Analysis of genomic arrays using quantile smoothing

Jan Oosting, Paul Eilers & Renee Menezes

Package *quantsmooth*.

October 24, 2023

Contents

Introduction

Genomic arrays give a detailed picture of deletions and amplifications along chromosomes. If changes in copy numbers occur, we expect these to be visible in segments that cover multiple probes, because fragments of chromosomes are generally affected. The spatial information can be used to reduce noise and increase the reliability of detecting changes.

Using spatial information means that some form of smoothing is being applied. But classical methods are not very helpful here: they blur the jumps that occur at the sudden changes of copy numbers and they round, rather than flatten the segments between the jumps.

One alternative approach is to model the data explicitly as a series of segments, with unknown boundaries and unknown heights. These have to be estimated from the data.

We emphasize visualization instead of breakpoint detection and present a smoothing method inspired by penalized least squares (Eilers, 2003). We use the L1 norm, the sum of absolute values, both in the measure of fit and in the roughness penalty. This leads to a large but sparse linear program, which can be solved efficiently with an interior point algorithm. When combined with 0/1 weights, the penalty makes smooth interpolation of left-out observations trivial, allowing elegant and efficient cross-validation.

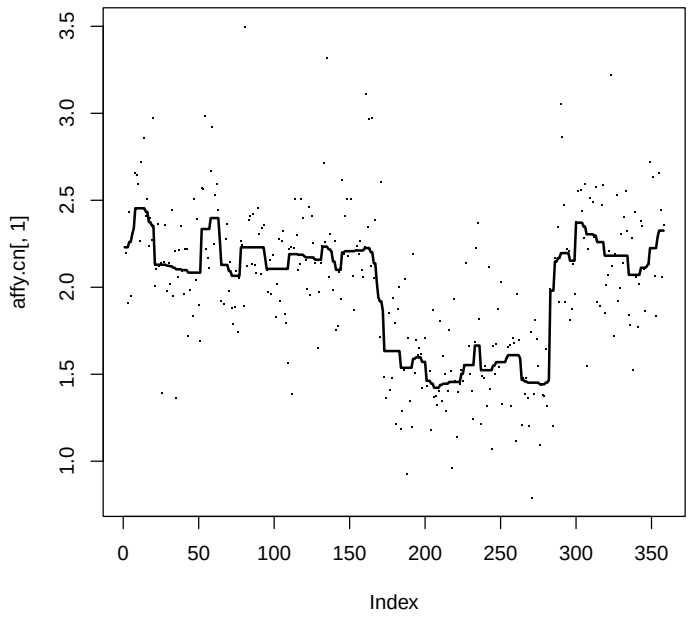
1 Usage

Throughout the examples the same example data are used. Genomic profiles of two tumors were examined using Affymetrix 10K SNP genechip arrays, Illumina Golden Gate Linkage Panel IV SNP arrays, and home spotted 1 mb spaced BAC

arrays. Chromosome 14 was selected to demonstrate the package on an affected chromosome.

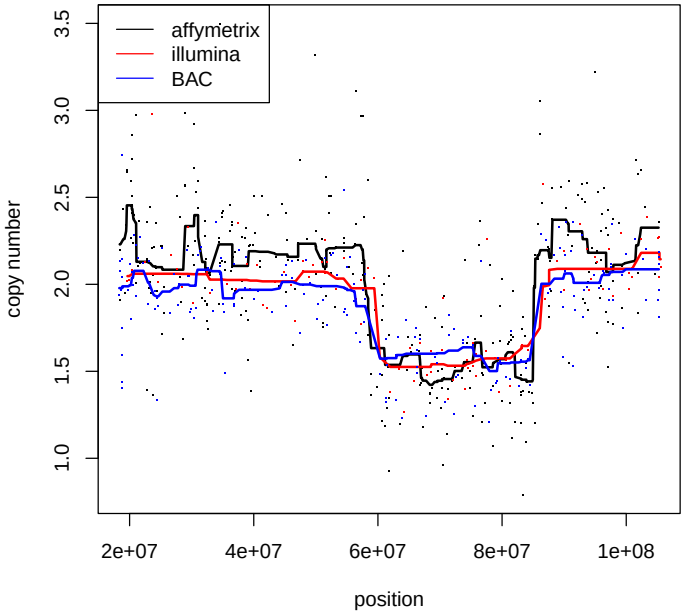
A simple way to show the effect of the smoother is to plot the raw data together with the smoothed line.

```
> library(quantsmooth)
> data(chr14)
> plot(affy.cn[,1],pch=".")
> lines(quantsmooth(affy.cn[,1]),lwd=2)
```



