

# TurboNorm: A fast scatterplot smoother with applications for microarray normalization

Chantal van Leeuwen and Maarten van Iterson  
Center for Human and Clinical Genetics,  
Leiden University Medical Center, The Netherlands  
Package *TurboNorm*, version 1.6.2

Modified: 19 Februari, 2013. Compiled: February 19, 2013

## Contents

1	Introduction	1
2	Smoothing using piecewise constant P-splines	2
3	Normalization of single- and two-colour data	3
4	Normalization of array-based DNA methylation data	4
5	Details	6

## 1 Introduction

```
■echo=false■= options(width=60) @
```

This vignette show how piecewise constant P-splines [1] can be used for normalization of either single- or two-colour data. The `pspline()`-function can be used for two-colour data objects of type `RGList` and `MarrayRaw` from respectively from *limma* [2] and from the package *marray*. For single colour microarray data wrapper functions are writting based on the *affy* [3] functions `normalize.loess()` and `normalize.AffyBatch.loess()` namely `normalize.pspline()` and `normalize.AffyBatch.pspline()`. Also a `panel`-function, `panel.pspline()`, is available for adding the smoothed curve to *lattice* [4] graphic panels.

The P-spline smoother introduced by Eilers and Marx [1] is a combination of B-splines with a difference penalty on the regression coefficients. P-splines belong to the family of penalized splines using B-spline basis functions, where the penalization is on the curvature of the smoothed function. For the P-splines of Eilers and Marx [1], a discrete approximation to the integrated squared second derivative of the B-splines is made. This results in an easy-to-construct penalty matrix, and the resulting band-diagonal system of equations can be efficiently solved. Using piecewise constant B-splines as a basis makes the construction of the B-spline basis even easier. The resulting linear system of equations can be solved either using a QR decomposition or a Cholesky decomposition [5].

Additionally to the P-spline smoother proposed by [1] we introduce a weighted P-spline smoother. The weighted P-spline smoother leads to the following system of equations:

$$(X'WX + \lambda D'D)\hat{\beta} = X'W\mathbf{y}, \quad (1)$$

where  $X$  is the B-spline basis matrix (with  $X'$  its transpose),  $W$  is a diagonal matrix of weights,  $D$  is a matrix operator for the second-order differences and  $\mathbf{y}$  represents the vector of observations. The value of penalty parameter,  $\lambda$ , can be determined by cross-validation, for example. The original P-spline smoother of Eilers and Marx [1] has  $W$ , the identity matrix. When piecewise constant basis functions are used, both  $X'WX$  and  $X'W\mathbf{y}$  become diagonal matrices, and can be constructed very efficiently [6]. The regression coefficients of the weighted P-spline smoother are now given by:

$$\hat{\beta} = (X'WX + \lambda D'D)^{-1} X'W\mathbf{y}. \quad (2)$$

See for a detailed description of the method and several applications van Iterson *et al.* [7].

