Introduction to the *les* package: Identifying Differential Effects in Tiling Microarray Data with the *Loci of Enhanced Significance* Framework

Julian Gehring, Clemens Kreutz

October 14, 2015

1 Introduction

Tiling microarrays have become an important platform for the investigation of regulation in expression, DNA modification, and DNA-protein interaction. Since unbiased in relation to annotation they provide an powerful tool for biological research. Beside the analysis of single conditions, the investigation of regulation between multiple experimental conditions is an important part of current research. A common approach consists in assessing the differential effect on the level of individual probes and thereby computing p-values p_i for each probe i with a suitable statistical test, for example an F-test, independently. Since the targets exhibiting differential effects cover multiple probes, a reasonable next step involves combining information from neighboring probes. While several approaches for this purpose exist, most of them lack a statistical interpretation of the results or are specific for certain platforms and statistical tests.

The approach used here utilizes the fact that p-values in regions without an differential effect are uniformly distributed. In the case that regulation is present the distribution of p_i is shifted towards zero; such regions are therefore referred to as *Loci of Enhanced Significance (LES)*. By assessing the uniform component of the p-value distribution within a sliding window, the local degree of significant probes along the genome is estimated. The *les* package implements the method for detecting *LES* in tiling microarray data sets, independent of the underlying statistical test and hence is suitable for a wide range of applications.

2 Data set and statistic testing on probe level

The data set used in this vignette is part of a quality control study for tiling microarrays [2], in which spike-ins were used to assess the influence of microarray platforms, preparation procedures, and analysis algorithms on the accuracy

and reproducibility of ChIP-chip experiments. Here, the expression intensities of one region from the *undiluted* data set investigated with Affymetrix tiling microarrays is selected, consisting of 452 probes and two conditions with three replicates each. The data has been normalized using quantile normalization and probe positions remapped to a common reference.

Normalized expression intensities are stored in the matrix `exprs`, with rows representing probes and columns arrays. The corresponding names contain the position of the probes and the condition of the samples, respectively. The properties of the spike-in DNA, yielding a region with differential effect, are described in the `reference` data frame.

```
> library(les)
> data(spikeInData)
> head(exprs)
```

```
               1        1        1        2        2        2
5230083 6.671288 6.826426 6.762979 6.779357 6.868966 6.782944
5230089 5.548030 5.478211 5.817685 5.679769 5.347560 6.009189
5230096 5.746862 5.487193 5.931810 5.639733 5.951060 5.570534
5230103 5.254493 5.070491 5.134345 5.821151 5.588138 5.152873
5230110 5.627828 5.656203 5.848418 5.425308 5.372586 5.249253
5230117 5.510349 5.912182 4.699285 5.112509 5.470490 5.181388
```

```
> dim(exprs)
```

```
[1] 452   6
```

```
> pos <- as.integer(rownames(exprs))
> condition <- as.integer(colnames(exprs))
> reference
```

```
     chr   start     end length fold nProbes
26 chr11 5232536 5233080    545    2      74
```

```
> region <- as.vector(reference[ ,c("start", "end")])
```

In order to assess the differential effect between the two conditions on the level of individual probes, we use a modified t-test provided by the *limma* package [3]. This offers an increased statistical power compared to a classical Student's t-test for small sample sizes, as present in most tiling microarray experiments.

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

```
> plot(pos, pval, pch=20, xlab="Probe position", ylab=expression(p))
> abline(v=region)
```