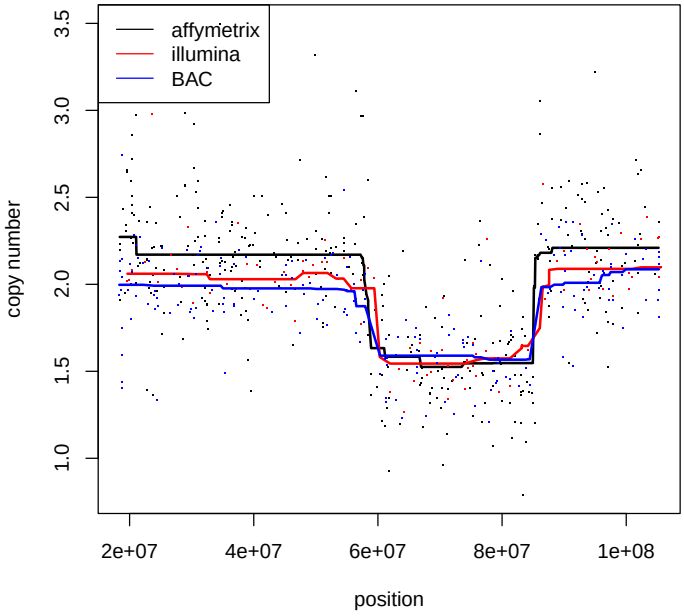
To compare the 3 methods the data can be plotted using the vectors with the chromosomal positions of the probes

```
> plot(affy.pos,affy.cn[,1],ylab="copy number",xlab="position",pch=".")
> lines(affy.pos,quantsmooth(affy.cn[,1]),lwd=2)
> points(bac.pos,bac.cn[,1],col="red",pch=".")
> lines(bac.pos,quantsmooth(bac.cn[,1]),col="red",lwd=2)
> points(ill.pos,ill.cn[,1],col="blue",pch=".")
> lines(ill.pos,quantsmooth(ill.cn[,1]),col="blue",lwd=2)
> legend("topleft",legend=c("affymetrix","illumina","BAC"),col=c("black","red","blue"),lty
```



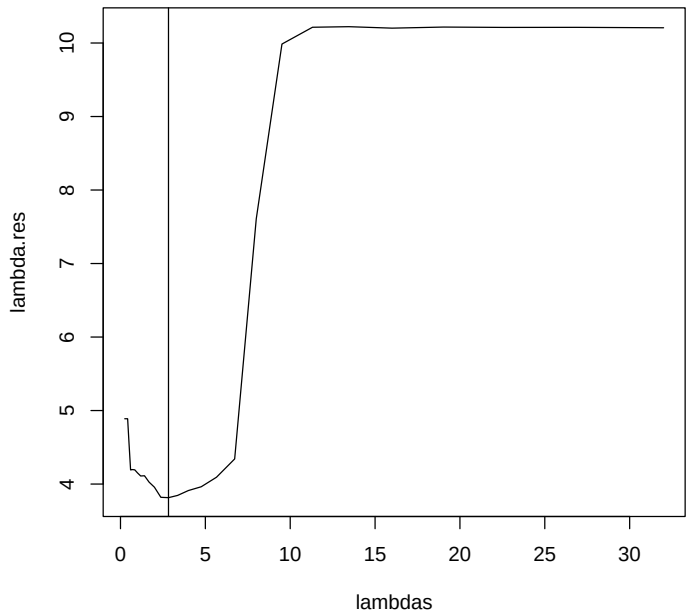
Inspection of this plots shows that the behaviour of the 3 smoothed lines is quite different, i.e. the line for affymetrix is more erratic than for the other two. This can be caused by a difference in the number of probes for this chromosome, or the fact that the variability for the raw affymetrix data is higher. To compensate for the first factor it is possible to adapt the smoothing parameter to the number of probes under investigation.

```
> lambda.divisor<-50
> plot(affy.pos,affy.cn[,1],ylab="copy number",xlab="position",pch=".")
> lines(affy.pos,quantsmooth(affy.cn[,1],smooth.lambda=length(affy.pos)/lambda.divisor),lw
> points(bac.pos,bac.cn[,1],col="red",pch=".")
> lines(bac.pos,quantsmooth(bac.cn[,1],smooth.lambda=length(bac.pos)/lambda.divisor),col="
> points(ill.pos,ill.cn[,1],col="blue",pch=".")
> lines(ill.pos,quantsmooth(ill.cn[,1],smooth.lambda=length(ill.pos)/lambda.divisor),col="
> legend("topleft",legend=c("affymetrix","illumina","BAC"),col=c("black","red","blue"),lty
```