## 2 Smoothing using piecewise constant P-splines

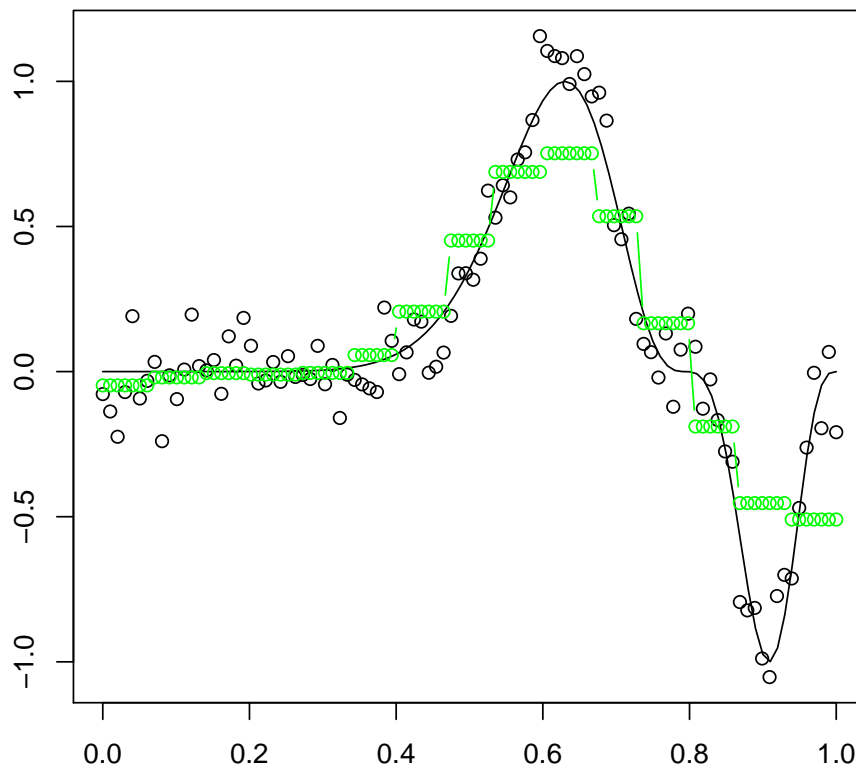
The main workhorse of the package is the function `turbotrend()`. Given data the function returns an object containing the smoothed values and some additional information like, effective degrees of freedom, optimized penalty value,  $\lambda$ , and the generalized cross-validation error at the optimal penalty value.

The following toy example shows the use of the `turbotrend()`. First we load the library and generate some data:

```
> library(TurboNorm)
> funky <- function(x) sin(2*pi*x^3)^3
> m <- 100
> x <- seq(0, 1, length=m)
> y <- funky(x) + rnorm(m, 0, 0.1)
```

Next we plot the data and the underlying function that generated the data together with the smoothed curves based on the original piecewise constant B-spline basis.

```
> plot(x, y, type='p', xlab="", ylab="")
> curve(funky, add=TRUE)
> fitOrig <- turbotrend(x, y, n=15, method="original")
> lines(fitOrig, col="green", type='b', pch=1)
```



In order to get some more detail on the regression parameters a `show`-method is implemented.

```
> fitOrig
```

Call:

```
turbotrend(x = x, y = y, n = 15, method = "original")
```

Effective degrees of freedom: 6.282501

Number of bins: 15

Penalty value: 8.606445

GCV : 0.05784094

### 3 Normalization of single- and two-colour data

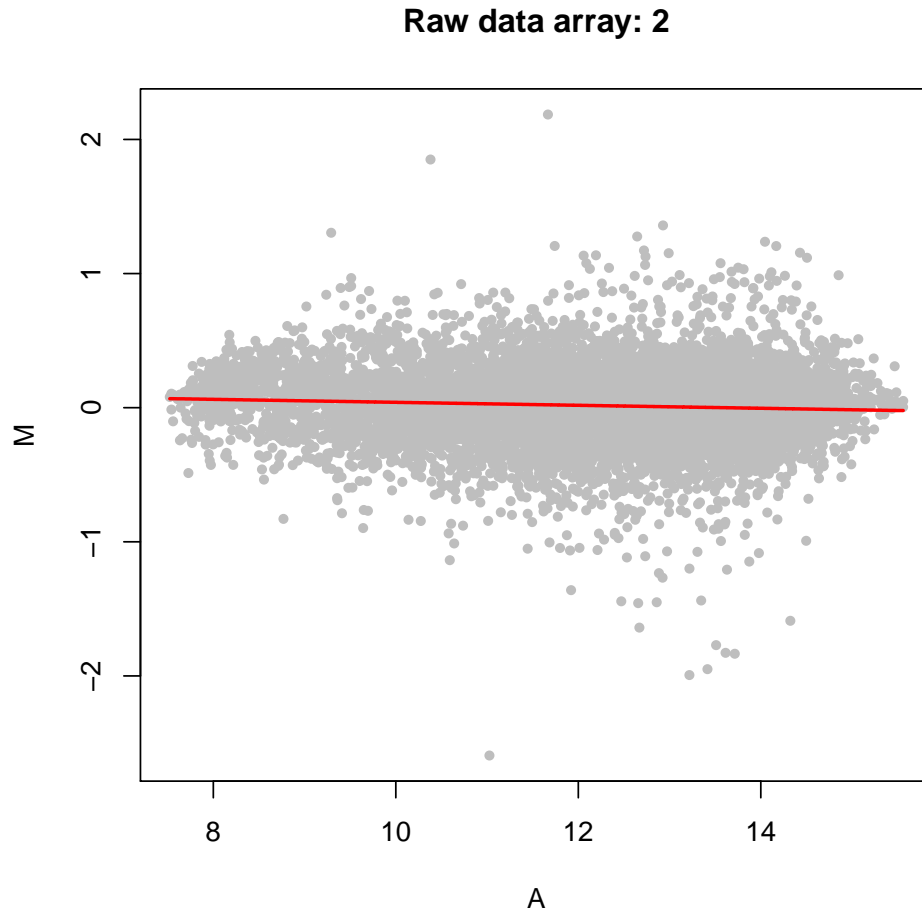
For single colour microarray data normalization the following functions are available `normalize.pspline()` and `normalize.AffyBatch.pspline()` these functions are based on functions for normalization from the *affy* package.

