

# EbayesThresh: R and S-PLUS programs for Empirical Bayes thresholding

Iain M. Johnstone and Bernard W. Silverman

Stanford University and University of Bristol

August 28, 2002

## Abstract

This report sets out a package of R and S-PLUS routines that implement a class of Empirical Bayes thresholding methods. The prior considered for each parameter in a sequence is a mixture of an atom of probability at zero and a heavy-tailed density. The package allows for the heavy-tailed density to be either a Laplace (double exponential) density or else a mixture of normal distributions with tail behavior similar to that of the Cauchy distribution. The mixing weight, or sparsity parameter, is chosen by marginal maximum likelihood. In the case of the Laplace density, the scale parameter may also be chosen 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 report gives details of the calculations needed for implementing the procedures. In addition, 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. Finally, a routine is provided that applies the method level by level to wavelet transforms as obtained using S+WAVELETS.

## 1 Introduction

### 1.1 Background

There are many statistical problems where the object of interest is a sequence of parameters  $\mu_i$  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  $\epsilon_i$  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 & Silverman (2002*b*) for further discussion.

Clearly, without some knowledge of the  $\mu_i$  we are not going to be able to estimate them very efficiently, and the method implemented in the package associated with this report takes advantage of possible sparsity in the sequence. A natural approach to this problem is *thresholding*: if the absolute value of a particular  $X_i$  exceeds some threshold  $t$  then it is taken to correspond to a nonzero  $\mu_i$  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 quite 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 general, one would hope that such methods would estimate thresholds that stably reflect the gradation from sparse to dense signals as the scale changes from fine to coarse. The papers Johnstone & Silverman (2002*b,a*) show, by both theoretical and practical consideration, that a suitable Empirical Bayes approach has many good properties.

## 1.2 Bayesian approaches

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

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

The nonzero part of the prior,  $\gamma$ , is assumed to be a fixed unimodal symmetric density. Particular possibilities for the function  $\gamma$  are set out in Section 2 below. For each of these possibilities, the procedures we set out are entirely feasible computationally.

Suppose  $\mu$  has the prior distribution (2) and  $X \sim N(\mu, 1)$ . We can then find the posterior distribution of  $\mu$  conditional on  $X = x$ . Let  $\hat{\mu}(x; w)$  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  $|x| \leq t(w)$ . Once  $w$  has been specified, there are other possible estimation rules, for example the posterior mean  $\tilde{\mu}(x; w)$  of  $\mu$  given  $X = x$ , or hard or soft thresholding with threshold  $t(w)$ .

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  $\mu_i$ . 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.

## 1.3 Choosing the threshold

It is very important to make a good choice of mixing weight  $w$ , or equivalently of threshold  $t(w)$ . Assume the  $X_i$  are independent; we can 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

$$\ell(w) = \sum_{i=1}^n \log\{(1 - w)\phi(X_i) + wg(X_i)\} \quad (3)$$

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, provided the function  $g$  is tractable.

Our basic approach is then an Empirical Bayes one. We use 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  $\mu_i$  by a Bayesian procedure using this value of  $w$ . In our implementation the cost of both parts of the procedure is linear in the number of observations considered.

#### 1.4 Extensions

The same approach can be used to estimate other parameters of the prior. In particular, if a scale parameter  $a$  is incorporated by considering a prior density  $(1 - w)\delta_0(\mu) + wa\gamma(a\mu)$ , define  $g_a$  to be the convolution of  $a\gamma(a\cdot)$  with the normal density. Then both  $a$  and  $w$  can be estimated by finding the maximum over both parameters of

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

If the observations have variance equal to  $\sigma^2$  rather than 1, then the natural approach is to renormalize the data by dividing by  $\sigma$  and then to multiply the resulting estimates of the means by  $\sigma$ . If the variance of the data is not known, then the package allows for its estimation from the median absolute deviation from zero. The motivation for this is that provided the sequence  $\mu_i$  is only reasonably sparse, the median of the absolute deviations will not be affected by those observations that have nonzero means  $\mu_i$ , but clearly some care may be appropriate according to the context.

Though the basic approach is much more widely applicable, our original motivation was function estimation using wavelets. In the wavelet context, 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. Full details are given in Johnstone & Silverman (2002a).

Another extension is to allow the threshold to increase as  $i$  increases, reflecting the notion that early  $\mu_i$  have a reasonably large probability of being nonzero, but that as one proceeds along the sequence nonzero  $\mu_i$  become rarer. Detailed consideration of problems of this kind is in progress, but software based on an iterated least squares monotone regression algorithm is available as part of the package. See Section 4 for more details.