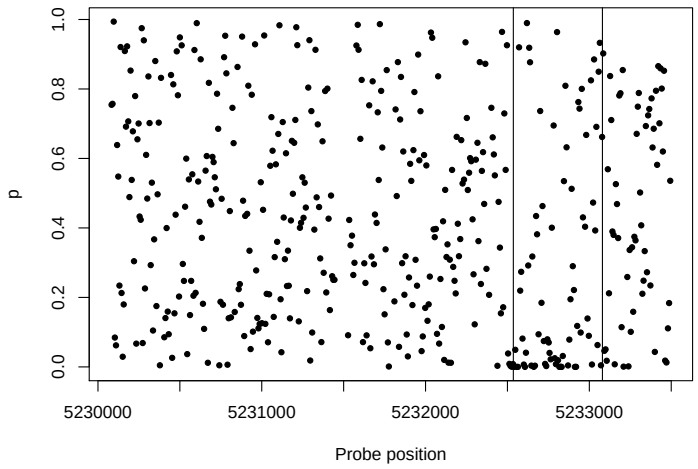

Figure 1: Probe-level p-values

3 Incorporation of neighboring probes

The accumulation of small p-values indicates a *LES* at the location of the spike-in, whereas the response of individual probes may be not reliable (Figure 1). Therefore, an incorporation of neighboring probes should be beneficial and yield improved results in the detection of the differential effect. For each probe i the surrounding p-values p_j get weights assigned, corresponding to the distance of probe i and j. Being free in the definition of the weighting function, the spatial relationship of probes in observing the same effect can be accounted for. A weighted cumulative density is computed and the fraction of significant p is estimated by an iterative regression. The method is based on the fact that p-values are uniformly distributed under the null hypothesis H_0 whereas p-values violating H_0 are shifted towards smaller values [1], which is closely related to the estimation of a false discovery rate. This results in Λ_i, constituting an estimate of the fraction of p-values violating H_0 and therefore the degree of significance in the local surrounding of the evaluated position.

For the analysis, we first store all relevant data in an object of class *Les* by calling the *Les* method. The only data required are the positions of the probes i, the corresponding p-values p_i from the statistical test, and optionally their chromosomal location.

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

Then we can compute our estimate Λ_i for which we have to specify a weighting window. Here, we use a triangular window taken as default with a size of 200bp. The power of detecting a region will be high if the window matches

approximately the properties of the regulated region. In many experiments a rough prior knowledge is available which can be sufficient for choosing a suitable window. Further, optimal weighting windows can be estimated from the data directly.

```
> res <- estimate(res, win=200)
```

All data, results and parameters are stored in the `res` object of class *Les*. We can get an overview of the results with the *print*, *summary*, or *plot* method (Figure 2).

```
> res

** Object of class 'Les' **
* 452 probes on 1 chromosomes
* Lambda in range [0, 0.531522] with window size 200

> summary(res)

** Object of class 'Les' **
* 452 probes on 1 chromosomes
* Lambda in range [0, 0.531522] with window size 200
* Available slots:
 [1] "pos"       "pval"      "chr"       "win"       "weighting"
 [6] "grenander" "minProbes" "method"    "lambda"    "lambda0"
[11] "nProbes"   "nChr"      "state"

> plot(res)
> abline(v=region)
```

For comparison we analyze and plot the same data with a rectangular weighting window (Figure 3). In this example the rectangular window leads to better results. The *les* package includes four predefined windows; custom functions can also be used, as described in Section 8.

```
> res2 <- estimate(res, win=200, weighting=rectangWeight)
> res2

** Object of class 'Les' **
* 452 probes on 1 chromosomes
* Lambda in range [0, 0.540663] with window size 200

> plot(res2)
> abline(v=region)
```

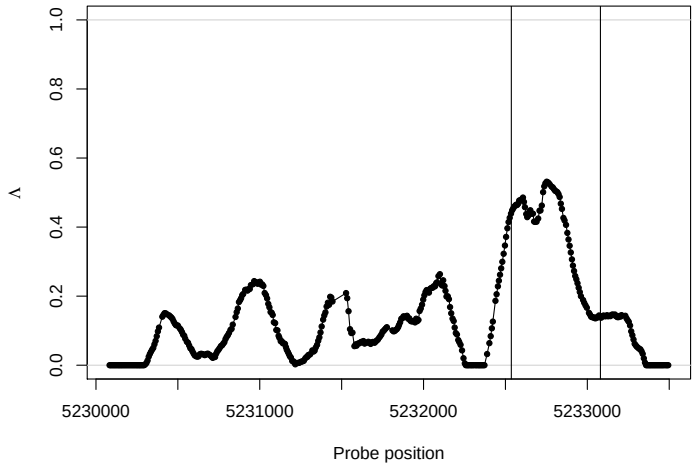

Figure 2: Results with triangular window.

