Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. http://www.jstatsoft.org/

EbayesThresh: R Programs for Empirical Bayes Thresholding

Iain M. Johnstone
Stanford University

Bernard W. Silverman
St Peter's College, Oxford

Abstract

Suppose that a sequence of unknown parameters is observed subject to independent Gaussian noise. The **EbayesThresh** package in the S language implements a class of Empirical Bayes thresholding methods that can take advantage of possible sparsity in the sequence, to improve the quality of estimation.

The prior for each parameter in the sequence is a mixture of an atom of probability at zero and a heavy-tailed density. Within the package, this can be either a Laplace (double exponential) density or else a mixture of normal distributions with tail behavior similar to the Cauchy distribution. The mixing weight, or sparsity parameter, is chosen automatically by marginal maximum likelihood. If estimation is carried out using the posterior median, this is a random thresholding procedure; the estimation can also be carried out using other thresholding rules with the same threshold, and the package provides the posterior mean, and hard and soft thresholding, as additional options.

This paper reviews the method, and gives details (far beyond those previously published) of the calculations needed for implementing the procedures. It explains and motivates both the general methodology, and the use of the **EbayesThresh** package, through simulated and real data examples.

When estimating the wavelet transform of an unknown function, it is appropriate to apply the method level by level to the transform of the observed data. The package can carry out these calculations for wavelet transforms obtained using various packages in R and S-PLUS. Details, including a motivating example, are presented, and the application of the method to image estimation is also explored.

The final topic considered is the estimation of a single sequence that may become progressively sparser along the sequence. An iterated least squares isotone regression method allows for the choice of a threshold that depends monotonically on the order in which the observations are made. An alternative possibility, also discussed in detail, is a particular parametric dependence of the sparsity parameter on the position in the sequence.

Keywords: adaptive estimation, curve estimation, data mining, image estimation, nonlinear smoothing, marginal maximum likelihood, sparsity, wavelets.

1. Introduction

1.1. Background

There are many statistical problems where the object of interest is a sequence of parameters μ_i on each of which we have a single observation X_i subject to noise, so that

$$X_i = \mu_i + \epsilon_i \tag{1}$$

where the ϵ_i are $N(0,1)$ random variables.

Problems of this kind arise, for example, in astronomical and other image processing contexts, in data mining, in model selection, and in function estimation using wavelets and other dictionaries. See, for example Johnstone and Silverman (2004) for further discussion. In many practical contexts, the sequence μ_i may be sparse, in some sense, and the method implemented in the **EbayesThresh** package takes advantage of possible sparsity in order to obtain more efficient estimators of the sequence μ_i.

A natural approach that can make use of sparsity is *thresholding*: if the absolute value of a particular X_i exceeds some threshold t then it is taken to correspond to a a nonzero μ_i which is then estimated, most simply by X_i itself. If $|X_i| < t$ then the coefficient $|\mu_i|$ is estimated to be zero. The quality of estimation is sensitive to the choice of threshold, with the best choice being dependent on the problem setting. In general terms, "sparse" signals call for relatively high thresholds (3, 4, or even higher) while "dense" signals might demand choices of 2 or even lower.

In essence, the Empirical Bayes approach implemented in the **EbayesThresh** package is a thresholding method with a threshold estimated from the data. Both theoretical and practical considerations, demonstrated in the papers Johnstone and Silverman (2004, 2005) and in the present paper, show that the Empirical Bayes approach has excellent adaptivity properties, in particular by adjusting stably to the sparsity or otherwise of the underlying signal, by choosing an appropriate threshold from the data.

The present paper reviews the approach of Johnstone and Silverman (2004, 2005) and introduces the R package **EbayesThresh**. In addition, we provide algorithmic calculations and details far beyond those given in the original papers. We also consider two novel extensions of the method to the case where the thresholds vary with i in different ways, by increasing monotonically or by being of a particular parametric form.

In addition to its scientific contribution, the paper is intended to provide a tutorial introduction to the package. The full documentation of the routines in the package is provided as an appendix.

The **EbayesThresh** package is implemented in R but the programs have been written to work as far as possible in S-PLUS as well. The source code and user manual are available at the CRAN archive `www.r-project.org` as the package **EbayesThresh** (Silverman 2004) and other documentation is available from `www.bernardsilverman.com`. A MATLAB implementation has been carried out by Antoniadis, Jansen, Johnstone, and Silverman (2004).

1.2. Examples

Before explaining the algorithms in detail, we consider two examples that give a feeling for the way that the method works in practice. In both the cases considered, the estimate is

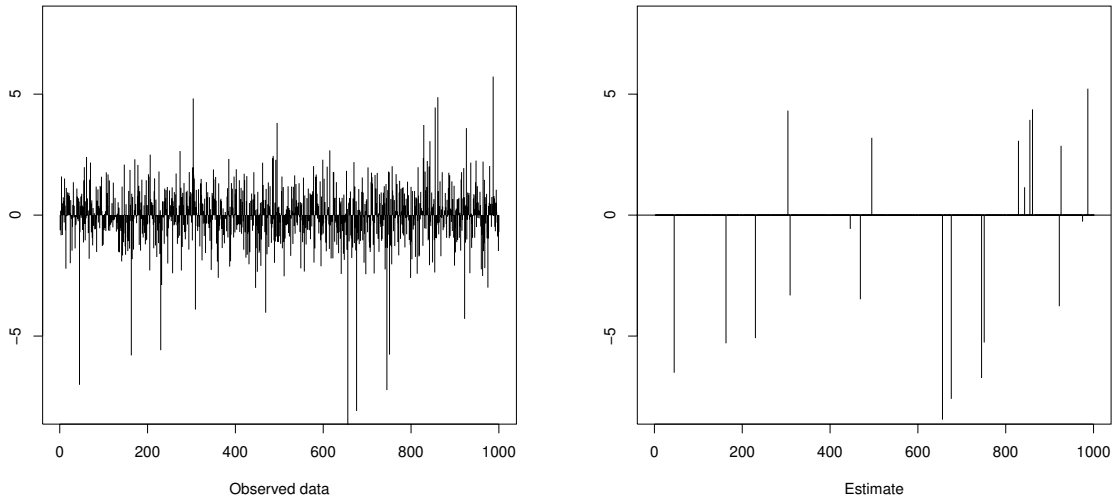

Figure 1: Simulated data `x` and estimate `ebayesthresh(x, sdev=1)` for sparse example. Only 25 of the 1000 underlying parameters μ_i are nonzero.

obtained from the data simply as `ebayesthresh(x, sdev=1)` using the main routine in the package, and supplying the actual value of the standard deviation of the data. As will be explained later, the default form of thresholding is not exactly hard thresholding, but the basic principle is the same.

The first example is of data with a relatively sparse mean vector of length 1000, with 25 nonzero entries uniformly distributed on $(-7, 7)$. The data are generated in R by

```
set.seed(1)
x <- rnorm(1000) + sample(c( runif(25,-7,7), rep(0,975)))
```

and are plotted in Figure 1, together with the estimate `ebayesthresh(x, sdev=1)`. It can be seen that relatively stringent thresholding has been applied to the data; the numerical value of the threshold chosen by the procedure turns out to be 2.99, in the sense that any data point within 2.99 standard deviations of zero is presumed to be pure noise.

The corresponding procedure was carried out for a data set constructed in exactly the same way containing 250 nonzero mean values, a much denser signal. This time the numerical value of the threshold was only 1.67, and so it was much easier for an observation with zero mean to be considered as containing some signal. The results are shown in Figure 2.

Further insight into the comparison of these examples is given in Figure 3. For the sparse signal, the high threshold has the effect that all but 4 of the 975 zero parameters are estimated to be zero. The other 971 are estimated perfectly. However 9 of the 25 nonzero parameters are estimated to zero, thereby presumably incurring slightly more error than if they were not thresholded. In the case of the dense signal, a far higher proportion of the zero parameters, 89 out of 750, are estimated to be nonzero, but the proportion of nonzero parameters incorrectly classified as zeroes is lower: 51 out of 250. The way in which the empirical Bayes method automatically adjusts this tradeoff will be discussed more systematically in Section 3.2 below.

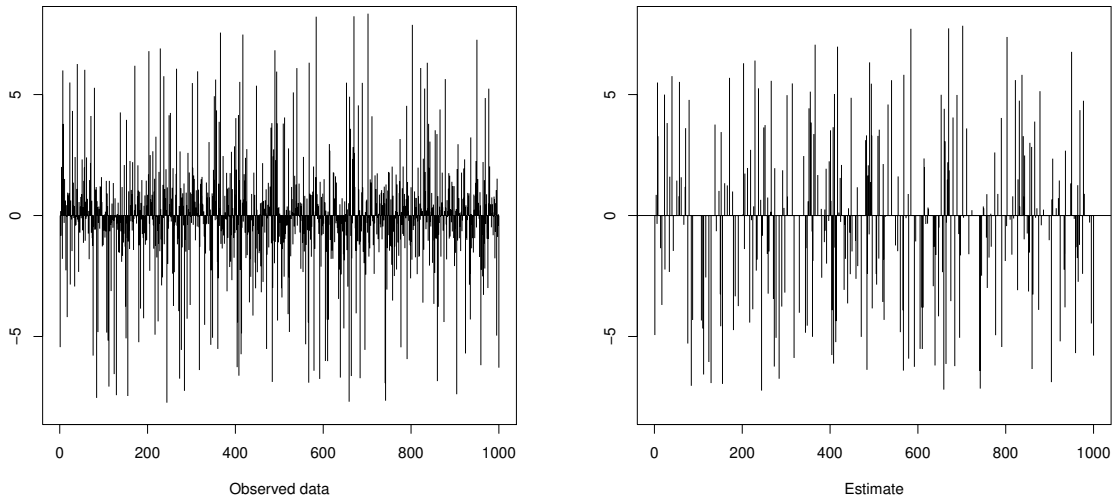

Figure 2: Simulated data `x` and estimate `ebayesthresh(x, sdev=1)` for dense example. In this case 250 of the 1000 underlying parameters μ_i are nonzero.

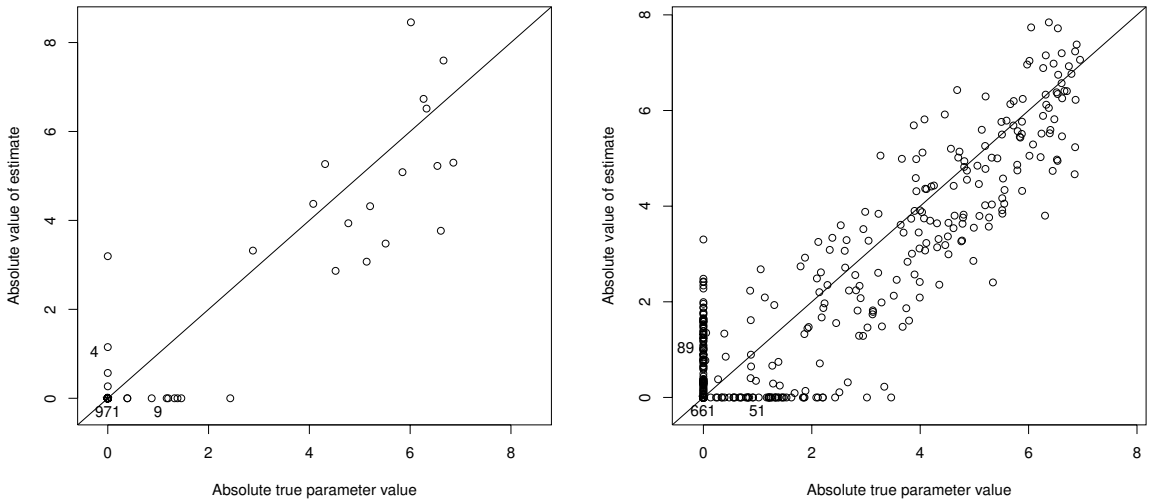

Figure 3: Comparison of the performance of the estimation for the sparse and dense examples. In each figure, the absolute value of the estimates is plotted against the absolute value of the corresponding parameters. In the left panel, there are 971 parameters correctly estimated to be zero, 4 zero parameters estimated to be nonzero, and 9 nonzero parameters estimated to be zero. The corresponding quantities for the dense signal considered in the right panel are 661, 89 and 51.

1.3. Brief overview of the method and the master routine

Very briefly, the main aspects of our method are as follows; they will be discussed further in Section 2.

- A Bayesian model is used for the parameters μ_i. Under this model, each μ_i is zero with probability $(1-w)$, while, with probability w, μ_i is drawn from a symmetric heavy-tailed density γ.
- The mixing weight w is the key parameter in the prior. It is chosen automatically from the data, using a marginal maximum likelihood approach, and then substituted back into the Bayesian model.
- Estimation within the Bayesian model is a thresholding procedure, and the choice of w is equivalent to a choice of threshold $t(w)$. The method uses this data-based threshold in estimating the underlying vector of parameters from the data.

Most users will only need to use the master routine `ebayesthresh`. This takes the vector `x` and returns an estimate of the parameter values μ_i, which will be obtained by simply carrying out

```
mu <- ebayesthresh(x)
```

The fuller syntax of the routine is

```
ebayesthresh(x, prior = "laplace", a = 0.5, bayesfac = FALSE,
       sdev = NA, verbose = FALSE, threshrule = "median")
```

and reviewing the optional arguments gives an overview of some of the topics discussed in more detail below. Further details are given in the help file for the routine.

The argument `prior` specifies the density $\gamma(u)$, the default choice for which is a double exponential, or Laplace, density $\frac{1}{2}a \exp(-a|u|)$. The parameter a is given by the argument `a`. In Section 2.1, this choice of prior is discussed further, together with an alternative possibility.

The arguments `bayesfac` and `threshrule` determine the exact way in which the data are processed once the threshold has estimated. For most practical purposes their default values can be used, but details of other possible approaches are given in Sections 2.2 and 2.3.

The argument `sdev` gives the standard deviation of the noise $X_i - \mu_i$ in the data; the default is for this to be estimated from the observed data, from the median absolute value of the X_i. The motivation for this is that even if the sequence μ_i is only reasonably sparse, the median absolute value will not be affected by those observations that have nonzero means μ_i. Clearly some care may be needed, if there is a possibility that the signal is very far from sparse. If a numerical value of the standard deviation is known, or is estimated by other means, then it can be supplied as the value of `sdev`.

