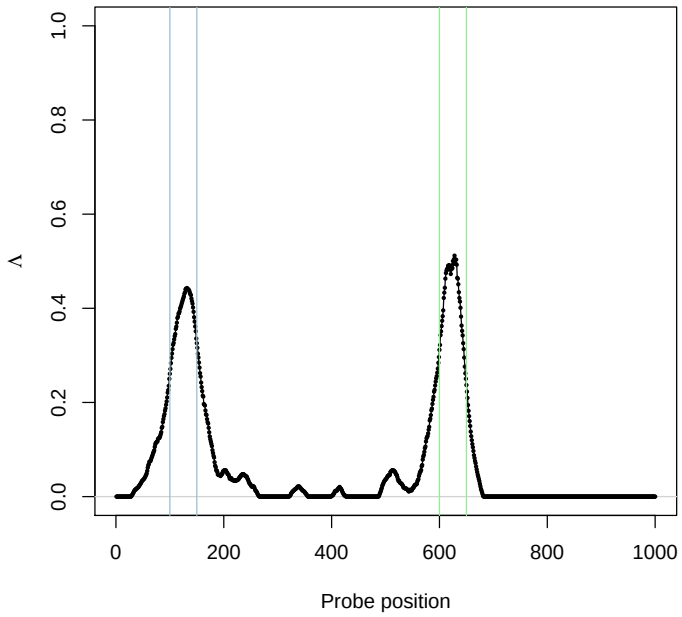
For comparison we will analyze and plot the same data with a different window size. This allows us to explore our data set.

```
> win2 <- 50
> res <- estimate(res, win2)

> res

** Object of class 'Les' **
* 1000 probes on 1 chromosomes
* Lambda in range [0, 0.511849] with kernel size 50

> plot(res)
> abline(v = regions, col = cols)
```



We can already see two distinct peaks that correspond to the regions of simulated regulation.

4 Parameter estimation from the data

To turn the continuous Λ_i into regions of interest we have to define a threshold Θ. It can be derived from the data by estimating the number of probes with a significant effect R on the whole array. Then Θ can be chosen such that $\mid \Lambda_i \geq \Theta \mid = R$. The content of any slot can be accessed by using the [function.

```
> res <- cutoff(res, grenander = TRUE, verbose = TRUE)

[1] "52 significant probes estimated with limit lambda>=0.409992"

> res["nSigProbes"]

    GSRI
51.71602
```

Based on Θ we can search for regions that have a continuous $\Lambda_i \geq \Theta$. The regions function takes by default the estimated $\hat{\Theta}$ as shown before. We can also pass our own Θ. Further restrictions can be imposed on the regions search such

as the minimal length of a region and the maximum gap allowed between probes of one region.

```
> theta <- 0.3
> res <- regions(res, limit = theta)
> res["regions"]
```

```
  start end size nProbes     ri     se      rs chr
1   104 152   49      49 0.4558 0.0092 49.4698   0
2   600 645   46      46 0.5550 0.0246 22.5977   0
```

The optimal window for a region can be estimated using a leave-one-out cross validation. This approach can be used to optimize the window size parameter for each region seperately in a second step.

```
> winSize <- seq(10, 100, by = 10)
> reg2 <- matrix(c(500, 700), nrow = 1)
```

5 Calculation of confidence intervals

In some cases it is also useful to provide confidence (CI) intervals for Λ_i. These are computed by bootstrapping the probes in the window. Since bootstrapping is by its nature computationally demanding and CI are primarily interesting in regions of interest it it possible to compute CI for a subset of probes.

```
> subset <- pos >= 580 & pos <= 670
```

The two RDEs can clearly be detected.
Now look at the second RDE and plot also the 0.95 CIs.

6 Specification of own window functions

With the triangWeight and rectangWeight two window functions are already include in the les package. We can also specifiy own window functions and pass it via the weighting argument in the estimate function. The have to be given in the following format:

```
> weightFcn <- function(distance, win) {
+     weight <- 1 - distance/weight
+     return(weight)
+ }
```

7 References

A Session information

```
> sessionInfo()

R version 2.11.1 (2010-05-31)
x86_64-pc-linux-gnu

locale:
 [1] LC_CTYPE=de_DE.UTF-8       LC_NUMERIC=C
 [3] LC_TIME=de_DE.UTF-8        LC_COLLATE=de_DE.UTF-8
 [5] LC_MONETARY=C              LC_MESSAGES=de_DE.UTF-8
 [7] LC_PAPER=de_DE.UTF-8       LC_NAME=C
 [9] LC_ADDRESS=C               LC_TELEPHONE=C
[11] LC_MEASUREMENT=de_DE.UTF-8 LC_IDENTIFICATION=C

attached base packages:
[1] tools     stats     graphics  grDevices utils     datasets
[7] methods   base

other attached packages:
[1] limma_3.4.3   Biobase_2.8.0 les_0.8.0     fdrtool_1.2.6

loaded via a namespace (and not attached):
 [1] annotate_1.26.0      AnnotationDbi_1.10.1 boot_1.2-42
 [4] DBI_0.2-5            gdata_2.8.0          genefilter_1.30.0
 [7] gplots_2.8.0         GSRI_1.1.3           gtools_2.6.2
[10] multicore_0.1-3      RColorBrewer_1.0-2   RSQLite_0.9-1
[13] splines_2.11.1       survival_2.35-8      xtable_1.5-6
```