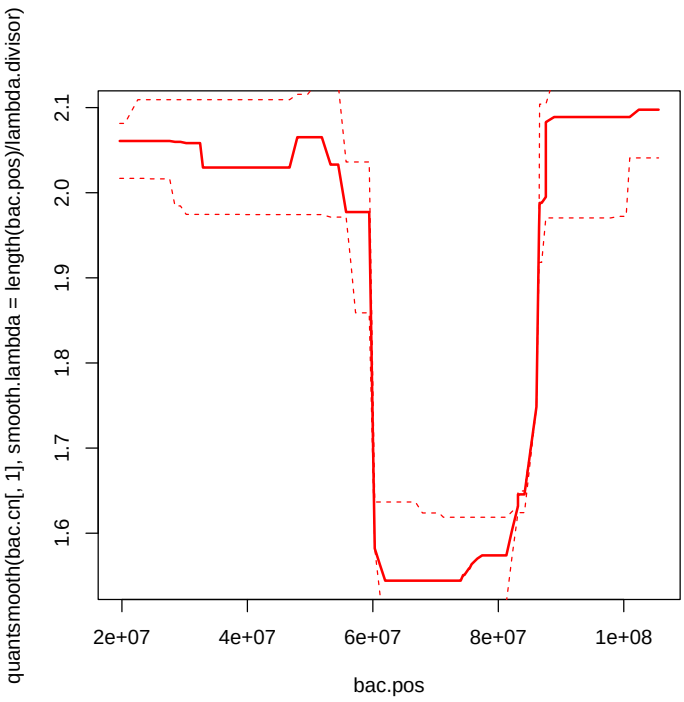
Another method to determine the smoothing parameter is to use cross validation

```
> lambdas<-2^seq(from=-2,to=5,by=0.25)
> lambda.res <- rep(NA, length(lambdas))
> for (lambda in 1:length(lambdas)) lambda.res[lambda] <- quantsmooth.cv(bac.cn[,1], lambd
> plot(lambdas,lambda.res,type="l")
> abline(v=lambdas[which.min(lambda.res)])
```



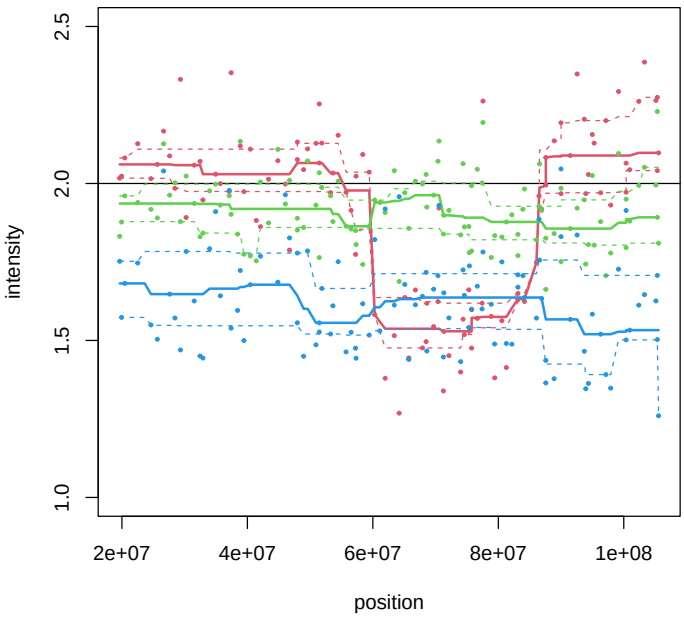
Quantile smoothing can show the variability of the data by also plotting other quantiles besides the median

```
> plot(bac.pos,quantsmooth(bac.cn[,1],smooth.lambda=length(bac.pos)/lambda.divisor),col="r
> lines(bac.pos,quantsmooth(bac.cn[,1],smooth.lambda=length(bac.pos)/lambda.divisor,tau=0.
> lines(bac.pos,quantsmooth(bac.cn[,1],smooth.lambda=length(bac.pos)/lambda.divisor,tau=0.
```