Finally, the argument `verbose`, if set to `TRUE`, causes the routine to produce a list containing a number of different aspects of the thresholding, such as the numerical value of the threshold used, the estimated standard deviation, and so on. This is most useful for research purposes rather than for the actual processing of data.

2. Description of the method

In this section, we describe and explain the various aspects of the method. For reasons of clarity, full details of the various algorithms and calculations are not given in this section, but they are set out, after the more practical part of the paper, in Section 6.

2.1. The Bayesian model

In this discussion, we assume throughout that the observations $X_i \sim N(\mu_i, 1)$. If the observations have variance equal to σ^2 rather than 1, then we renormalize the data by dividing by σ, and then multiply the resulting estimates of the means by σ.

Within a Bayesian context, the notion of sparsity is naturally modeled by a suitable prior distribution for the parameters μ_i. We model the μ_i as having independent prior distributions each given by the mixture

$$f_{\text{prior}}(\mu) = (1-w)\delta_0(\mu) + w\gamma(\mu). \tag{2}$$

The nonzero part of the prior, γ, is assumed to be a fixed unimodal symmetric density.

We concentrate on two particular possibilities for the function γ. Setting `prior="laplace"`, the default, uses the Laplace density with scale parameter $a > 0$

$$\gamma_a(u) = \tfrac{1}{2} a \exp(-a|u|).$$

The default value of the argument `a`, the parameter a, is 0.5, but any other value can be set by the user.

Another possibility for γ is obtained by setting `prior="cauchy"`. (The parameter `a` is then ignored.) The density γ for μ is specified by the mixture

$$\mu|\Theta = \theta \sim N(0, \theta^{-1} - 1) \text{ with } \Theta \sim \text{Beta}(\tfrac{1}{2}, 1). \tag{3}$$

This yields the density

$$\gamma(u) = (2\pi)^{-1/2}\{1 - |u|\tilde{\Phi}(|u|)/\phi(u)\}, \tag{4}$$

which has tails that decay as u^{-2}, the same weight as those of the Cauchy distribution. We refer to the density (4) as the *quasi-Cauchy* density. Some further discussion is provided in Section 2.3 of Johnstone and Silverman (2004). The density is one of a family whose tails decay at polynomial rates; the main motivation for the quasi-Cauchy is its combination of heavy tails with reasonable tractability in the present context. It is the heaviest tailed density satisfying the theoretical assumptions made in Johnstone and Silverman (2004).

For both the Laplace and quasi-Cauchy densities, the procedures we develop are feasible computationally, and this feasibility is exploited within the **EbayesThresh** package. Details of the various calculations are given in Section 6.

2.2. Thresholding rules

Suppose μ has the prior distribution (2) and $X \sim N(\mu, 1)$. We can now find the posterior distribution of μ conditional on $X = x$; see Section 6 for details. Define $\hat{\mu}(x; w)$ to be the median of this posterior distribution; for any fixed w, the estimation rule $\hat{\mu}(x, w)$ will be a

monotonic function of x with the *thresholding property* that there exists $t(w) > 0$ such that $\hat{\mu}(x; w) = 0$ if and only if $|x| \leq t(w)$.

Given a sequence of observations, we can apply the Bayesian procedure separately to each observation X_i to yield an estimate of the corresponding parameter μ_i. The default setting `threshrule="median"` uses the posterior median $\hat{\mu}(X_i; w)$ as this estimate. This is an exact Bayesian procedure if the X_i are independent; if the X_i are not exactly independent then there is some loss of information in the estimation procedure, but if there is not too much dependence then the method will give at least reasonable results.

The posterior median is not the only possible estimation rule once w has been specified; for example one could use the posterior mean $\tilde{\mu}(x; w)$ of μ given $X = x$, and this is obtained by setting `threshrule="mean"`. Another possibility is to determine the threshold $t(w)$ associated with the posterior median with weight w, but then to carry out the actual estimation by hard or soft thresholding with threshold $t(w)$. This is achieved within the package by setting `threshrule` to `"hard"` or `"soft"` respectively.

Finally, if the setting `verbose=TRUE` is used, it may not be appropriate to process the data at all, but only to obtain other aspects of the estimation. Setting `threshrule="none"` will have this effect.

2.3. Choosing the threshold

The key aspect of the empirical Bayes approach is the choice of mixing weight w, or equivalently of threshold $t(w)$. Assume the X_i are independent; we then estimate w by a marginal maximum likelihood approach. Let $g = \gamma \star \phi$, where $\star$ denotes convolution.

The marginal density of the observations X_i will then be

$$(1 - w)\phi(x) + wg.$$

We define the marginal maximum likelihood estimator $\hat{w}$ of w to be the maximizer of the marginal log likelihood

$$\ell(w) = \sum_{i=1}^{n} \log\{(1 - w)\phi(X_i) + wg(X_i)\} \tag{5}$$

subject to the constraint on w that the threshold satisfies $t(w) \leq \sqrt{2 \log n}$. For our priors, the derivative $\ell'(w)$ is a monotonic function of w, so its root is very easily found numerically, since the function g is tractable in each case.

Having used the data once to obtain the estimate $\hat{w}$ by marginal maximum likelihood, we then plug the value $\hat{w}$ back into the prior and then estimate the parameters μ_i by a Bayesian procedure using this value of w, for example as the posterior median $\hat{\mu}(X_i, \hat{w})$. In our implementation the cost of both parts of the procedure is linear in the number of observations considered.

Other parameters of the prior can also be estimated by marginal maximum likelihood. In particular, if the Laplace prior is used with `a=NA`, then the scale parameter a is estimated by defining g_a to be the convolution of $a\gamma(a\,\cdot)$ with the normal density. Then both a and w are estimated by finding the maximum over both parameters of

$$\ell(w, a) = \sum_{i=1}^{n} \log\{(1 - w)\phi(X_i) + wg_a(X_i)\}.$$

The bound $\sqrt{2\log n}$ on the threshold is the so-called *universal threshold* for a sample of size n. As explained on page 445 of Donoho and Johnstone (1994), it is, asymptotically, the maximum absolute value of a sequence of n independent $N(0,1)$ random variables. If the universal threshold is used, then with high probability every zero signal value will be estimated correctly. Therefore, in simple terms, if we wish to take advantage of the possible economy of a signal by thresholding, there is no need to consider thresholds any larger than the universal threshold.

An alternative to the posterior median threshold is the *Bayes factor threshold*, defined as the value $\tau_b(w) > 0$ such that

$$P(\mu \neq 0 | X = \tau_b(w)) = 0.5.$$

Invoking the routine `ebayesthresh` with the argument `bayesfac=TRUE` will use the Bayes factor threshold instead of the posterior median threshold whenever a threshold is calculated explicitly, for example if hard or soft thresholding is used or if the routine is invoked with the argument `verbose=TRUE` to return the threshold value and other statistics of the estimation.

2.4. Wavelet thresholding and other extensions

Though the basic approach is much more widely applicable, our original motivation was function estimation using wavelets. It is typical that the wavelet coefficients of a true signal will be sparse at the fine resolution scales, and dense at the coarser scales. It is therefore desirable to develop threshold selection methods that adapt the threshold level by level, and so our approach in the wavelet case is to apply the Empirical Bayes method separately to each level of the transform. In Section 4 we provide a full discussion, including an explanation and demonstration of the way that wavelet thresholding is implemented within the **EbayesThresh** package. A detailed treatment of the approach, including both theoretical and practical aspects, is given in Johnstone and Silverman (2005).

Another extension, explored in Section 5.1, is to allow the threshold to increase as i increases, reflecting the notion that early μ_i have a reasonably large probability of being nonzero, but that as one proceeds along the sequence nonzero μ_i become rarer. An iterated least squares monotone regression algorithm is available as part of the package makes it possible to estimate the weights and thresholds under these monotonicity conditions.

Alternatively, it is possible to constrain the prior mixing weight to be proportional to some prescribed sequence c_i, subject to the constraint that it remains bounded between some reasonable lower limit and 1. This approach is discussed further in Section 5.2. Algorithmic details of both these additional approaches are given in Section 7.

3. Examples and aspects of the package

In this section, we consider a simple illustrative example, and then an example related to one studied in Johnstone and Silverman (2004). We then go on briefly to explore some other aspects of the methodology and of the **EbayesThresh** package.

3.1. A simple illustrative example

Construct a signal of length 100 with ten values equal to 3.5, and the remaining values 0. Add normal independent noise to the signal and estimate the underlying mean vector using

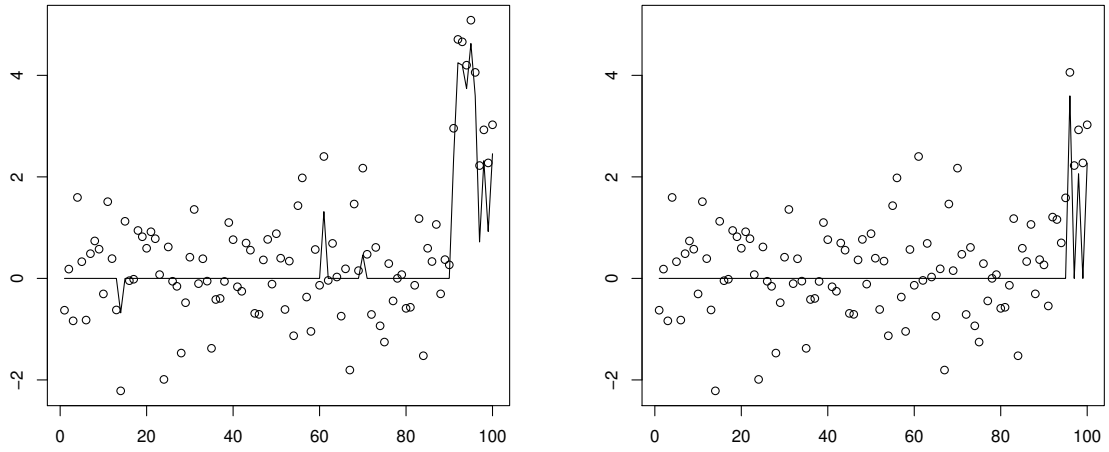

Figure 4: Simulated data, with $N(0, 1)$ noise, from a signal of length 100 which has most of its values zero and the remainder equal to 3.5. The result of applying the routine `ebayesthresh`, with the default options, is shown as a plotted line. Left: first 90 signal values zero; Right: first 95 signal values zero.

`ebayesthresh`. The following code plots the data, and calculates and plots the estimate, joining successive estimated points with lines, as shown in the left panel of Figure 4:

```
set.seed(1)
mu <- c( rep(0, 90), rep(3.5, 10) )
x  <- rnorm(100, mu)
plot(x)
lines(ebayesthresh(x))
```

Most of the ninety zeroes are correctly estimated, though three of them are not estimated to be zero; conversely, the ten values (the last ten to be plotted) that should be equal to 3.5 are all estimated to be nonzero, though there is some shrinkage towards zero. Closer examination of the plot shows that the largest two data points from the first part of the plot are very similar in value to the two smallest data points in the last segment.

The right panel shows the effect of the same procedure on a sparser sample, in which only the last five points have nonzero mean. The adaptivity of the method is demonstrated by the fact that all the zero means are now estimated correctly; the price paid is greater shrinkage and smoothing of the nonzero values, to the point that two of them are estimated to be zero.

This example is demanding of the method, because the sample size is relatively small, and the value of the nonzero parameters, 3.5, is in the range where individual observations with negative error can easily be confused with observations from a large sample with zero mean.

3.2. Varying sparsity in a larger sample

In Figure 1 of Johnstone and Silverman (2004) a number of artificial images of varying degrees

of sparseness are considered. The images and data are plotted there as 100×100 images, but for simplicity we consider them as a single vector of length 10000. To construct a sparse vector and then find the empirical Bayes threshold, the following code can be used in R:

```
m  <- 50
mu <- sample( c( runif(m,-5,5), rep(0,10000-m) ) )
x  <- rnorm(10000, mu)
tt <- tfromx(x)
```

The routine `tfromx` finds the empirical Bayes choice of threshold, assuming the noise standard deviation to be 1. Once the threshold has been found, one can hard threshold the original data to find an estimate of μ by using the `threshld` routine:

```
muhat <- threshld(x, tt)
```

Note that the same estimate `muhat` can be obtained by the single call `ebayesthresh(x, sdev=1, threshrule="hard")`, but this does not yield the threshold explicitly unless the option `verbose=TRUE` is used.