## 1.5 The software

At present the algorithms are implemented in R and S-PLUS, and the programs have been written to work as far as possible in either language. The source code is available direct from the EbayesThres web site associated with this report (at the time of writing [www.statistics.bristol.ac.uk/~bernard/ebayesthresh](http://www.statistics.bristol.ac.uk/~bernard/ebayesthresh)). A MATLAB implementation is in progress.

The main routines are documented on pages linked to the EbayesThres web site, and included as an appendix to the printed version of this report. Most users will only need to call the master routine `ebayesthresh`, and for most purposes the default choices of parameters should suffice.

For wavelet thresholding, the routine `ebayesthresh.wavelet` can be applied in S-PLUS in conjunction with the module S+WAVELETS. The comments provided for this routine should allow users of other wavelet software easily to write an appropriate translation.

The software is copyright, but may be freely used for academic purposes provided it is acknowledged. Commercial users should contact the authors to discuss licence terms. Strictly speaking, the routines are provided for illustrative purposes only.

## 2 Mixture priors

### 2.1 Priors with heavy tails

Particular heavy-tailed densities that we shall consider for the nonzero part of the prior distribution are the Laplace density with scale parameter  $a > 0$

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

and the mixture density given by

$$\mu|\Theta = \theta \sim N(0, \theta^{-1} - 1) \text{ with } \Theta \sim \text{Beta}(\frac{1}{2}, 1). \quad (4)$$

More explicitly, the latter density for  $\mu$  has

$$\gamma(u) = (2\pi)^{-1/2} \{1 - |u| \tilde{\Phi}(|u|) / \phi(u)\} \quad (5)$$

and has tails that decay as  $u^{-2}$ , the same weight as those of the Cauchy distribution. For this reason we refer to the density (5) as the *quasi-Cauchy* density.

In both cases the posterior distribution of  $\mu$  given an observed  $X$ , and the marginal distribution of  $X$  are tractable, so that the choice of  $w$  by marginal maximum likelihood,

and the estimation of  $\mu$  by posterior mean or median, can be performed in practice, as outlined in the following paragraphs. We begin by setting out generic calculations for the relevant quantities, and then give specific details for the Laplace and quasi-Cauchy priors.

## 2.2 Generic calculations

**Posterior probability that parameter is nonzero** Define

$$\beta(x) = g(x)/\phi(x) - 1. \quad (6)$$

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

$$w_{\text{post}}(x) = wg(x)/\{wg(x) + (1-w)\phi(x)\} = (1 + \beta(x))/(w^{-1} + \beta(x)), \quad (7)$$

and hence can be found using the function  $\beta$  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  $\mu$  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

$$\begin{aligned} \hat{\mu}(x; w) &= 0 && \text{if } w_{\text{post}}(x)\tilde{F}_1(0|x) \leq \frac{1}{2} \\ \tilde{F}_1(\hat{\mu}(x; w)|x) &= \{2w_{\text{post}}(x)\}^{-1} && \text{otherwise} \end{aligned} \quad (8)$$

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

$$S(w) = \sum_{i=1}^n \frac{g(x_i) - \phi(x_i)}{(1-w)\phi(x) + wg(x)} = \sum_{i=1}^n \frac{\beta(x_i)}{1 + w\beta(x_i)} = \sum_{i=1}^n \frac{\beta_i}{1 + w\beta_i} \quad (9)$$

where  $\beta_i = \beta(x_i)$ . Define  $w_{lo}$  to be such that  $\tau(w_{lo}) = \sqrt{2 \log n}$ . 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  $\beta_i$  have been found, and the endpoint  $w_{lo}$ , the algorithm does not depend on the prior at all, only on the values  $\beta_i$ . Therefore to cater for a new prior, it is only necessary to have a routine to find the function  $\beta$ , and to evaluate the limiting weight  $w_{lo}$ .

### 2.3 Bayes factor threshold

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

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

Thus  $\tau_p(w)$  is the largest value of the observed data for which the estimated  $\mu$  will be zero, if the estimate is carried out by the posterior median.

In some cases, simpler calculations may arise from the use of the Bayes factor threshold, defined as the value  $\tau_b(w) > 0$  such that

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

We will have

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

and so  $\tau_p(w) \geq \tau_b(w)$ . But as  $w$  decreases, the two thresholds will become closer together, as illustrated in Figure 1. To calculate  $\tau_b$ , we have, working in odds rather than probabilities,

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

Rearranging, this yields

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

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

$$\beta(\tau_b) = w^{-1} - 2. \tag{10}$$

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  $\beta$  in order to find  $w$  from  $\tau$ .