The `pspline()`-function can be used for normalization of two-colour microarray data. The data input is either an object of type `RGList` as defined in the package *limma* or an object of type `MarrayRaw` defined in

the package *marray*. The `pspline()`-function recognizes the type of the object and returns the normalized object of the same type, i.e. `MAList` and `MarrayNorm`.

Here is an example code using the `swirl`-data from *marray*. Using the option `showArrays=2` the smoothed curve is plotted together with the data in a MA plot for array 2 (by default no plot is shown).

```
> library(marray)
> data(swirl)
> x <- pspline(swirl, showArrays=2, pch=20, col="grey")
```



## 4 Normalization of array-based DNA methylation data

Here we show how a weighted normalization can be performed. This is especially useful for array-based DNA methylation data, where there is large number of differential methylation expected.

Using `data(methylation)` a random subset of the data of one of the cell lines described in the paper by van Iterson *et al.* [7] is loaded as an `RGList`. The element `weights` of the `RGList` contains the subset of invariant fragments, those without methylation-sensitive restriction sites, as a logical matrix where each column represents an array those fragments that are part of the subset are `TRUE` and those that are not `FALSE`. The data dependent weight is in this example approximately 250.

```

> library(TurboNorm)
> data(methylation)
> indices <- methylation$weights[,1]
> weights <- rep(1, length(indices))
> weights[indices] <- length(indices)/sum(indices)
> MA <- normalizeWithinArrays(methylation, method="none", bc.method="none")
> labels <- paste("NMB", c("(untreated)", "(treated)"))
> labels <- paste(rep(c("Raw"), each=2), labels)

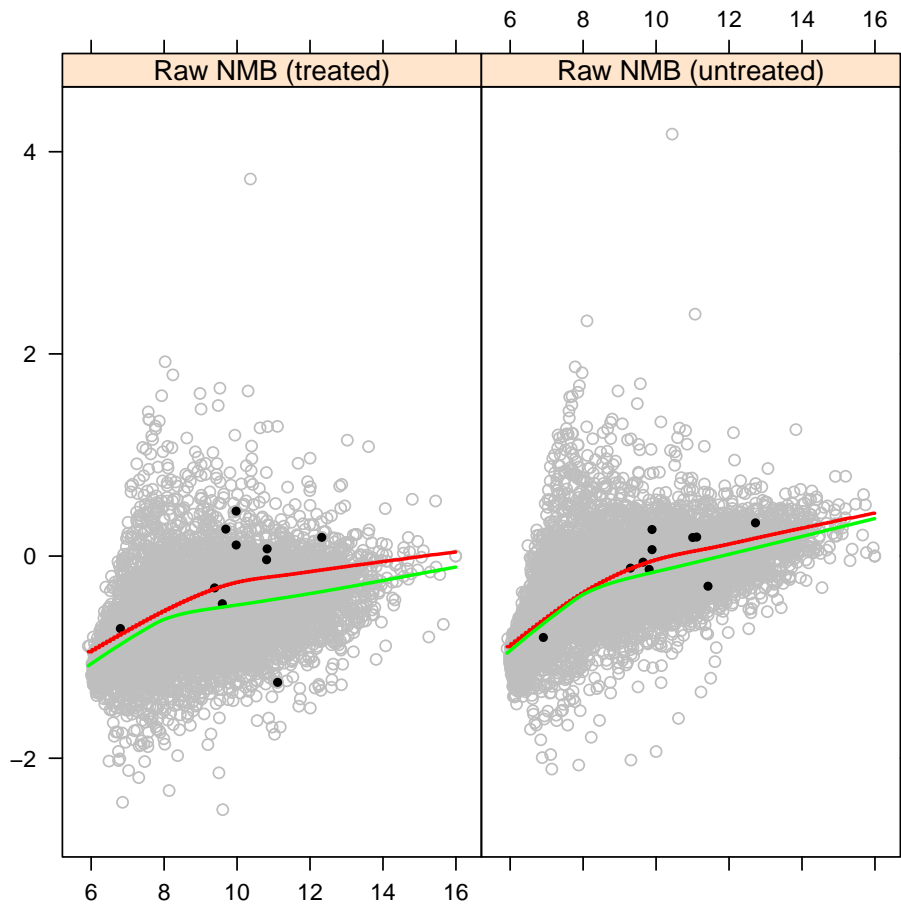
```