We compare the chosen threshold with the threshold that yields the smallest mean square error. The following routine constructs a signal of length 10000 with `m` nonzero values uniformly distributed on $(-5, 5)$. It returns the empirical Bayes choice `tebayes` of threshold and, by comparison, the threshold `tbest` that achieves the minimum mean square error for hard thresholding. The corresponding summed square errors `rebayes` and `rbest` are also returned. The two routines used from the **EbayesThresh** package are the routine `tfromx` to find the empirical Bayes threshold, and the routine `threshld` which applies hard thresholding.

```
ebdem1 <- function(m)
{
  set.seed(1)
  zz <- rnorm(10000)
  mu <- c( runif(m,-5,5), rep(0,10000-m) )
  x  <- mu + zz
  tt <- tfromx(x)
  tvec <- seq(from = 0, to = 5, by = 0.1)
  rvec <- rep(NA, 51)
  for ( j in (1:51)) rvec[j] <- sum( (threshld(x, tvec[j]) - mu )^2 )
  reb <- sum( (threshld(x, tt) - mu )^2 )
  rbest <- min(rvec)
  tbest <- mean ( tvec[rvec==rbest] )
  return(list(tebayes=tt, rebayes=reb, tbest=tbest, rbest=rbest))
}
```

To provide an example, the parameter `m` was allowed to vary to yield signals from the very sparse ($m = 5$) to the completely dense ($m = 10000$). Two plots of the output of the routine `ebdem1` are shown in Figure 5. The left plot compares the Empirical Bayes choice `tebayes` with the ideal, but in practice unattainable, threshold `tbest`. The right plot compares the summed square errors of hard thresholded estimates with the two thresholds. It can be clearly seen

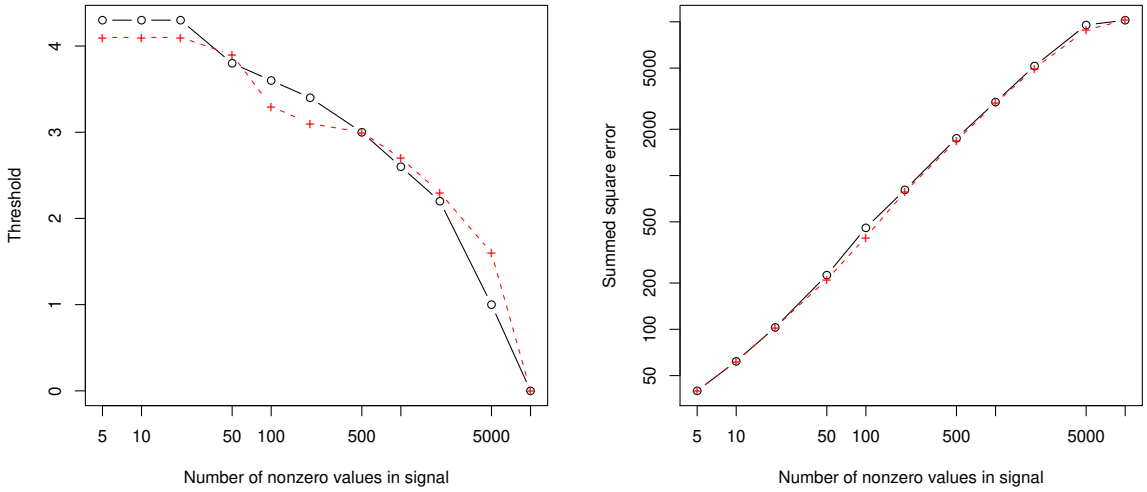

Figure 5: Comparison of empirical Bayes choice of threshold with ideal choice. The signals consist of 10000 observations; the nonzero elements in the signal are uniformly distributed on the interval $[-5, 5]$. Left: Solid black, empirical Bayes threshold; dashed red, threshold that yields hard thresholded estimate with smallest summed square error. Right: Solid black, summed square error for empirical Bayes threshold used in hard thresholding; dashed red, minimum summed square error for any hard thresholded estimate.

how the Empirical Bayes threshold closely tracks the best possible threshold across the range of possible sparsity, and nearly attains the best possible summed square error right across the range. The reason that we use hard thresholding for the Empirical Bayes threshold, rather than the default posterior median estimator, is that allows an honest comparison with the threshold chosen to minimize error under hard thresholding.

3.3. The posterior median thresholding function

Section 3.2 demonstrated the use of empirical Bayes method to estimate a threshold, which was then applied as a hard threshold to the original data, and could be compared directly with a hard threshold fixed explicitly. On the other hand, the posterior median function, as used in Section 3.1, is the default option within the package. To gain some insight, one can plot the thresholding function using the routine `postmed`, which finds the posterior median function for given data and given mixing weight in the prior distribution. Thus, for weight value 0.02, say, and using the quasi-Cauchy prior, we can calculate

```
postmed( seq(from=-8, to=8, by=0.1), w=0.02, prior="cauchy")
```

to produce a vector of results plotted in Figure 6, together with the diagonal $y = x$ for comparison. Similar figures for the Laplace prior are given in Figure 5 of Johnstone and Silverman (2004). It should be stressed that plotting the thresholding function, or accessing the value of the mixing weight explicitly, are not necessary for the direct use of the method using the routine `ebayesthresh`.

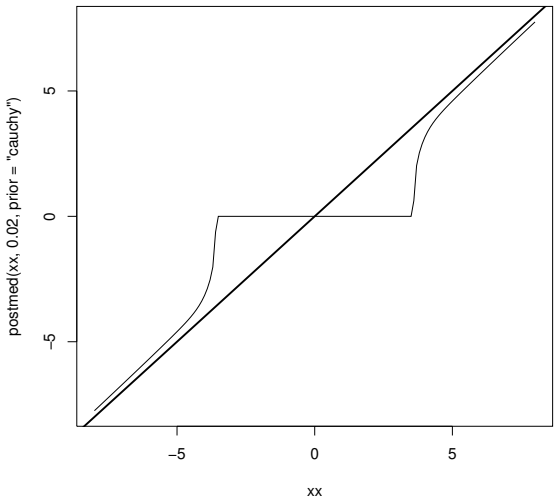

Figure 6: Posterior median thresholding function, for quasi-Cauchy prior with mixing weight 0.02

3.4. Other control parameters in the package's routines

The simulations reported in Section 3 of Johnstone and Silverman (2004) compared a number of different approaches. Some of these are alternative approaches, and are not included in the package. However, some illustrate the use of control arguments in routines within the package. Given a data vector `x`, some of the methods compared are implemented as follows:

```
ebayesthresh(x, a=NA)           # Laplace prior, scale factor also estimated

ebayesthresh(x, prior="cauchy") # quasi-Cauchy prior

ebayesthresh(x, a=NA,           # Laplace prior, scale factor estimated
       threshrule="mean")       # use posterior mean as estimator

ebayesthresh(x, a=NA,           # Laplace prior, scale factor estimated
       threshrule="hard")       # use hard threshold with estimated threshold

ebayesthresh(x)                 # Laplace prior with a=0.5 (default)

ebayesthresh(x, a=0.2)          # Laplace prior with a=0.2

tuniv <- sqrt(2 * length(x))
threshld(x, tuniv)              # hard thresholding with universal threshold
threshld(x, tuniv, hard=FALSE)  # soft thresholding with universal threshold
```

Of course, this list of possibilities is not exhaustive, and the reader should consult the help page for the routine `ebayesthresh` for a full list of optional parameters and their values.

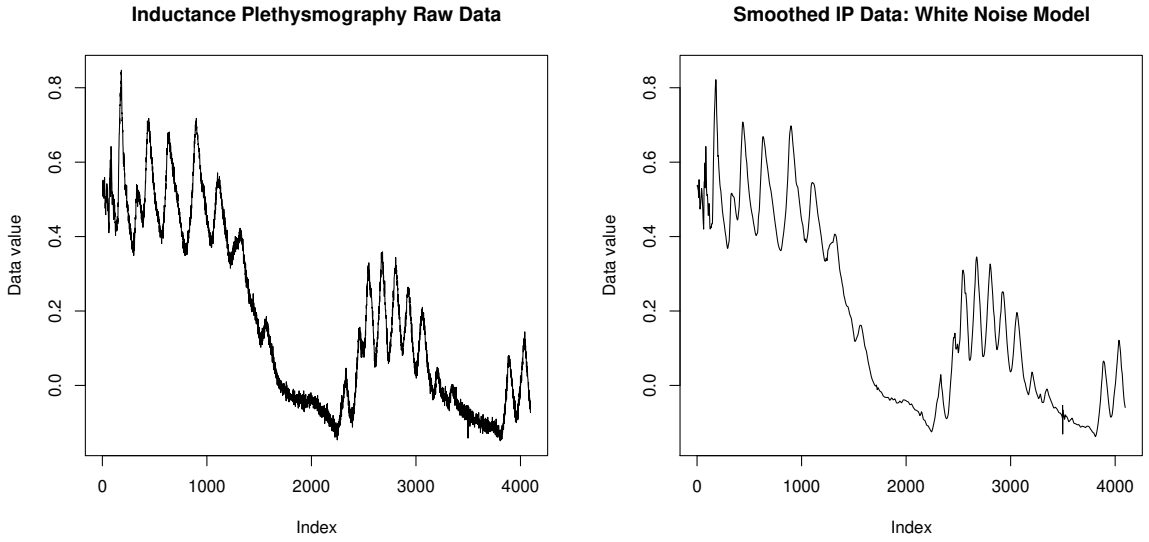

Figure 7: Left: Inductance plethysmography data; right: result of applying the `ebayesthresh.wavelet` routine with the default option of the same noise standard deviation at all levels.

4. Wavelet thresholding

In this section, we go beyond estimation of a single vector or sequence of parameters. We consider the application of the approach to wavelet thresholding, as investigated in Johnstone and Silverman (2005). Our main emphasis is on the first example of that paper, and is intended both to illustrate the use of the `ebayesthresh.wavelet` routine within the **EbayesThresh** package, and also to motivate and explain the use of the general approach in the wavelet context. We then go on to consider an image estimation example.

Within R, the routine `ebayesthresh.wavelet` can be used in conjunction with the wavelet packages **waveslim** (Whitcher 2004) or **wavethresh** (Nason 1998, 2004). In S-PLUS, it will work with the module **S+WAVELETS** (Bruce and Gao 1996), which was used for the computations in Johnstone and Silverman (2005). However, because of the present paper's focus on R, we shall use the **waveslim** package. For this reason, the image estimation discussion is based around a data set available in **waveslim** rather than the less easily available data set considered in Johnstone and Silverman (2005).

4.1. Empirical Bayes thresholding of the discrete wavelet transform

Nason (1996) described a data set obtained in an anesthesiological study using inductance plethsmography. The data were collected in an investigation of the recovery of patients after general anesthesia. The data are available as part of the **wavethresh3** package (Nason 1998). They are also included as the file `ipd.data` accompanying this paper; this file may be read into R by using the `scan` command. The inductance plethysmography data are the first data example contained in Johnstone and Silverman (2005).

The original data `ipd` are plotted in Figure 7. Within the **EbayesThresh** package, the routine

`ebayesthresh.wavelet` applies the empirical Bayes thresholding method level by level within a wavelet transform of the data. In the **waveslim** package, we calculate the discrete wavelet transform transform of the data, allowing for reflection end conditions and calculating six levels of the transform, by

```
ipd.dwt <- dwt(ipd, boundary="reflection", n.levels=6)
```

To apply the empirical Bayes approach level-by-level, using the default options within the routine, we calculate

```
ipdsmooth.dwt <- ebayesthresh.wavelet(ipd.dwt)
```

To find the smoothed version of the data, we invert the discrete wavelet transform. (The package **waveslim** deals with reflection boundary conditions by augmenting the original data vector by a reflected version, and then using periodic boundary conditions. However its inversion routine does not take account of this, and so to obtain a vector the same length as the original we use only the first 4096 values.)

```
ipdsmooth <- idwt(ipdsmooth.dwt)[1:4096]
```

A plot of `ipdsmooth` is given in Figure 7.

The default action of the routine `ebayesthresh.wavelet` is to assume that the original signal is observed with independent Gaussian noise with mean zero and constant variance. The variance of the wavelet coefficients is then estimated from the coefficients at the finest level by a robust approach, using the median absolute deviation from zero as implemented in the `mad` command in R. This approach gives a value of 0.0108 for the noise standard deviation of the wavelet coefficients. Setting the argument `verbose=TRUE` in the routine `ebayesthresh` yields various properties of the wavelet transform. In particular we can obtain the value of the estimated threshold either in absolute terms or as a multiple of the noise standard deviation.

4.2. The estimated thresholds

If the wavelet transforms are calculated using the **waveslim** package, then the **EbayesThresh** package calls the routine `ebayesthresh.wavelet.dwt` to find the empirical Bayes thresholded wavelet transform. Within this routine, the coefficients at the level labelled `j` are found by the call

```
ebayesthresh(x.dwt[[j]], prior, a, bayesfac,vscale, FALSE, threshrule)
```

The value `FALSE` is the value given to the variable `verbose` in the call to `ebayesthresh`. To obtain the values of the thresholds themselves, we change this value to `TRUE`; the call then yields a list, one of whose members is the value of the threshold. To obtain the threshold in terms of the noise standard deviation, we use the call

```
ebayesthresh(x.dwt[[j]], prior, a, bayesfac, vscale,
                    TRUE, threshrule)$threshold.sdevscale
```

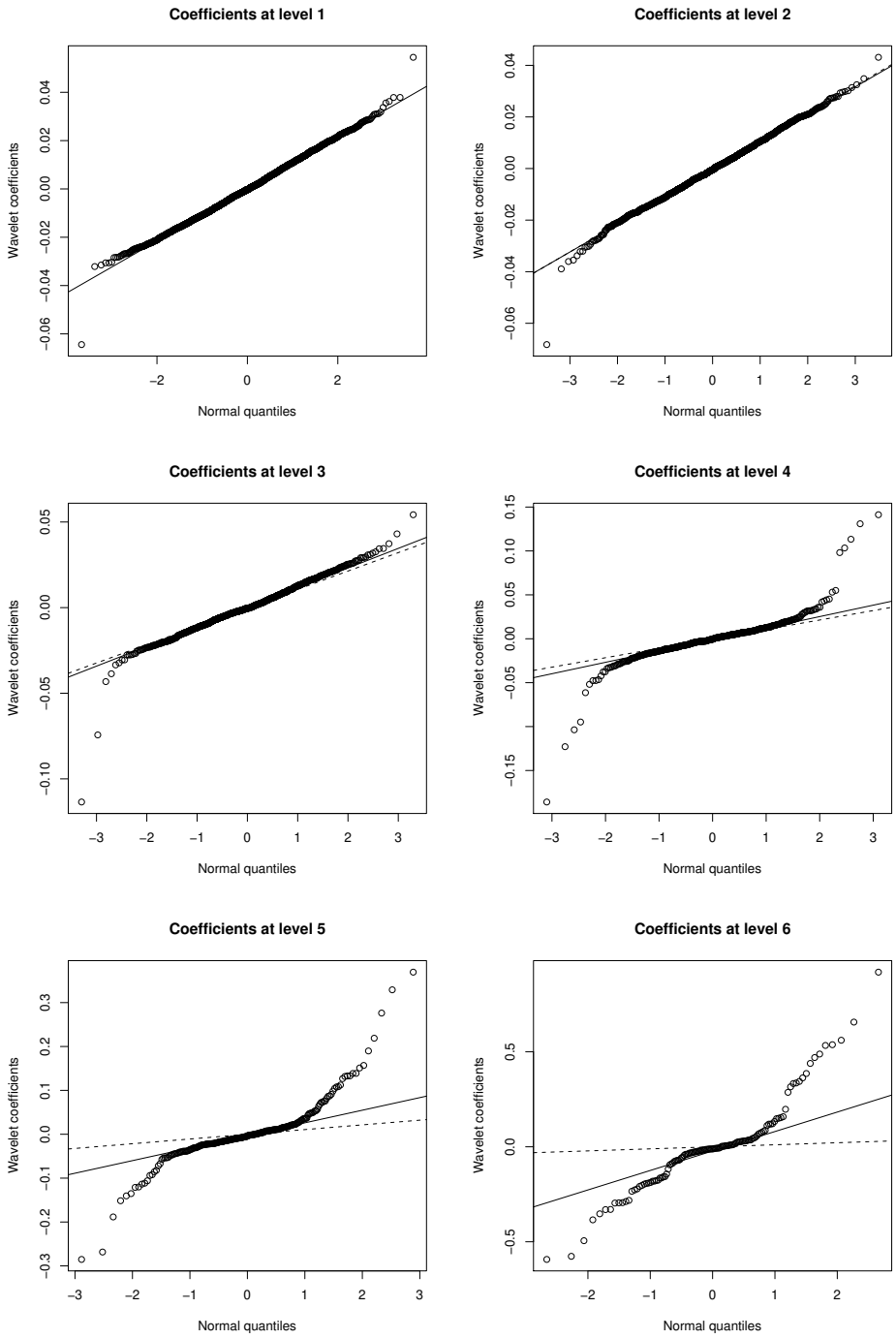

Figure 8: Normal plots of the coefficients at each level of the discrete wavelet transform of the inductance plethysmography data. In each case the solid line gives the expected plot that would be obtained for a normally distributed data set with median absolute deviation equal to that of the given data; the dashed line gives the corresponding line for the median absolute deviation of the coefficients at the finest level.