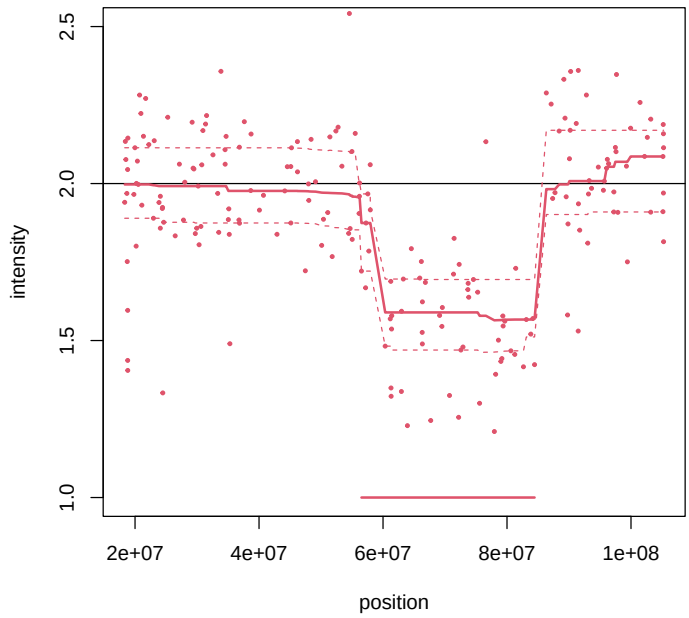
The function plotSmoothed can be used to do this all with 1 command

```
> plotSmoothed(bac.cn,bac.pos,ylim=c(1,2.5),normalized.to=2,smooth.lambda=length(bac.pos)/
```



To identify genomic regions that contain losses or gains also these functions can be used.

```
> plotSmoothed(ill.cn[,1],ill.pos,ylim=c(1,2.5),normalized.to=2,smooth.lambda=length(ill.p
> res<-getChangedRegions(ill.cn[,1],ill.pos,normalized.to=2,interval=0.5)
> segments(res[,"start"],1.0,res[,"end"],1.0,col=2,lwd=2)
```



2 Session Information

The version number of R and packages loaded for generating the vignette were:

- R Under development (unstable) (2023-10-22 r85388), `x86_64-pc-linux-gnu`
- Locale: `LC_CTYPE=en_US.UTF-8`, `LC_NUMERIC=C`, `LC_TIME=en_US.UTF-8`, `LC_COLLATE=en_US.UTF-8`, `LC_MONETARY=en_US.UTF-8`, `LC_MESSAGES=en_US.UTF-8`, `LC_PAPER=en_US.UTF-8`, `LC_NAME=C`, `LC_ADDRESS=C`, `LC_TELEPHONE=C`, `LC_MEASUREMENT=en_US.UTF-8`, `LC_IDENTIFICATION=C`
- Time zone: `America/New_York`
- TZcode source: `system (glibc)`
- Running under: `Ubuntu 22.04.3 LTS`
- Matrix products: default

- BLAS: `/home/biocbuild/bbs-3.19-bioc/R/lib/libRblas.so`
- LAPACK: `/usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.10.0`
- Base packages: base, datasets, graphics, grDevices, grid, methods, stats, utils
- Other packages: quantreg 5.97, quantsmooth 1.69.0, SparseM 1.81
- Loaded via a namespace (and not attached): compiler 4.4.0, lattice 0.22-5, MASS 7.3-60.1, Matrix 1.6-1.1, MatrixModels 0.5-2, splines 4.4.0, survival 3.5-7, tools 4.4.0