First we transform the intensities to M- and A-values without background correction and then the normalization is performed both weighted P-spline and ordinary lowess using *limma*. Now we use the *lattice* in order to illustrate the difference. We highlight the invariant subset in black.

```

> data <- data.frame(M=as.vector(MA$M),
+                   A=as.vector(MA$A),
+                   Array=factor(rep(labels, each=nrow(MA$A)), levels=rev(labels)))
> library(lattice)
> print(xyplot(M~A|Array, xlab="", ylab="", data=data, type='g',
+             panel = function(x, y)
+             {
+                 panel.xyplot(x, y, col="grey")
+                 lpoints(x[indices], y[indices], pch=20, col="black")
+                 panel.pspline(x, y, weights = weights, col="red", lwd=2)
+                 panel.loess(x, y, col="green", lwd=2)
+             }
+             ))

```



This example also shows how the `panel.pspline()`-function can be used. The smoothed curve obtained by the P-spline smoother can be added to *lattice* graphics.

## 5 Details

This document was written using:

- R version 2.15.2 (2012-10-26), i386-w64-mingw32
- Locale: LC\_COLLATE=C, LC\_CTYPE=English\_United States.1252, LC\_MONETARY=English\_United States.1252, LC\_NUMERIC=C, LC\_TIME=English\_United States.1252
- Base packages: base, datasets, grDevices, graphics, methods, stats, utils
- Other packages: Biobase 2.18.0, BiocGenerics 0.4.0, TurboNorm 1.6.2, convert 1.34.0, lattice 0.20-13, limma 3.14.4, marray 1.36.0
- Loaded via a namespace (and not attached): BiocInstaller 1.8.3, affy 1.36.1, affyio 1.26.0, grid 2.15.2, preprocessCore 1.20.0, tools 2.15.2, zlibbioc 1.4.0

## References

- [1] P.H.C. Eilers and B.D. Marx. Flexible smoothing with B-splines and penalties. *Statistical Science*, 11(2):89–121, 1996.
- [2] Gordon K. Smyth. Limma: linear models for microarray data. In R. Gentleman, V. Carey, S. Dudoit, and W. Huber R. Irizarry, editors, *Bioinformatics and Computational Biology Solutions using R and Bioconductor*, pages 397–420. Springer, New York, 2005.
- [3] B.M. Bolstad, R.A. Irizarry, M. Astrand, and T.P. Speed. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19(2):185–93, 2003.
- [4] D. Sarkar. *lattice: Lattice Graphics*, 2009. R package version 0.17-26.
- [5] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. New York:Springer-Verlag, 2001.
- [6] P.H.C. Eilers. Fast computation of trends in scatterplots. *Kwantitatieve Methoden*, 71:38–45, 2004.
- [7] M. van Iterson. in preparation, 2010.