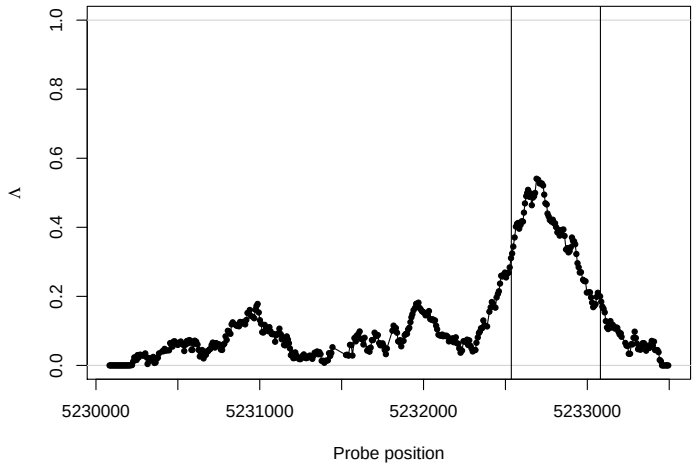

Figure 3: Results with rectangular window.

4 Parameter estimation

To turn the continuous Λ_i into distinct regions of interest we define a threshold Θ. It can be derived from the data by estimating the number of probes with a significant effect on the whole array.

```
> res2 <- threshold(res2, grenander=TRUE, verbose=TRUE)

[1] "53 significant probes estimated for Lambda >= 0.327448"
```

Given Θ we can look for regions that have a continuous $\Lambda_i \geq \Theta$. The *regions* method takes by default the estimated Θ from the previous step. We can also pass our own estimate for Θ with the `limit` argument. Further restrictions can be imposed on the regions such as the minimal length of a region and the maximum gap allowed between probes in one region. The *[* method allows to access any data slot of an object of class *Les*. Here, we use it to extract the data frame with the estimated regions.

```
> res2 <- regions(res2, verbose=TRUE)

[1] "1 regions found for lambda>=0.327448"

> res2

** Object of class 'Les' **
* 452 probes on 1 chromosomes
* Lambda in range [0, 0.540663] with window size 200
* 53 regulated probes estimated for lambda >= 0.327448
* 1 regions detected

> res2["regions"]

  chr   start     end size nProbes     ri     se      rs
1   0 5232548 5232926  379      53 0.5185 0.0087 59.4146

> plot(res2, region=TRUE)
> abline(v=region)
```

5 Calculation of confidence intervals

By bootstrapping probes in each window, confidence intervals for the statistic Λ_i can be computed. Since confidence intervals are primarily interesting in regions of interest and bootstrapping is by its nature computationally demanding, we can restrict the calculation to a subset of probes.

```
> subset <- pos >= 5232400 & pos <= 5233100
> res2 <- ci(res2, subset, nBoot=50, alpha=0.1)
> plot(res2, error="ci", region=TRUE)
```

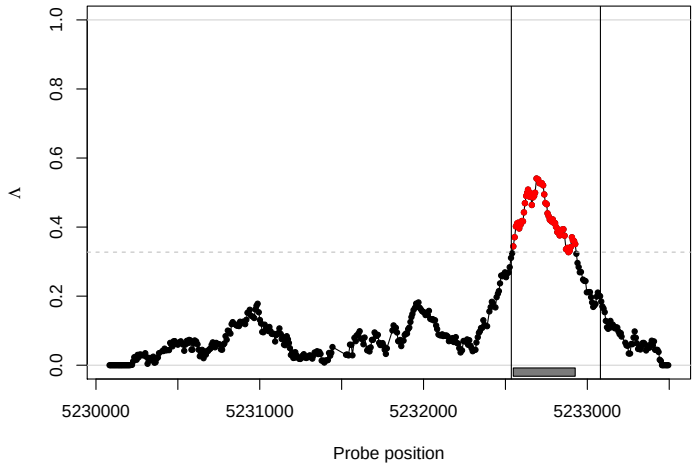

Figure 4: Results with estimates for regions.

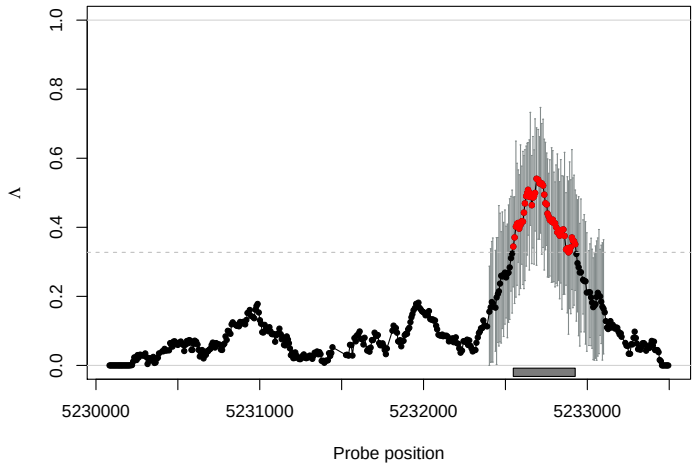

Figure 5: Results with confidence intervals and estimates for regions.