while to find the threshold in the original scale of the wavelet coefficients we pick out the `threshold.origscale` member of the list given by `ebayesthresh`. The internal labelling convention in **waveslim** is to label the finest scale as level 1, and then to number subsequent scales consecutively. With this labelling, the thresholds estimated by the method are as in the following table:

Level	1	2	3	4	5	6
Threshold (as multiple of noise std dev)	4.08	3.91	3.16	2.29	0	0
Threshold (in absolute terms)	.044	.042	.034	.025	0	0

These thresholds are instructive. At the two finest scales of the transform, the threshold chosen is around four standard deviations, and is in each case the *universal* threshold $\sqrt{2 \log n}$ where n is the length of the vector of coefficients at the relevant level. This is the largest threshold that the method can choose, and is the value appropriate to a very sparse signal. On the other hand, at levels 5 and 6, the chosen threshold is zero, so that essentially no thresholding is carried out; this is the treatment appropriate for a signal that contains no zeroes at all. On the other hand at the intermediate levels 3 and 4, a threshold is chosen between these extremes, corresponding to the notion that the signal is moderately sparse.

Further insight can be gained by examining Figure 8. This gives a normal quantile plot of the wavelet coefficients at each level of the transform. In every case, the dashed line shows the expected plot that would be obtained if the relevant coefficients were all normally distributed with mean zero and standard deviation equal to the value $\hat{\sigma}_1$ estimated from the coefficients at level 1. (Ignore the solid lines for the moment.) It can immediately be seen that, if the noise in the data is assumed to be $N(0, \hat{\sigma}_1^2)$ at every level, it is reasonable to assume that virtually all the underlying signal values at levels 1 and 2 are zero, so that the observed data, except for a very small number of extreme values, come from the noise distribution. On the other hand, the dashed line is a poor fit at levels 5 and (especially) 6, even in the part of the distribution near zero, so it is reasonable that the empirical Bayes method chooses a prior probability of one that the signal values are nonzero. Finally, one can see the appropriateness of considering the signals at levels 3 and 4 to be mixtures of a mass at zero and a nonzero distribution. As one moves from coarser to finer levels, the treatment chosen by the method corresponds to increasing sparsity, and hence to a higher choice of threshold at finer levels.

Even if one did not constrain the noise at all levels to have the same standard deviation, the plots still indicate that the distributions are further from normal at the coarser levels. The solid lines show the expected plots that would be obtained if the data were normally distributed with standard deviation estimated separately at each level; the increasingly heavy tails of the observations at coarser levels are clear.

4.3. The stationary noise model

Johnstone and Silverman (1997) considered the use of wavelet thresholding methods for data where the original noise is stationary but correlated. They showed that an appropriate approach is to carry out wavelet thresholding as if the noise were independent, but to allow different noise variances at different levels. In our context, this would correspond to estimating the noise variance using the median absolute deviation `mad` function separately at each level. Such an approach, based on stationary noise rather than white noise, is available in the

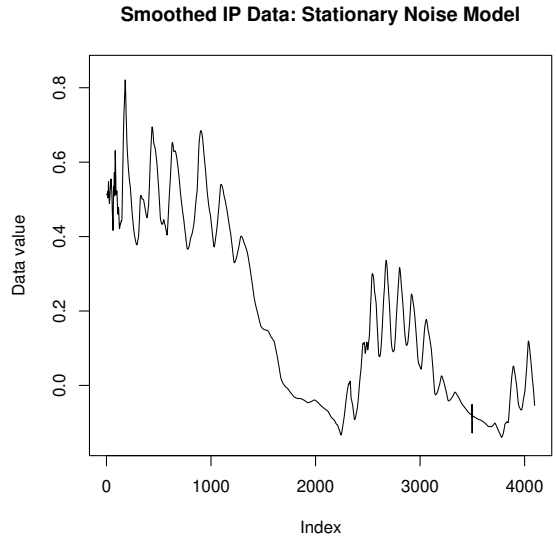

Figure 9: Smoothed inductance plethysmography data, obtained by applying the `ebayesthresh.wavelet` routine with the option `vscale="level"`, which corresponds to an assumption of stationary correlated noise.

`ebayesthresh.wavelet` routine using the parameter value `vscale="level"`. If this is used then the resulting estimated thresholds are as follows:

Level	1	2	3	4	5	6
Threshold (as multiple of noise std dev)	4.08	3.91	3.40	2.61	2.07	1.70
Threshold (in absolute terms)	0.044	0.042	0.039	0.033	0.064	0.184

The treatment of the finest two levels is the same as previously, but coarser levels are thresholded somewhat more severely (higher thresholds) than before, whether the thresholds are expressed in terms of the individual standard deviations or in absolute terms. Another interesting feature is the way that the signal is judged to be progressively less sparse as the scale becomes coarser, again bearing out the impression given by Figure 8.

To obtain an estimate based on the stationary noise model, we use the following code:

```
ipdsmooth.dwt <- ebayesthresh.wavelet(ipd.dwt, vscale="level"))
ipdsmoothstat <- idwt(ipdsmoothstat.dwt)[1:4096]
```

A plot of the resulting estimate is given in Figure 9. A comparison between this estimate and the estimate based on a white noise error model is given in Figure 10. The first segment presented there is the one containing the highest peak in the data. There is little noticeable difference between the two estimates in this region, but if anything the peak is more sharply estimated in the stationary noise model. The second segment is one in which there is regular oscillatory variation at a fairly low frequency; the slight additional smoothness in the stationary noise estimate perhaps yields slightly preferable estimates. In the third short segment, including the high frequency glitch at time 3500, both methods retain the presumably spurious high frequency effect, but the stationary noise method removes the other local variability.

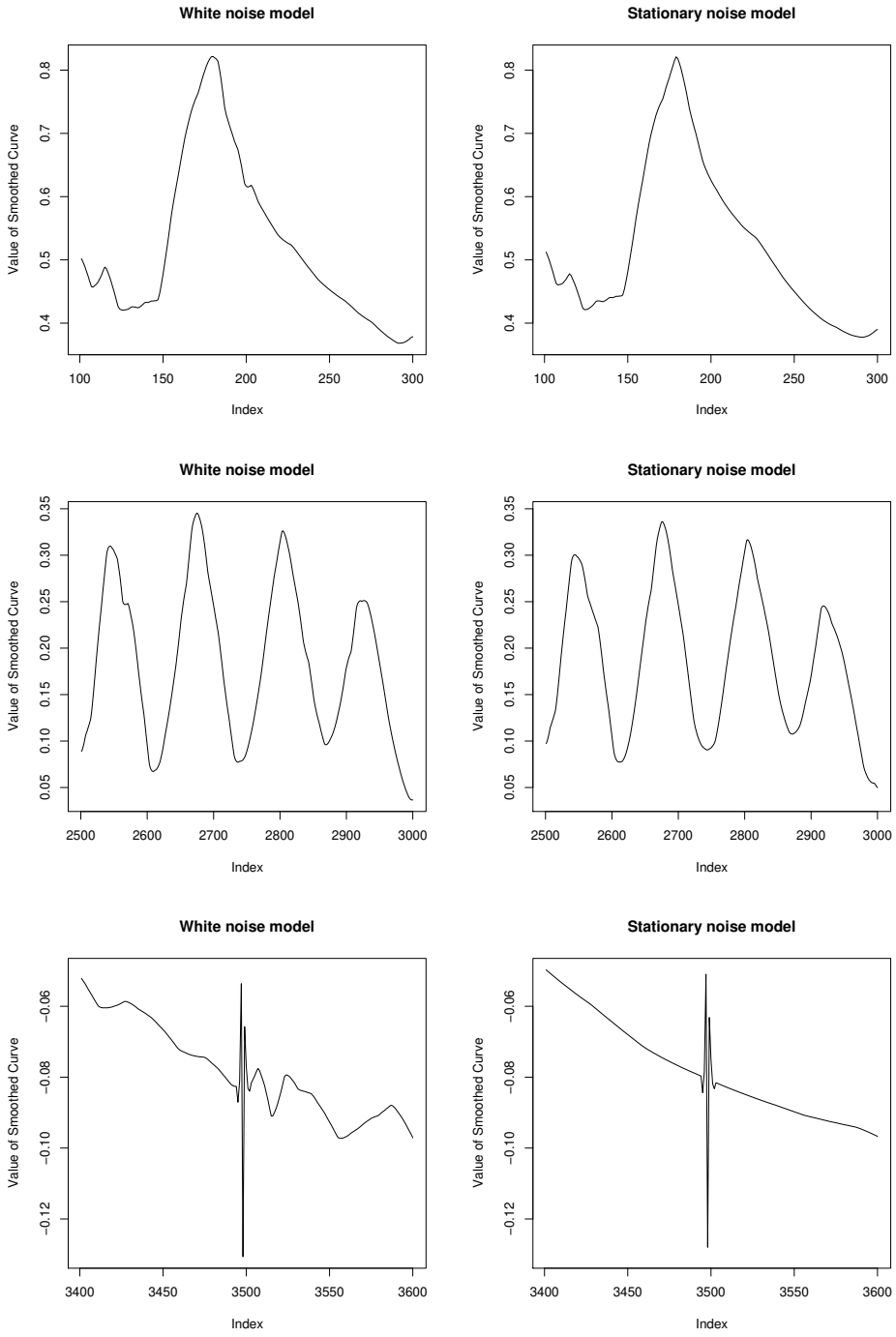

Figure 10: Comparison between wavelet smoothing with a white noise model (left column of plots) for the error and a stationary noise model (right column), where the noise variance in the wavelet coefficients is estimated separately at each level. The comparison is made for various short segments of the data, intervals $(101, 300)$, $(2501, 3000)$ and $(3401, 3600)$ of the original index set.

Overall the stationary noise plots remove some moderately high frequency effects still present in the white noise plots.

The 'glitch' around point 3500 is caused by a single wavelet coefficient at the finest level taking a value six estimated standard deviations from zero, and this, and its partner in the reflected sequence of coefficients, are the only wavelet coefficients at the finest level that survives the thresholding; such a coefficient is highly significant by any accounts. The numerical values of the observations in the interval $[3491, 3510]$ are

```
[3491]  -.078  -.088  -.076  -.086  -.090  -.083  -.054  -.142  -.081  -.071
[3510]  -.098  -.086  -.083  -.078  -.059  -.086  -.090  -.073  -.086  -.076
```

Thus, observation 3497 is somewhat higher than its neighbours, and is immediately followed by the anomalously low observation $-.142$ at time 3498. Given that the instrumentation is generally more stable than this, one possible safeguard in future data analysis would be specifically to look out for outliers of this kind; a simple way of doing this would be to zero out all the wavelet coefficients at the finest level, in other words to use an infinite threshold, and the effect of this in the current plots would be just to remove the glitch.

4.4. The translation-invariant wavelet transform

It is generally recognized that improved smoothing results can often be obtained using the *translation-invariant* wavelet transform; see, for example, Coifman and Donoho (1995). This transform is also called the non-decimated, maximal overlap, or stationary wavelet transform. Given an original sequence of length N, this transform yields a sequence of N coefficients at each scale, rather than a pyramid of coefficient vectors whose length divides by two at successively coarser levels. In **waveslim**, the transform is implemented by the routine `modwt`.

As discussed in detail in Johnstone and Silverman (2005), the most straightforward way of applying the empirical Bayes approach is to threshold the coefficient vector at each level as if it were an independent sequence. The estimated curve is then obtained by the standard inversion algorithm for the translation-invariant wavelet transform, as implemented in the **waveslim** routine `imodwt`.

The routine `ebayesthresh.wavelet`, and the routines that it calls, are written to handle results of appropriate translation-invariant wavelet transforms in exactly the same way as the conventional discrete wavelet transform. To apply empirical Bayes wavelet smoothing to the inductance plethysmography data using the translation-invariant transform, and also allowing for a stationary noise model, we therefore proceed as follows:

```
ipd.modwt       <- modwt(ipd, boundary="reflection", n.levels=6)
ipdsmooth.modwt <- ebayesthresh.wavelet(ipd.modwt, vscale="level")
ipdsmooth       <- imodwt(ipdsmooth.modwt)[1:4096]
```

The result of this procedure is plotted in Figure 11. It can be seen that the estimated curve is somewhat smoother than those obtained using the standard discrete wavelet transform. The high frequency effect at index 3500 is reduced in size. This is because, at the finest level, the inverse of the translation-invariant wavelet transform involves averaging the inverse of discrete transforms at two different positions, corresponding to basing the wavelets at odd or even positions in the original data sequence. The very large wavelet coefficient only occurs in one of these sequences, and so in the estimated curve the amplitude of the 'glitch' is halved.