To find  $\tau$  from  $w$ , on the other hand, provided  $w \leq 1/(2 + \beta(0))$ , we can solve equation (10) by a binary search, because of the monotonicity of  $\beta$ . If we have an explicit formula for  $\beta^{-1}$ , then we can calculate directly

$$\tau_b(w) = \beta^{-1}(w^{-1} - 2). \quad (11)$$

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  $\tau_b = 0$ .

### 3 Calculations for specific priors

The calculations set out above show that the key quantities are the marginal density  $g$ , the function  $\beta$ , the mean function  $\mu_1(x)$ , and the tail conditional probability function  $\bar{F}_1$ . In this section, much fuller details are given for the Laplace prior and for the quasi-Cauchy prior. The subheadings in each case give the names of the relevant routines in the software package.

#### 3.1 Laplace prior

##### The routine `beta.laplace`

For the Laplace distribution prior, we have

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

and therefore

$$\beta(x) = \beta(x, a) = \frac{1}{2}a \left\{ \frac{\Phi(x-a)}{\phi(x-a)} + \frac{\tilde{\Phi}(x+a)}{\phi(x+a)} \right\} - 1. \quad (12)$$

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 numerical problems. Also, it is best to write a routine for positive  $x$  and to use the symmetry of  $\beta$  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 (9). The routine `wfromx` does the calculations for the actual maximization of the likelihood, in the manner set out in Section 2.2; 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 2.2, the calculation (7) 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  $\mu$  conditional on  $\mu \neq 0$ . We have

$$f_1(\mu|x) = \begin{cases} e^{ax}\phi(\mu - x - a)/\{e^{-ax}\Phi(x - a) + e^{ax}\tilde{\Phi}(x + a)\} & \text{if } \mu \leq 0 \\ e^{-ax}\phi(\mu - x + a)/\{e^{-ax}\Phi(x - a) + e^{ax}\tilde{\Phi}(x + a)\} & \text{if } \mu > 0 \end{cases} \quad (13)$$

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

$$\mu_1(x) = x - \frac{a\{e^{-ax}\Phi(x - a) - e^{ax}\tilde{\Phi}(x + a)\}}{e^{-ax}\Phi(x - a) + e^{ax}\tilde{\Phi}(x + a)}. \quad (14)$$

The posterior mean is now equal to  $w_{\text{post}}(x)\mu_1(x)$ , as set out in Section 2.2.

### 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 (8),

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

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

$$z_0 = a^{-1}\phi(x - a)\{w^{-1} + \beta(x)\}. \quad (15)$$

As  $x \rightarrow \infty$ , 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 (15) 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  $\tau_p(w)$  such that  $\tau_p - a - \Phi^{-1}(z_0) = 0$ . Therefore

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

an equation that can be solved by binary search to find the  $\tau_p$  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  $w = w(\tau_p)$  in terms of  $\tau_p$ :

$$w(\tau_p)^{-1} = a\Phi(\tau_p - a)/\phi(\tau_p - a) - \beta(\tau_p). \quad (16)$$

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  $0 \leq \tau_p \leq \sqrt{2 \log n}$ , and also to gain numerical stability, we work in terms of  $\tau_p$  and  $a$ . The function to be minimized is then

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

where  $w(\tau)$  is given by (16). This function is evaluated by the routine `negloglik.laplace`. The optimization is carried out subject to the box constraints  $0 \leq \tau \leq \sqrt{2 \log n}$  and, to avoid unrealistic values of  $a$ ,  $0.04 \leq a \leq 3$ . The optimization routine used is `optim` in R and `nlminb` in S-PLUS.

## 3.2 Quasi-Cauchy prior

The calculations for the quasi-Cauchy prior largely depend on standard Bayesian manipulations from the definition given in (4). Some fuller details are given in the next subsection.

### The routine `beta.cauchy`

Conditional on  $\Theta = \theta$ , the distribution of  $X$  is  $N(0, \theta^{-1})$ . Integrating out over the distribution of  $\Theta$  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

$$\mu_1(x) = x(1 - e^{-\frac{1}{2}x^2})^{-1} - 2x^{-1}. \quad (17)$$

### The routine `postmed.cauchy`

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

$$\tilde{F}_1(\mu|x) = (1 - e^{-\frac{1}{2}x^2})^{-1} \{ \tilde{\Phi}(\mu - x) - x\phi(\mu - x) + (\mu x - 1)e^{\mu x - \frac{1}{2}x^2} \tilde{\Phi}(\mu) \}. \quad (18)$$