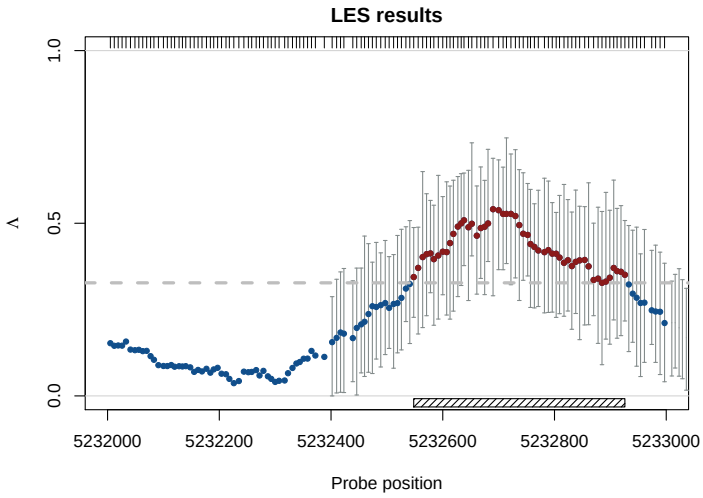

Figure 6: Results with customized graphical parameters.

6 Visualization

The *plot* method uses a special system in order to customize the graphical elements of the figure. It allows to refer to all its components with the name of the additional input argument; its value is a list containing named graphical parameters for the underlying plot function. As an example, we plot smaller region of the chromosome with confidence intervals, estimated region, and the probe density. Further, we adapt several parameters changing the graphical representation. For details, please refer to the help of the *les* package.

```
> plot(res2, error="ci", region=TRUE, rug=TRUE, xlim=c(5232000, 5233000), sigArgs=list(col='
```

7 Exporting results to external software

With the *export* method the estimated regions as well as Λ_i can be saved to standard files formats, in order to facilitate the export of the results to other software and genome browsers. The region estimates can be exported to the *bed* and *gff* formats, the test statistic Λ_i to the *wig* format.

```
> bedFile <- paste(tempfile(), "bed", sep=".")
> gffFile <- paste(tempfile(), "gff", sep=".")
> wigFile <- paste(tempfile(), "wig", sep=".")
> export(res2, bedFile)
> export(res2, gffFile, format="gff")
> export(res2, wigFile, format="wig")
```

8 Specification of custom weighting windows

With the `triangWeight`, `rectangWeight`, `epWeight`, and `gaussWeight` functions, four weighting windows are included in the *les* package, providing a triangular, rectangular, Epanechnikov, and Gaussian window, respectively. We can also specify custom window functions and pass it as `weighting` argument in the *estimate* method. They have to be specified as a function depending on the distance of the probes (`distance`) and the window size (`win`), as illustrated here with a triangular weighting.

```
> weightFoo <- function(distance, win) {
+ weight <- 1 - distance/win
+ return(weight)
+ }
> resFoo <- estimate(res, 200, weighting=weightFoo)
```

References

[1] Kilian Bartholomé, Clemens Kreutz, and Jens Timmer. Estimation of gene induction enables a Relevance-Based ranking of gene sets. *Journal of Computational Biology*, 16(7):959–967, 2009.

[2] David S. Johnson et al. Systematic evaluation of variability in ChIP-chip experiments using predefined DNA targets. *Genome Research*, 18(3):393–403, March 2008.

[3] Gordon K. Smyth. Limma: linear models for microarray data. In R. Gentleman, V. Carey, S. Dudoit, R. Irizarry, and W. Huber, editors, *Bioinformatics and Computational Biology Solutions using R and Bioconductor*, pages 397–420. Springer, New York, 2005.

Session information

- R version 3.2.2 Patched (2015-10-08 r69496), `x86_64-apple-darwin10.8.0`
- Base packages: base, datasets, graphics, grDevices, methods, stats, utils
- Other packages: fdrtool 1.2.15, les 1.20.0, limma 3.26.0
- Loaded via a namespace (and not attached): bitops 1.0-6, boot 1.3-17, caTools 1.17.1, gdata 2.17.0, gplots 2.17.0, gtools 3.5.0, KernSmooth 2.23-15, RColorBrewer 1.1-2, tools 3.2.2