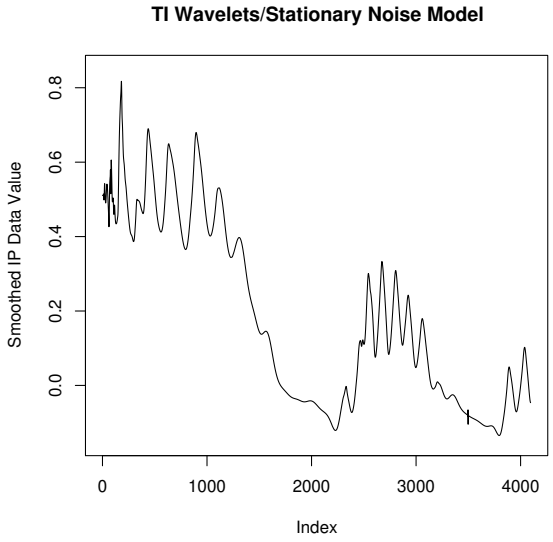

Figure 11: Smoothed inductance plethysmography data, obtained from a translation-invariant wavelet transform and then applying the `ebayesthresh.wavelet` routine with the option `vscale="level"`, which corresponds to an assumption of stationary correlated noise.

4.5. Smoothing an image

We now consider the possible use of the method for the processing of the wavelet transform of a two-dimensional image. The example we use will be the image of Ingrid Daubechies contained in the **waveslim** package. Especially when processing images, it may be appropriate to use dictionaries other than the standard two-dimensional wavelet transform. Therefore this section should be read in a 'tutorial' way; its purpose is not to set out a black box recipe, but to illustrate how the basic `ebayesthresh` routine can be used in a broader context.

Having loaded the package **waveslim**, we make the test image available and we reverse its sign, in order to obtain an image that comes out in positive rather than negative when using `image` with the option `col=gray(1:100/100)`. We then construct a noisy image `dauerr` by adding random normal noise. The standard deviation of the original image is about 35 and for this example we use noise with standard deviation 10. Finally, we construct the two-dimensional wavelet transform of `dauerr` using the routine `dwt.2d` and the Daubechies `d6` wavelet.

```
data(dau)                                  # load data set
dau <- -dau                                # negate the image
set.seed(55)                               # set random seed
dauerr <- dau + rnorm(256*256, sd=10)      # generate and add noise
dauerr.dwt <- dwt.2d(dauerr, wf="d6")      # perform wavelet transform
```

The transform `dauerr.dwt` is a list of length 13, with names

```
"LH1" "HL1" "HH1" "LH2" "HL2" "HH2" "LH3" "HL3" "HH3" "LH4" "HL4" "HH4" "LL4"
```

Each member of the list is a matrix of coefficients. In each case, the first two letters of the name indicate whether the corresponding basis functions correspond to a high or low pass

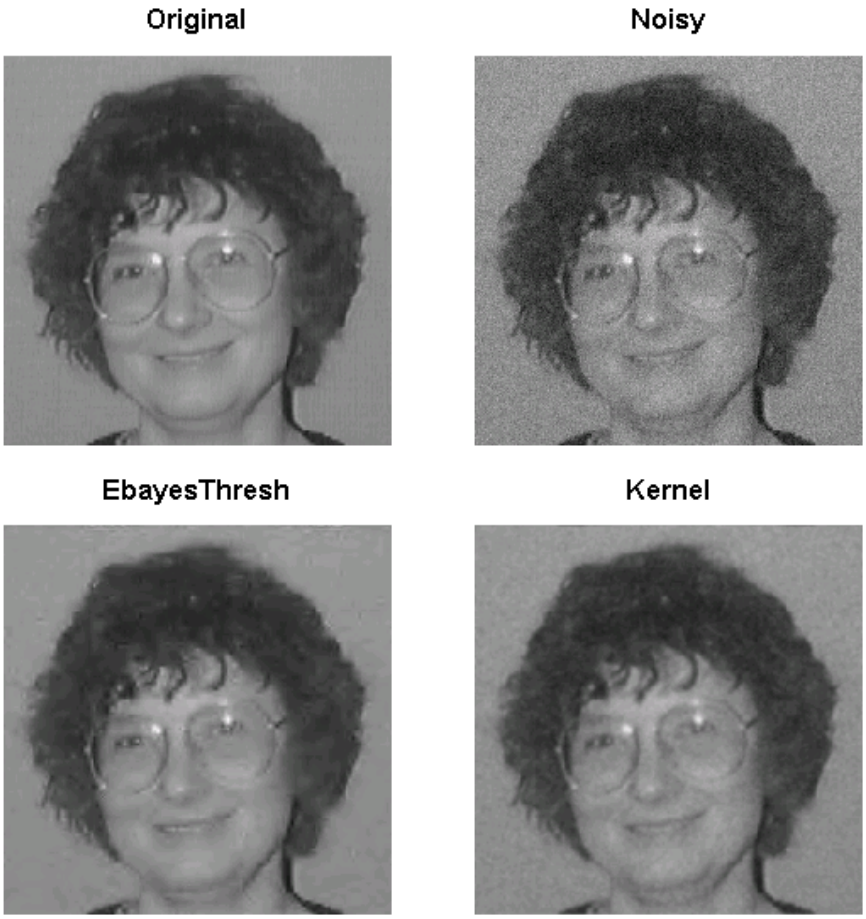

Figure 12: Top left: original image of Ingrid Daubechies; top right: effect of adding normal independent noise; bottom left: result of applying the empirical Bayes smoothing method to the individual matrices of coefficients; bottom right: kernel smooth of noisy image, with bandwidth chosen to minimize the average absolute error.

filter in the X or Y direction respectively. The number refers to the scale. Thus, the first three matrices all relate to detail at the finest scale, and the first matrix `dauerr.dwt$LH1` gives the coefficients of the function $\phi(x)\psi(y)$ suitably rescaled and shifted, where ϕ is the scaling function of the wavelet family and ψ the mother wavelet.

In order to estimate the noise standard deviation from the data, we apply the median absolute deviation function to the wavelet coefficients at level 1:

```
sd <- mad( c(dauerr.dwt$LH1, dauerr.dwt$HL1, dauerr.dwt$HH1) )
```

which gives the result 10.25, very close to the theoretical value 10. The matrix `dauerr.dwt$LL4` gives the scaling coefficients at the coarsest level considered; we preserve these unchanged in the estimate. We apply the Empirical Bayes thresholding method to the remaining coefficients as follows, first copying `dauerr.dwt` to `dauerr.smooth.dwt`, then estimating each of the 12 matrices in the list using `ebayesthresh`, and finally inverting the transform to find the estimate:

```
dauerr.smooth.dwt <- dauerr.dwt
for ( j in (1:12) ) {
  dauerr.smooth.dwt[[j]] <- ebayesthresh( dauerr.dwt[[j]], sdev=10.25 ) }
dauerr.smooth <- idwt.2d(dauerr.smooth.dwt)
```

It is of interest to consider the thresholds estimated by the method. To do this most easily, we define a function `get.thresh` and then apply it to each of the matrices in the transform:

```
get.thresh <- function(xx) ebayesthresh(xx,verbose=T,sdev=sd)$threshold.sdevscale
ttl <- lapply ( dauerr.dwt , get.thresh )
```

The list `ttl` then contains the estimated thresholds, each in the scale of the standard deviation; printing out `round(unlist(ttl))` gives the result

```
 LH1  HL1  HH1  LH2  HL2  HH2  LH3  HL3  HH3  LH4  HL4  HH4  LL4
4.41 3.81 4.41 2.42 2.58 3.26 1.01 0.92 1.85 0.00 0.00 0.00 0.00
```

These thresholds are interesting. There is no thresholding at level 4. At the finest level 1, on the other hand, the LH and HH coefficients are thresholded at the universal threshold $\sqrt{2}\log(128 \times 128)$ for data sets of their size, while the HL coefficients are subject to thresholding nearly as stringent. At levels 2 and 3 the thresholding applied to the HH coefficients is higher than that applied to the LH or HL coefficients. It is reasonable to consider the HH2 coefficients as being at a level intermediate between level 1 and 2, and the HH3 as being intermediate between levels 2 and 3, and so on; with this convention, the estimated thresholds increase monotonically as one moves from coarser to finer levels.

Figure 12 shows the original image, the noisy image, and the estimate of the image as obtained using `ebayesthresh` as set out in this section. In addition, we show a kernel smooth calculated in R by

```
kk            <-  kernel("fejer", r=2, m=2)
dsm1          <-  kernapply(t(dauerr), kk, circular=TRUE)
dauerr.smooth <-  kernapply(t(dsm1), kk, circular=TRUE)
```

The bandwidth parameter `m=2` has been adjusted to minimize the L_1 distance between the estimate and the original image `dau`; the average L_1 error of both the kernel smooth and the fully automatic `ebayesthresh` smooth is 3.07.

In the kernel smooth, there is considerable remaining random error in the flat parts of the plot, and some of the highlights and details are slighly more smoothed out than in the wavelet plot. On the other hand, in the wavelet plot there are some spurious artefacts. In addition, it is encouraging that the **EbayesThresh** method has succeeded in automatically achieving the L_1 error of the kernel estimate chosen by reference to the true image. Furthermore, it is now well understood that the standard two-dimensional wavelet transform is not a very good dictionary for the representation of images. In principle, our empirical Bayes approach is equally applicable whatever the transform used, and will take advantage of sparsity in the representation of the function or image being estimated, and so if a dictionary more specifically suited to the representation of this type of image were used, one could expect even better results.

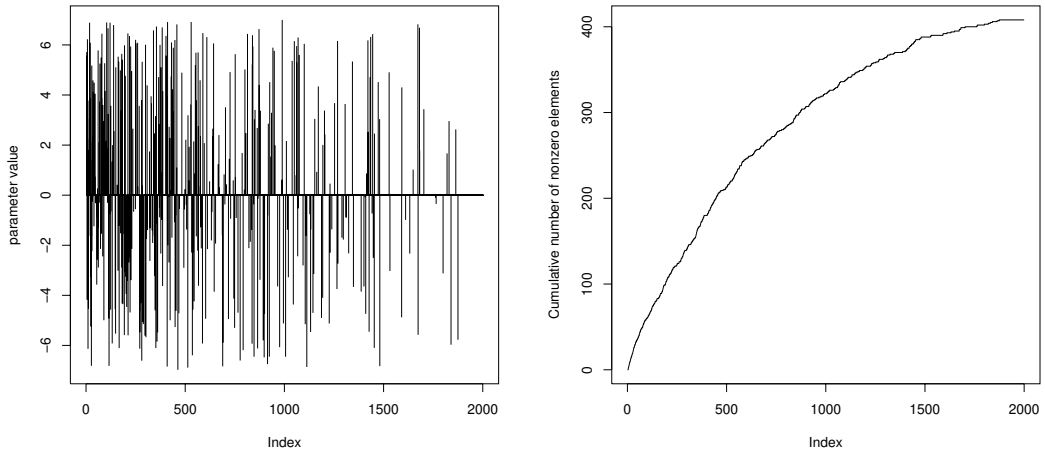

Figure 13: Left panel: A signal of increasing sparsity. Right panel: the cumulative number of nonzero elements up to each point of the sequence.

5. More general mixture weight structure

In considering wavelet decompositions, the empirical Bayes approach was applied allowing a different mixture weight w at each level of the decomposition. In this section, we consider two further extensions of our original method, allowing the mixture weight to vary in two different ways.

5.1. Increasing sparsity along the sequence

Suppose we have a single sequence μ_i believed to become more sparse as i increases, in the sense that the prior probability that μ_i is zero may reasonably be considered to increase as i increases. For example, μ_i may be the coefficients of a function in a dictionary where the early terms in the sequence describe large-scale aspects of the function or phenomenon of interest, while as we proceed further along the sequence, the terms describe finer and finer detail. For example, Jansen, Nason, and Silverman (2004) construct just such a basis for the analysis of data observed on an irregular set of points in two dimensions.

A natural approach is to model μ_i as having prior distributions of the same form as previously, but with weight w_i depending on i, so that μ_i has prior density

$$(1 - w_i)\delta(u) + w_i\gamma(u)$$

where δ is a Dirac delta function at zero. If we assume only that the weights w_i decrease as i increases, then we can, in principle, estimate the weights by marginal maximum likelihood. The estimating sequence $\hat{w}_i$ will be chosen to maximize the log marginal likelihood

$$\ell(w_1, \ldots, w_n) = \sum_{i=1}^{n} \log\{(1 - w_i)\phi(x_i) + w_i g(x_i)\} \tag{6}$$

subject to the constraint $w_1 \geq w_2 \geq \ldots \geq w_n$. An algorithm for carrying out this maximization is described in Section 7.1. Once the weights have been estimated, we estimate each μ_i

separately, using a thresholding rule based on the Bayesian model with mixing parameter w_i.

The routine `wmonfromx` carries out this estimation procedure, subject to the additional constraint that all the thresholds corresponding to the w_i are bounded by $\sqrt{2 \log n}$. As an example, of a signal with increasing sparsity, see the sequence of length 2000 plotted in Figure 13. This was generated using the code

```
set.seed(1)
pp <- 1- ((1:2000)/2000)^0.25
mu <- runif(2000, -7,7)*rbinom(2000,1,pp)
```

Each nonzero value in the signal is uniformly distributed on $[-7, 7]$. The increasing sparsity is demonstrated by the right hand panel in the figure, which shows the number of nonzero values up to a particular point in the sequence. It can be seen, for example, that more than half the nonzero values occur in the first quarter of the sequence. There are 123 nonzero values in the first 250 places in the sequence, but only 8 in the last 250.

Noise `rnorm(2000)` was added to give a data sequence `x`. This sequence is shown in Figure 14, together with the mixing weights `wmon` estimated by the monotone estimation procedure, the corresponding thresholds `thresh`, and the result `muhat` of using the monotone weights in the estimation. The code for these calculations is as follows:

```
wmon   <- wmonfromx(x)
thresh <- tfromw(wmon)
muhat  <- postmed(x,wmon)
```

Note that the thresholds do not enter into the calculation of the posterior median estimator, because that calculation is carried out using the weights. It can be seen from Figure 14 that the estimated weights remain constant over intervals and then jump downwards (with corresponding upward jumps in the thresholds). This is characteristic of sequences estimated subject to monotonicity constraints. In some contexts, the points at which the jumps take place can be used as candidates for segmenting the series into artificial 'levels'. See, for example, Jansen *et al.* (2004) for an example of this approach, when the series considered consists of the coefficients of the representation of a function, by a dictionary constructed to represent finer detail as the sequence progresses.

5.2. Parametric dependence

In Section 5.1 we considered weights that were constrained to decrease along the sequence. Another possibility is that we have known constants c_i and that we wish, as far as possible, for the weights to be proportional to c_i, leaving a constant of proportionality to be estimated. Such a possibility could arise, for instance, when the θ_i are the coefficients of the expansion of an unknown function in a particular dictionary of increasingly finer scale functions, and the c_i were some measure of the size of support of the ith dictionary function. See Jansen *et al.* (2004) for an example. The algorithm described in this section is implemented as the routine `zetafromx` in the **EbayesThresh** package.

Suppose, therefore, we have data z_i for $i = 1, \ldots, n$, and consider the basic model $w_i = c_i\zeta$ where c_i are known constants. Let w_{lo} be the weight whose corresponding threshold $t(w_{lo}) =$

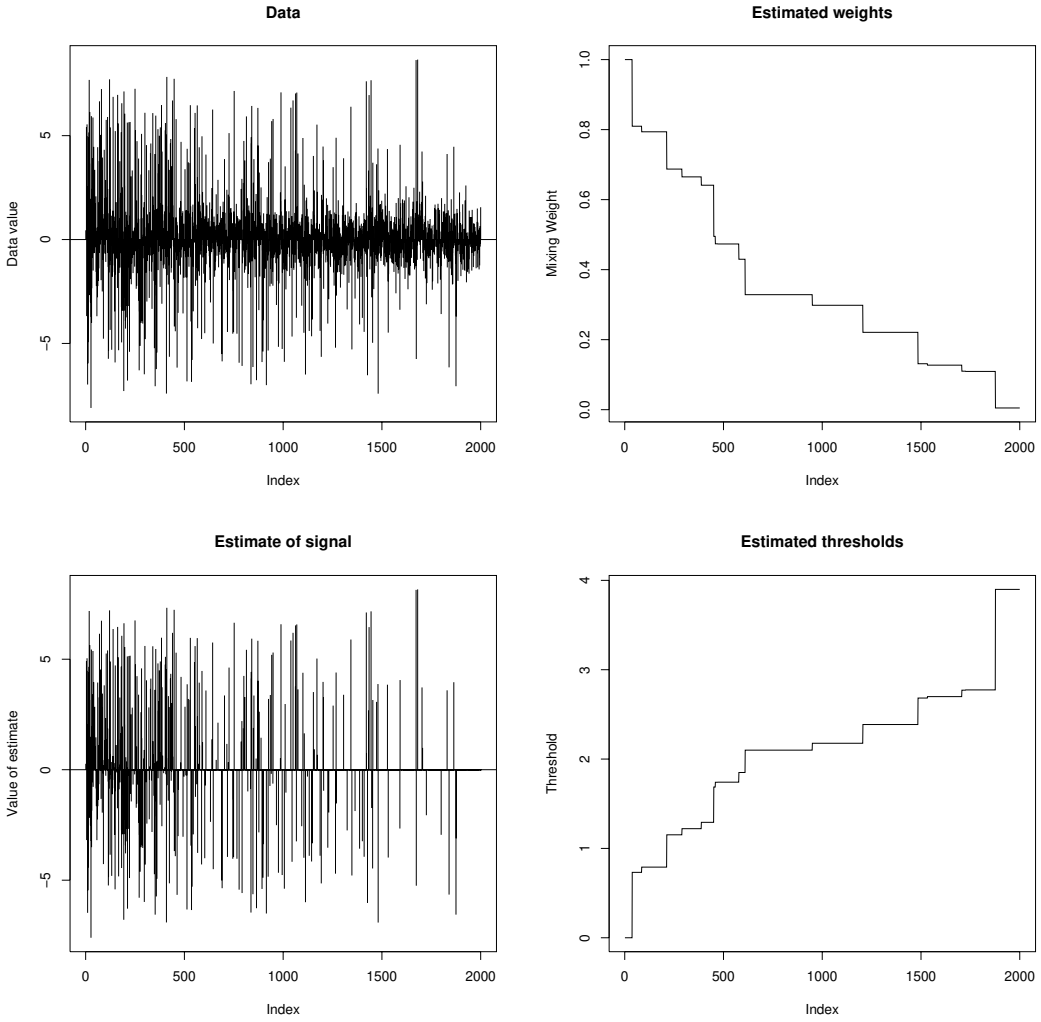

Figure 14: Top left: Raw data; top right: weights estimated by the monotone marginal maximum likelihood approach; bottom left: estimate of underlying signal; bottom right: posterior median thresholds corresponding to estimated weights. The effect of the estimation is to allow through the data in the early part of the signal, but to perform quite stringent thresholding near the end.

$\sqrt{2 \log n}$. In order to enforce the constraints $w_{lo} \leq w_i \leq 1$ we refine the model to

$$w_i(\zeta) = \text{median}\{w_{lo}, c_i\zeta, 1\}, \tag{7}$$

so that ζ becomes a *weight parameter* rather than simply a constant of proportionality. Letting g be the convolution of γ with ϕ, the marginal log likelihood function is then given by

$$\ell(\zeta) = \sum_i \log[\{1 - w_i(\zeta)\}\phi(z_i) + w_i(\zeta)\gamma(z_i)] \tag{8}$$

The maximization of $\ell(\zeta)$ is not entirely straightforward, but can be carried out by an approach described in detail in Section 7.2. We consider, as a simple example, the same data as in Section 5.1. We shall set $c_i = i^{-1}$ so that the weights decrease, as far as possible, inversely as the position in the sequence. We now find the value of ζ, the corresponding values $w(\zeta)$, the thresholds, and the estimate of the signal, by using the routine `zetafromx`. This provides a list including elements `zeta`, the estimated value of ζ (in this case 236) and the weights $w(236)$. So the weights `wts`, the estimate `muhat` and the corresponding thresholds `thresh` can be found as follows:

```
wts   <- zetafromx(x, cs=1/(1:2000))$w     #   find weights
muhat <- postmed(x, wts)                   #   carry out estimation
thresh<- tfromw(wts)                       #   find thresholds
```

Figure 15 shows the estimated weight sequence and the corresponding thresholds, as well as the estimate obtained using these weights. The figure also shows the discrepancy between the estimates obtained by the parametric and monotone methods, plotted on the same scale as the estimate itself. It is interesting that most of the difference is in the first part of the sequence, where the 'parametric' threshold is exactly zero, but the 'monotone' threshold starts to move away from zero.

There are also some individual large discrepancies further on in the sequence, the largest two of which are set out in the following table:

			Monotone			Parametric		
i	x	μ	weight	threshold	estimate	weight	threshold	estimate
1158	2.36	0	0.298	2.177	0.924	0.204	2.438	0
1205	−2.37	0	0.298	2.177	−0.952	0.196	2.462	0

In both cases, it can be seen that the discrepancy between the two estimates arises because the observation falls just above the threshold for the 'monotone' method but just below that for the 'parametric' method. As it happens, both the true parameter values are equal to zero, and each observation is in the extreme 1% of the relevant tail of the error distribution. The thresholds chosen by the two methods are not dramatically discrepant, but observations that fall between the two thresholds will be treated somewhat differently by the two methods.

6. Algorithmic and mathematical details for the basic method

In this section, we give details of the calculations involved in implementing our core methodology, where the same w is used in the prior for each μ_i in the sequence under consideration.

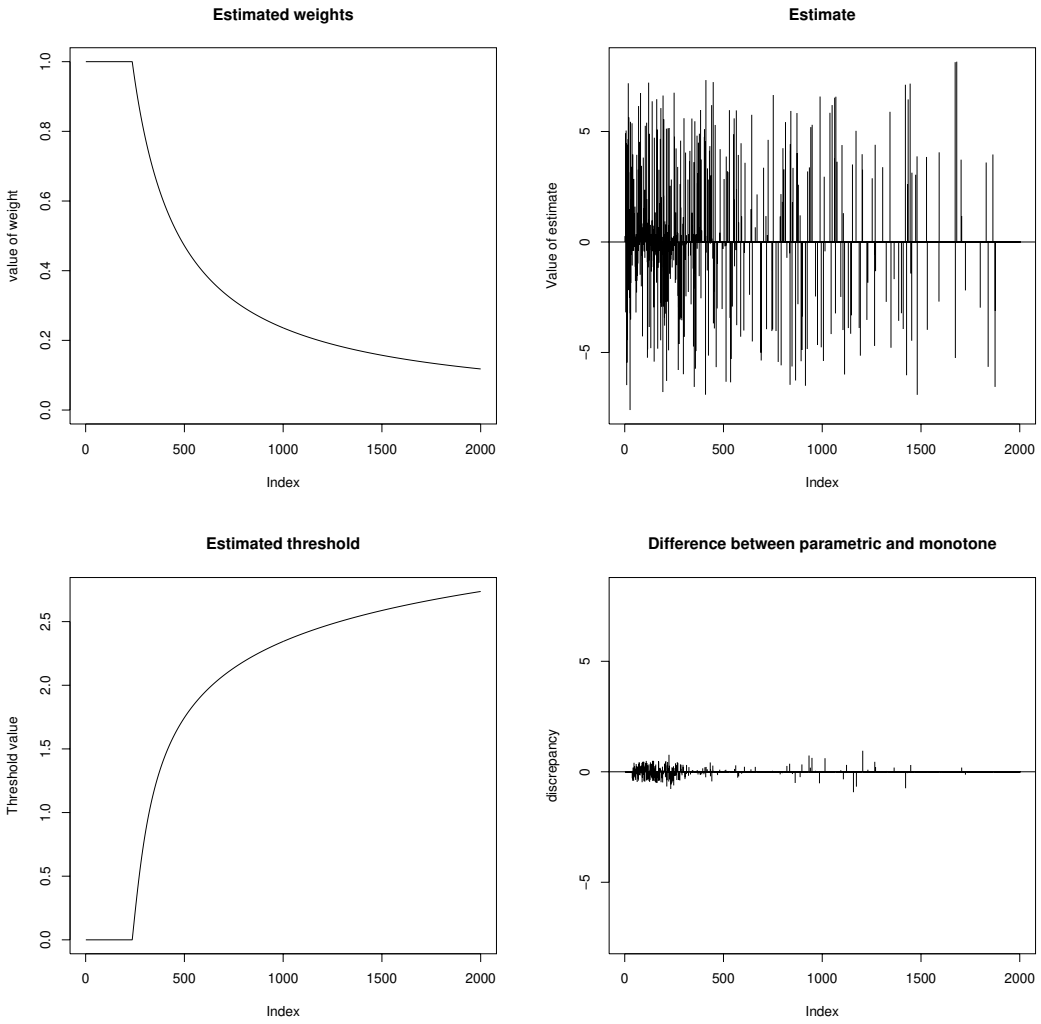

Figure 15: Top left: weights chosen by parametric approach; top right: posterior median estimate found using these weights; bottom left: thresholds corresponding to the estimated weights; bottom right: difference between estimate obtained by the parametric method and that obtained by the monotone method.

The details go considerably beyond those presented in previous papers. We begin by setting out generic calculations for the relevant quantities, and then address the Laplace and quasi-Cauchy priors specifically. Throughout, we use ϕ and Φ to denote, respectively, the standard normal density and cumulative distribution functions, and we define $\tilde{\Phi}(t) = 1 - \Phi(t)$ for all t.

6.1. Generic calculations

Posterior probability that parameter is nonzero Define

$$\beta(x) = g(x)/\phi(x) - 1. \tag{9}$$

Then the posterior probability $w_{\text{post}}(x) = P(\mu \neq 0 | X = x)$ will satisfy

$$w_{\text{post}}(x) = wg(x)/\{wg(x) + (1-w)\phi(x)\} = (1 + \beta(x))/(w^{-1} + \beta(x)), \tag{10}$$

and hence can be found using the function β alone.

Posterior mean Define

$$f_1(\mu | X = x) = f(\mu | X = x, \mu \neq 0),$$

so that the posterior density

$$f_{\text{post}}(\mu | X = x) = (1 - w_{\text{post}})\delta_0(\mu) + w_{\text{post}} f_1(\mu | x).$$

Let $\mu_1(x)$ be the mean of the density $f_1(\cdot|x)$. The posterior mean $\tilde{\mu}(x; w)$ is then equal to $w_{\text{post}}(x)\mu_1(x)$.

Posterior median To find the posterior median $\hat{\mu}(x; w)$ of μ given $X = x$, let

$$\tilde{F}_1(\mu | x) = \int_{\mu}^{\infty} f_1(u|x) du.$$

If $x > 0$, we can find $\hat{\mu}(x, w)$ from the properties

$$\begin{array}{lcll} \hat{\mu}(x; w) & = & 0 & \text{if } w_{\text{post}}(x)\tilde{F}_1(0|x) \leq \frac{1}{2} \\ \tilde{F}_1(\hat{\mu}(x; w)|x) & = & \{2w_{\text{post}}(x)\}^{-1} & \text{otherwise} \end{array} \tag{11}$$

Note that if $w_{\text{post}}(x) \leq \frac{1}{2}$ then the median is necessarily zero, and it is unnecessary to evaluate $\tilde{F}_1(0|x)$. If $x < 0$, we use the antisymmetry property $\hat{\mu}(-x, w) = -\hat{\mu}(x, w)$.

Marginal maximum likelihood weight The explicit expression for the function g facilitates the computation of the maximum marginal likelihood weight in the single sequence case. Define the score function $S(w) = \ell'(w)$. Then

$$S(w) = \sum_{i=1}^{n} \frac{g(x_i) - \phi(x_i)}{(1-w)\phi(x) + wg(x)} = \sum_{i=1}^{n} \frac{\beta(x_i)}{1 + w\beta(x_i)} = \sum_{i=1}^{n} \frac{\beta_i}{1 + w\beta_i} \tag{12}$$

where $\beta_i = \beta(x_i)$. Define w_{lo} to be such that $t(w_{lo}) = \sqrt{2 \log n}$. Then it is easy to show that $S(w)$ is a decreasing function of w for w in $[0, 1]$. Furthermore, the maximum marginal likelihood estimate of w will be given by the root of $S(w) = 0$ for w in the interval $[w_{lo}, 1]$; this can be found by a binary search algorithm. In the package this binary search is done on the logarithmic scale, so that at each stage the next value of w is the geometric mean of the endpoints of the interval on which the root is known to lie. If $S(1) > 0$ then the estimate of w is 1, while if $S(w_{lo}) \leq 0$ then the estimate is w_{lo}. Note that once the β_i have been found, and the endpoint w_{lo}, the algorithm does not depend on the prior at all, only on the values β_i. Therefore to cater for a new prior, it is only necessary to have a routine to find the function β, and to evaluate the limiting weight w_{lo}.

6.2. Finding the Bayes factor threshold

The posterior median threshold is defined to be the value $t(w)$ such that

$$P(\mu > 0 | X = t(w)) = 0.5.$$

Thus $t(w)$ is the largest value of the observed data for which the estimated μ will be zero, if the estimate is carried out by the posterior median. By contrast, the Bayes factor threshold, as introduced in Section 2.3, is defined as the value $\tau_b(w) > 0$ such that

$$P(\mu \neq 0 | X = \tau_b(w)) = 0.5, \tag{13}$$

and leads to simpler calculations in some cases.

To find the Bayes factor threshold $\tau_b = \tau_b(w)$, we express the definition (13) in odds rather than probabilities, to give

$$\text{Odds}(\mu \neq 0 | X = \tau_b) = \frac{w g(\tau_b)}{(1 - w)\phi(\tau_b)} = 1.$$

Rearranging, this yields

$$\beta(\tau_b) + 1 = (1 - w)/w = w^{-1} - 1.$$

so that, provided $w \leq 1/(2 + \beta(0))$

$$\beta(\tau_b) = w^{-1} - 2. \tag{14}$$

This equation can be used in two ways. If we wish to find the w corresponding to a given threshold, we have

$$w = (2 + \beta(\tau_b))^{-1}$$

so we only need to be able to evaluate the function β in order to find w from τ_b. To find τ_b from w, on the other hand, provided $w \leq 1/(2 + \beta(0))$, we can solve equation (14) by a binary search, because of the monotonicity of β. If $w > 1/(2 + \beta(0))$, then we will have $P(\mu \neq 0 | X = x) > 0.5$ for all x, even $x = 0$, and we define the Bayes factor threshold $\tau_b = 0$.

6.3. Laplace prior

The calculations set out above show that the key quantities are the marginal density g, the function β, the mean function $\mu_1(x)$, and the tail conditional probability function $\tilde{F}_1$. In this

section, much fuller details are given for the Laplace prior. The subheadings in each case give the names of the relevant routines in the software package.

The routine `beta.laplace`

For the Laplace distribution prior, we have

$$g(x) = \tfrac{1}{2}a\exp(\tfrac{1}{2}a^2)\{e^{-ax}\Phi(x-a) + e^{ax}\tilde{\Phi}(x+a)\},$$

and therefore

$$\beta(x) = \beta(x,a) = \tfrac{1}{2}a\{\frac{\Phi(x-a)}{\phi(x-a)} + \frac{\tilde{\Phi}(x+a)}{\phi(x+a)}\} - 1. \tag{15}$$

In practice, it is appropriate to use the approximation $\tilde{\Phi}(y)/\phi(y) \approx 1/y$ for $y > 35$, say, in order to avoid obtaining a result containing the indeterminate `NaN`. (The quantity $\Phi(x-a)/\phi(x-a)$ can yield a result that overflows and is therefore numerically infinite, but this does not cause problems in subsequent calculations.) Also, it is best to write a routine for positive x and to use the symmetry of β if x is negative.

Once the values $\beta(x_i)$ have been evaluated, the only quantity needed to find the marginal maximum likelihood weight for fixed a is the lower bound w_{lo} on the weight as defined just after (12). The routine `wfromx` does the calculations for the actual maximization of the likelihood, in the manner set out in Section 6.1; the bound on the weight corresponding to the upper bound $\sqrt{2\log n}$ on the threshold is found using the routine `wfromt` described below.

The routine `postmean.laplace`

Consider first the evaluation of the posterior mean. As explained in Section 6.1, the calculation (10) allows the posterior probability $w_{\text{post}}(x)$ to be found using the routine `beta.laplace`, and it remains to find the mean $\mu_1(x)$ of the posterior distribution of μ conditional on $\mu \neq 0$. We have

$$f_1(\mu|x) = \begin{cases} e^{ax}\phi(\mu - x - a)/\{e^{-ax}\Phi(x-a) + e^{ax}\tilde{\Phi}(x+a)\} & \text{if } \mu \leq 0 \\ e^{-ax}\phi(\mu - x + a)/\{e^{-ax}\Phi(x-a) + e^{ax}\tilde{\Phi}(x+a)\} & \text{if } \mu > 0 \end{cases} \tag{16}$$

which is a weighted sum of truncated normal distributions. Hence it can be shown that, for $x > 0$,

$$\mu_1(x) = x - \frac{a\{e^{-ax}\Phi(x-a) - e^{ax}\tilde{\Phi}(x+a)\}}{e^{-ax}\Phi(x-a) + e^{ax}\tilde{\Phi}(x+a)}. \tag{17}$$

The posterior mean is now equal to $w_{\text{post}}(x)\mu_1(x)$, as set out in Section 6.1.

The routine `postmed.laplace`

Now turn to the determination of the posterior median. Suppose $x > 0$. For $\mu \geq 0$, we have

$$\tilde{F}_1(\mu|x) = \frac{e^{-ax}\tilde{\Phi}(\mu - x + a)}{e^{-ax}\Phi(x-a) + e^{ax}\tilde{\Phi}(x+a)}.$$

Hence, if the posterior median is greater than zero, we will have, using (11),

$$\begin{aligned} \frac{e^{-ax}\tilde{\Phi}(\hat{\mu} - x + a)}{e^{-ax}\Phi(x-a) + e^{ax}\tilde{\Phi}(x+a)} &= \frac{wg(x) + (1-w)\phi(x)}{2wg(x)} \\ &= a^{-1}w^{-1}\exp(-\tfrac{1}{2}a^2)\phi(x)\frac{1 + w\beta(x)}{e^{-ax}\Phi(x-a) + e^{ax}\tilde{\Phi}(x+a)}. \end{aligned}$$

Simplifying, this leads to

$$\tilde{\Phi}(\hat{\mu} - x + a) = a^{-1}w^{-1}\phi(x - a)\{1 + w\beta(x)\},$$

so that, using the property that $\tilde{\Phi}^{-1}(u) = -\Phi^{-1}(u)$, we have

$$\hat{\mu} = x - a - \Phi^{-1}(z_0),$$

where

$$z_0 = a^{-1}\phi(x - a)\{w^{-1} + \beta(x)\}. \tag{18}$$

As $x \to \infty$, the limiting value of z_0 is 0.5, and it is helpful to use this approximation when x is so large that $\phi(x-a)$ is numerically zero and $\beta(x)$ is numerically infinite. If x is sufficiently small that the value of z_0 given by (18) is greater than 1, or that $x - a - \Phi^{-1}(z_0)$ is negative, then the posterior median will be equal to zero. Therefore, we set

$$\hat{\mu} = \max[0, x - a - \Phi^{-1}\{\min(1, z_0)\}].$$

The routine `tfromw(prior="laplace")`

The posterior median threshold will be the value $t(w)$ such that $t-a-\Phi^{-1}(z_0) = 0$. Therefore

$$\Phi(t - a) - a^{-1}\phi(t - a)\{w^{-1} + \beta(t)\} = 0,$$

an equation that can be solved by binary search to find t from w. The left hand side of this equation is evaluated by the function `laplace.threshzero`. Furthermore, by rearranging we obtain an explicit expression giving $w = w(t)$ in terms of t:

$$w(t)^{-1} = a\Phi(t - a)/\phi(t - a) - \beta(t). \tag{19}$$

This is evaluated by the function `wfromt.laplace`.

The routine `wandafromx`

For the Laplace distribution, it is possible to estimate the scale factor by marginal maximum likelihood. In order to maintain the constraint $0 \le t \le \sqrt{2 \log n}$, and also to gain numerical stability, we work in terms of t and a. The function to be minimized is then

$$S^*(t, a) = -\sum_{i=1}^{n} \log\{1 + w(t)\beta(x_i, a)\}$$

where $w(t)$ is given by (19). This function is evaluated by the routine `negloglik.laplace`. The optimization is carried out subject to the box constraints $0 \le t \le \sqrt{2 \log n}$ and, to avoid unrealistic values of a, $0.04 \le a \le 3$. The optimization routine used is `optim` in R and `nlminb` in S-PLUS.

6.4. Quasi-Cauchy prior

We now turn to the quasi-Cauchy prior. The calculations for this case largely depend on standard Bayesian manipulations from the definition given in (3). Some fuller details are given in the next subsection.

The routine `beta.cauchy`

Conditional on $\Theta = \theta$, the distribution of μ is $N(0, \theta^{-1} - 1)$ and so the distribution of X is $N(0, \theta^{-1})$. Integrating out over the distribution of Θ for $0 < \Theta < 1$, we obtain

$$g(x) = (2\pi)^{-\frac{1}{2}} x^{-2}(1 - e^{-\frac{1}{2}x^2})$$

so that

$$\beta(x) = x^{-2}\{\phi(0)/\phi(x) - 1\} - 1.$$

The routine `postmean.cauchy`

Cognate to the Laplace calculations, the posterior weight $w_{\text{post}}(x)$ is found using the routine `beta.cauchy`. The key result for the determination of the posterior mean is then

$$\mu_1(x) = x(1 - e^{-\frac{1}{2}x^2})^{-1} - 2x^{-1}. \tag{20}$$

The routine `postmed.cauchy`

After some manipulation, for $\mu > 0$, we have

$$\tilde{F}_1(\mu|x) = (1 - e^{-\frac{1}{2}x^2})^{-1}\{\tilde{\Phi}(\mu - x) - x\phi(\mu - x) + (\mu x - 1)e^{\mu x - \frac{1}{2}x^2}\tilde{\Phi}(\mu)\}. \tag{21}$$

Suppose $x > 0$. If it has a positive solution, the equation $\tilde{F}_1(\hat{\mu}(x; w)|x) = \{2w_{\text{post}}\}^{-1}$ in (11) can then be solved numerically for $\hat{\mu}(x; w)$. This can be expressed as the equation

$$-\tilde{\Phi}(\hat{\mu} - x) + x\phi(\hat{\mu} - x) - (\hat{\mu}x - 1)e^{\hat{\mu}x - \frac{1}{2}x^2}\tilde{\Phi}(\hat{\mu}) + \tfrac{1}{2}(1 - e^{-\frac{1}{2}x^2}) + \tfrac{1}{2}(1/w - 1)x^2 e^{-\frac{1}{2}x^2} = 0. \tag{22}$$

For positive x, this can be solved by a binary search algorithm over the range $[0, x]$ to find $\hat{\mu}$. Since the tail probability is decreasing as a function of μ there is necessarily at most one root, and the property that the posterior median is a shrinkage rule ensures that we need not look higher than x. If there is no zero in the interval then the posterior median is zero.

The routine `tfromw(prior="cauchy")`

The threshold will be given by setting $\hat{\mu} = 0$ and $x = t$ in (22), to yield

$$-\Phi(t) + t\phi(t) + \tfrac{1}{2} + \tfrac{1}{2}t^2 e^{-\frac{1}{2}t^2}(1/w - 1) = 0.$$

In order to find t from w, a numerical search is required. However, it is straightforward to express w in terms of t. In particular, we can find explictly the value of w for which t is equal to $\sqrt{2 \log n}$. This calculation is carried out by the routine `wfromt.cauchy`.

6.5. Further details of calculations for quasi-Cauchy prior

Because the results we have used above are not entirely straightforward to derive, in this section some additional details are provided. The various components of the part of the model for $\mu \neq 0$ are

$$\begin{aligned} (X|\mu) &\sim N(\mu, 1) \\ (\mu|\theta) &\sim N(0, \theta^{-1} - 1) \\ f(\theta) &= \tfrac{1}{2}\theta^{-\frac{1}{2}} \quad 0 < \theta < 1. \end{aligned}$$

By integrating out over μ, we have

$$(X|\theta) \sim N(0, \theta^{-1}).$$

Hence the marginal density g is given by

$$\begin{aligned} g(x) &= \frac{1}{\sqrt{2\pi}} \int_0^1 \theta^{\frac{1}{2}} e^{-\frac{1}{2}x^2\theta} \times \tfrac{1}{2}\theta^{-\frac{1}{2}} d\theta \\ &= \frac{1}{\sqrt{2\pi}} \int_0^1 \tfrac{1}{2} e^{-\frac{1}{2}x^2\theta} d\theta \\ &= \frac{1}{\sqrt{2\pi}} x^{-2}(1 - e^{-\frac{1}{2}x^2}). \end{aligned}$$

Now turn to the derivation of the posterior distribution of μ given x. We will need the two results, for $u > 0$,

$$\frac{1}{2\sqrt{2\pi}} \int_0^1 s^{-1/2} e^{-\frac{1}{2}u^2/s} ds = \phi(u) - u\tilde{\Phi}(u) \tag{23}$$

and

$$\frac{1}{2\sqrt{2\pi}} \int_0^1 s^{-3/2} e^{-\frac{1}{2}u^2/s} ds = u^{-1}\tilde{\Phi}(u). \tag{24}$$

By standard Bayesian theory for inference in the Normal distribution, we have

$$(\mu|X = x, \theta) \sim N((1 - \theta)x, (1 - \theta)).$$

Also by standard Bayesian manipulations

$$f(\theta|X = x) \propto f(x|\theta)f(\theta) \propto e^{-\frac{1}{2}x^2\theta}.$$

Integrating over the interval $[0, 1]$ to obtain the constant of proportionality yields

$$f(\theta|X = x) = \frac{\frac{1}{2}x^2 e^{-\frac{1}{2}x^2\theta}}{1 - e^{-\frac{1}{2}x^2}}.$$

For $x = 0$ the posterior density of θ is 1 on $[0, 1]$.

We therefore have

$$E(\mu|X = x) = \int_0^1 E(\mu|X = x, \theta) f(\theta|X = x) d\theta = \int_0^1 (1 - \theta) x f(\theta|X = x) d\theta$$

which leads to the posterior mean (20).

Now turn to the difficult calculation of the posterior distribution itself. Assume $u > 0$ and $x \neq 0$. Use a standard result of conditional probability, then integrate by parts, substitute in the integral, and use the results (23) and (24), to obtain

$$\begin{aligned} &(1 - e^{-\frac{1}{2}x^2}) P(\mu > u|X = x) = (1 - e^{-\frac{1}{2}x^2}) \int_0^1 f(x|\theta) P(\mu > u|x, \theta) d\theta \\ &= \int_0^1 \tfrac{1}{2} x^2 e^{-\frac{1}{2}x^2\theta} \tilde{\Phi}\left(\frac{u - (1 - \theta)x}{\sqrt{1 - \theta}} \right) d\theta \end{aligned} \tag{25}$$

$$= -e^{-\frac{1}{2}x^2\theta}\tilde{\Phi}\left(\frac{u-(1-\theta)x}{\sqrt{1-\theta}}\right)\Big|_0^1 \qquad (26)$$
$$-\tfrac{1}{2}\int_0^1 e^{-\frac{1}{2}x^2\theta}\phi\left(\frac{u-(1-\theta)x}{\sqrt{1-\theta}}\right)\{(1-\theta)^{-3/2}u+(1-\theta)^{-1/2}x\}d\theta$$
$$= \tilde{\Phi}(u-x) - \frac{1}{2\sqrt{2\pi}}e^{ux-\frac{1}{2}x^2}\int_0^1 (us^{-3/2}+xs^{-1/2})e^{-\frac{1}{2}u^2/s}ds$$
$$= \tilde{\Phi}(u-x) - e^{ux-\frac{1}{2}x^2}\{\tilde{\Phi}(u)+x\phi(u)-ux\tilde{\Phi}(u)\} \qquad (27)$$

Standard manipulations of the normal density, and use of the property that $P(\mu > -\infty|X = x)$ lead to the result given in equation (21).

For completeness, we note that for $x = 0$, the posterior distribution of μ is a mixture of $N(0, 1-\theta)$ distributions over a uniform distribution for θ. Performing the mixture integral, using the result (23), we find that the posterior density of μ given $x = 0$ is $\phi(u) - |u|\tilde{\Phi}(|u|)$. For $u > 0$, this yields

$$P(\mu > u|X = 0) = \tfrac{1}{2}(1+u^2)\tilde{\Phi}(u) - \tfrac{1}{2}u\phi(u),$$

while for $u = 0$ the limiting value $1/2$ for this expression is the correct probability.

For values of $u < 0$, appropriate tail probabilities can be found by using the relation $P(\mu < u|X = x) = P(\mu > -u|X = -x)$. Finally, for $u = 0$, a separate calculation of the integral in (25) shows that the result in (27) still holds. The need for these separate calculations is demonstrated, among other things, by the necessity of $u > 0$ for the evaluation of (26) to be correct as stated.

7. Extended weight dependence: some algorithmic issues

In this section, we set out algorithmic details underlying the two approaches considered in Section 5, monotonic and parametric dependence of the weight w_i on the index i.

7.1. Monotone weight dependence

To maximize $\sum_{i=1}^n \log\{(1-w_i)\phi(x_i)+w_i g(x_i)\}$ subject to the constraint $w_1 \geq w_2 \geq \ldots \geq w_n$, an iteratively reweighted isotone regression method was used. Published algorithms for this problem do not seem to be readily available, so we have provided a routine for the purpose; the underlying algorithm is set out here.

Define $\beta_i = \beta(x_i)$ and note that the negative of the objective function differs by a constant depending only on the observations from

$$U(w) = -\sum_i \log(1+w_i\beta_i).$$

Given a current estimate (w_i^0), let $a_i^0 = (1+w_i^0\beta_i)/\beta_i = \beta_i^{-1} + w_i^0$. By a Taylor expansion,

$$U(w) - U(w_0) \approx \sum_i \{-(a_i^0)^{-1}(w_i - w_i^0) + \tfrac{1}{2}(a_i^0)^{-2}(w_i - w_i^0)^2\}$$
$$= \sum_i \tfrac{1}{2}(a_i^0)^{-2}\{w_i - (w_i^0 + a_i^0)\}^2 + C(w^0)$$

Therefore the next stage in an optimization of U is obtained by a weighted least squares isotonically decreasing fit to the values $w_i^0 + a_i^0$ with weights $(a_i^0)^{-2}$. The cycle of iteratively reweighting and re-estimating the isotone regression is repeated to convergence, which is typically achieved quite rapidly. Setting all the w_i^0 to 1 initially works well.

The weighted least squares isotone fit is carried out using the routine `isotone`; given a sequence of values b_i and of weights ω_i, the least squares isotone increasing regression finds the monotone increasing sequence b_i^* for which $\sum \omega_i(b_i - b_i^*)^2$ is minimized. It does this by the standard pool-adjacent violators algorithm, modified to allow for weights ω_i. Because of the weighting, it is convenient to carry out the weighted isotone regression as follows, for an increasing regression; for decreasing regression, work with the negatives of the data and negate the result:

1. Begin with the data b_i themselves
2. Find all local maxima and minima in the current sequence, and locate each decreasing subsequence, as the sequence between a local maximum and the succeeding local minimum.
3. Keeping track of the address of the beginning of the subsequence in the *original* sequence b_i, replace the subsequence by a single value equal to the weighted average of the subsequence, and the weight by the sum of the weights over the subsequence. This yields three sequences: values b_i', weights ω_i', and addresses j_i.
4. Iterate to termination, achieving a sequence $b_i^\dagger$ with addresses j_i.
5. Reconstruct a sequence of the same length as the original, by setting $b_k^* = b_i^\dagger$ for $j_i \leq k < j_{i+1}$.

7.2. Estimation of the weight in the case of parametric dependence

We now set out the algorithm for estimating the parameter ζ in the model considered in Section 5.2. Define $\zeta_{lo} = w_{lo}(\max c_i)^{-1}$ and $\zeta_{hi} = w_{hi}(\min c_i)^{-1}$. If $\zeta < \zeta_{lo}$ then all the w_i will be w_{lo} and if $\zeta > \zeta_{hi}$ the all the w_i will be 1, regardless of how far outside the interval ζ lies. Therefore, we need only consider ζ only over the interval $[\zeta_{lo}, \zeta_{hi}]$. Our strategy for finding the global maximum of ℓ is first to find all local *minima*, to search between consecutive local minima to find all local maxima, and then to evaluate the function ℓ at all local maxima to find which one is the global maximum.

Define

$$\beta_i = g(z_i)/\phi(z_i) - 1.$$

Then $\beta_i > -1$ for all i, and

$$\ell(\zeta) = \sum_i \log\{1 + \beta_i w_i(\zeta)\} + C(z)$$

where $C(z)$ is a constant that depends on the z_i but not on the parameter ζ. Each function $w(\zeta)$ is continuous and piecewise linear, with derivative c_i on the interval $(c_i^{-1}w_{lo}, c_i^{-1})$ and zero outside this interval. Hence $\ell'(\zeta)$ is continuous except on the set $\mathcal{J}$ of all points equal to $c_i^{-1}w_{lo}$ or c_i^{-1} for some i.

1. Demonstrate that the log likelihood is piecewise concave between points in $\mathcal{J}$: For any particular ζ, define

$$\mathcal{I}(\zeta) = \{i : c_i^{-1} w_{lo} < \zeta < c_i^{-1}\}.$$

Leaving aside points ζ in $\mathcal{J}$ for the moment, $\mathcal{I}(\zeta)$ is the set of i for which $w_i'(\zeta)$ is nonzero. The derivative $\ell'(\zeta)$ will have jumps at the points in $\mathcal{J}$, but otherwise will be given by

$$\ell'(\zeta) = \sum_{i \in \mathcal{I}(\zeta)} \frac{c_i \beta_i}{1 + \zeta c_i \beta_i}.$$

Between two consecutive points in $\mathcal{J}$, the set $\mathcal{I}(\zeta)$ will not change, and so the function $\ell'(\zeta)$ will be decreasing. Therefore local minima of $\ell(\zeta)$ can only occur at points ζ in $\mathcal{J}$.

2. Check which points in $\mathcal{J}$ are local minima: To deal with points in $\mathcal{J}$, define

$$\mathcal{I}^-(\zeta) = \{i : c_i^{-1} w_{lo} < \zeta \le c_i^{-1}\} \qquad \text{and} \qquad \mathcal{I}^+(\zeta) = \{i : c_i^{-1} w_{lo} \le \zeta < c_i^{-1}\}.$$

By elementary properties of the piecewise linear functions $w_i(\zeta)$, we then have, for all ζ,

$$\ell'(\zeta^-) = \sum_{i \in \mathcal{I}^-(\zeta)} \frac{c_i \beta_i}{1 + \zeta c_i \beta_i} \qquad \text{and} \qquad \ell'(\zeta^+) = \sum_{i \in \mathcal{I}^+(\zeta)} \frac{c_i \beta_i}{1 + \zeta c_i \beta_i}.$$

If a point ζ in $\mathcal{J}$ is a local minimum, then $\ell'(\zeta^-) \le 0$ and $\ell'(\zeta^+) \ge 0$, and so we locate all the local minima of ℓ by checking the signs of the left and right derivatives at each point in $\mathcal{J}$. Let M be the number of these local minima, and sort them into order as $\zeta_1 < \zeta_2 < \ldots < \zeta_M$.

3. Locate the local maximum between each consecutive pair of local minima: Consider each of the $M - 1$ intervals $[\zeta_j, \zeta_{j+1}]$ in turn. Within each of these the derivative $\ell'(\zeta)$ will cross (downwards) across zero at exactly one point, and this crossing point ζ_j^{max} can be found by a binary search. Our use of weak inequalities in constructing the list of local minima ensures that both the left and right derivatives are strictly positive on the interval (ζ_j, ζ_j^{max}) and strictly negative on the interval $(\zeta_j^{max}, \zeta_{j+1})$. For definiteness, the binary search is on the value of the right derivative. Note that $\ell'(\zeta+)$ may not be strictly decreasing across the interval; there may be jumps upwards, but none of these will be a jump across zero, because the interval contains no local minima.

The smallest and largest points in $\mathcal{J}$, ζ_{lo} and ζ_{hi}, need special consideration. If $\ell'(\zeta_{lo}+) > 0$ then ζ_{lo} is a local minimum, and is treated in the usual way. On the other hand if $\ell'(\zeta_{lo}+) \le 0$ then ζ_{lo} is a local maximum and is added to the list of local maxima without the need for a binary search. Similarly, ζ_{hi} is a local minimum if $\ell'(\zeta_{hi}-) < 0$ and a local maximum otherwise.

4. Find the global maximum: We now evaluate $\ell(\zeta_j^{max})$ for each j, and find the j for which this is greatest. The corresponding ζ is the global maximum.

8. Concluding remarks

The empirical Bayes method set out in this paper and implemented in the **EbayesThresh** package has wide potential applicability. There are increasingly many contexts where, implicitly or explicitly, one is estimating a large number of parameters and it is necessary or advisable to take advantage of possible sparsity in the parameter set. While it was the wavelet context that gave the authors the original motivation for investigating this methodology, both theoretical and practical considerations have already shown that it has much wider relevance, and it is our hope that the **EbayesThresh** package will enable this potential to be realised.

This work was supported in part by NIH EB1988-09 and NSF DMS 00-72661.

References

Antoniadis A, Jansen M, Johnstone IM, Silverman BW (2004). *EbayesThresh: MATLAB software for Empirical Bayes thresholding*. URL http://www-lmc.imag.fr/lmc-sms/Anestis.Antoniadis/EBayesThresh.

Bruce A, Gao HY (1996). *Applied Wavelet Analysis with S-PLUS*. Springer-Verlag.

Coifman RR, Donoho DL (1995). "Translation-Invariant De-Noising." In A Antoniadis (ed.), "Wavelets and Statistics," Lecture Notes in Statistics. Springer.

Donoho DL, Johnstone IM (1994). "Spatial Adaptation via Wavelet Shrinkage." *Biometrika*, **81**, 425–455.

Jansen M, Nason GP, Silverman BW (2004). "Multivariate nonparametric regression using lifting." Submitted for publication.

Johnstone IM, Silverman BW (1997). "Wavelet Threshold estimators for data with correlated noise." *Journal of the Royal Statistical Society, Series B*, **59**, 319–351.

Johnstone IM, Silverman BW (2004). "Needles and straw in haystacks: Empirical Bayes estimates of possibly sparse sequences." *Annals of Statistics*, **32**, 1594–1649.

Johnstone IM, Silverman BW (2005). "Empirical Bayes selection of wavelet thresholds." *Annals of Statistics*, **33**, 000–000.

Nason GP (1996). "Wavelet shrinkage using cross-validation." *Journal of the Royal Statistical Society, Series B*, **58**, 463–479.

Nason GP (1998). *WaveThresh3 Software*. Department of Mathematics, University of Bristol, UK. URL http://www.stats.bris.ac.uk/~wavethresh/.

Nason GP (2004). *wavethresh: Software to perform wavelet statistics and transforms*. URL http://cran.r-project.org/src/contrib/Descriptions/wavethresh.html.

Silverman BW (2004). *ebayesthresh: Empirical Bayes thresholding and related methods*. URL http://cran.r-project.org/src/contrib/Descriptions/ebayesthresh.html.

Whitcher B (2004). *The waveslim package: Basic wavelet routines for one-, two- and three-dimensional signal processing*. URL http://cran.r-project.org/src/contrib/Descriptions/waveslim.html.

Affiliation:

Iain M. Johnstone
Department of Statistics
Stanford University
Stanford CA 94305-4065, USA
E-mail: imj@stat.stanford.edu
URL: http://www-stat.stanford.edu/people/faculty/johnstone/

Bernard W. Silverman
St Peter's College
Oxford OX1 2DL, United Kingdom
E-mail: bernard.silverman@spc.ox.ac.uk
URL: http://www.bernardsilverman.com

Journal of Statistical Software
MMMMMM YYYY, Volume VV, Issue II.
http://www.jstatsoft.org/

Submitted: yyyy-mm-dd
Accepted: yyyy-mm-dd