Suppose  $x > 0$ . If it has a positive solution, the equation  $\tilde{F}_1(\hat{\mu}(x;w)|x) = \{2w_{\text{post}}\}^{-1}$  in (8) can then be solved numerically for  $\hat{\mu}(x;w)$ . This can be expressed as the equation

$$-\tilde{\Phi}(\hat{\mu} - x) + x\phi(\hat{\mu} - x) - (\hat{\mu}x - 1)e^{\hat{\mu}x - \frac{1}{2}x^2} \tilde{\Phi}(\hat{\mu}) + \frac{1}{2}(1 - e^{-\frac{1}{2}x^2}) + \frac{1}{2}(1/w - 1)x^2 e^{-\frac{1}{2}x^2} = 0. \quad (19)$$

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  $\mu$  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  $x = \tau_p$  in (19), to yield

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

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

## 3.3 Further details of calculations for 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

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

By integrating out over  $\mu$ , we have

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

Hence the marginal density  $g$  is given by

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

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

$$\frac{1}{2\sqrt{2\pi}} \int_0^1 s^{-1/2} e^{-\frac{1}{2}u^2/s} ds = \phi(u) - u\tilde{\Phi}(u) \tag{20}$$

and

$$\frac{1}{2\sqrt{2\pi}} \int_0^1 s^{-3/2} e^{-\frac{1}{2}u^2/s} ds = u^{-1}\tilde{\Phi}(u). \tag{21}$$

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  $\theta$  is 1 on  $[0, 1]$ .

We therefore have

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

which leads to the posterior mean (17).

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 (20) and (21), to obtain

$$\begin{aligned} (1 - e^{-\frac{1}{2}x^2})P(\mu > u|X = x) &= (1 - e^{-\frac{1}{2}x^2}) \int_0^1 f(x|\theta)P(\mu > u|x, \theta)d\theta \\ &= \int_0^1 \frac{1}{2}x^2 e^{-\frac{1}{2}x^2\theta} \tilde{\Phi}\left(\frac{u - (1 - \theta)x}{\sqrt{1 - \theta}}\right) d\theta \end{aligned} \quad (22)$$

$$= -e^{-\frac{1}{2}x^2\theta} \tilde{\Phi}\left(\frac{u - (1 - \theta)x}{\sqrt{1 - \theta}}\right) \Big|_0^1 \quad (23)$$

$$\begin{aligned} & - \frac{1}{2} \int_0^1 e^{-\frac{1}{2}x^2\theta} \phi\left(\frac{u - (1 - \theta)x}{\sqrt{1 - \theta}}\right) \{(1 - \theta)^{-3/2}u + (1 - \theta)^{-1/2}x\}d\theta \\ &= \tilde{\Phi}(u - x) - \frac{1}{2\sqrt{2\pi}} e^{ux - \frac{1}{2}x^2} \int_0^1 (us^{-3/2} + xs^{-1/2})e^{-\frac{1}{2}u^2/s} ds \\ &= \tilde{\Phi}(u - x) - e^{ux - \frac{1}{2}x^2} \{\tilde{\Phi}(u) + x\phi(u) - ux\tilde{\Phi}(u)\} \end{aligned} \quad (24)$$

Standard manipulations of the normal density, and use of the property that  $P(\mu > -\infty|X = x)$  lead to the result given in equation (18).

For completeness, we note that for  $x = 0$ , the posterior distribution of  $\mu$  is a mixture of  $N(0, 1 - \theta)$  distributions over a uniform distribution for  $\theta$ . Performing the mixture integral, using the result (20), we find that the posterior density of  $\mu$  given  $x = 0$  is  $\phi(u) - |u|\tilde{\Phi}(|u|)$ . For  $u > 0$ , this yields

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

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 (22) shows that the result in (24) still holds. The need for these separate calculations is demonstrated, among other things, by the necessity of  $u > 0$  for the evaluation of (23) to be correct as stated.

## 4 Monotone weight estimation

Consider, now, the situation where we have a sequence  $\mu_i$  believed to become more sparse as  $i$  increases. For example,  $\mu_i$  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, (?) construct just such a basis for the analysis of data observed on an irregular set of points in two dimensions.

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

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

where  $\delta$  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

$$\ell(w_1, \dots, w_n) = \sum_{i=1}^n \log\{(1 - w_i)\phi(x_i) + w_i g(x_i)\} \quad (25)$$

subject to the constraint  $w_1 \geq w_2 \geq \dots \geq w_n$ . Once the weights have been estimated, we can, as before, estimate each  $\mu_i$  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  $\sqrt{2 \log n}$ . The routine makes use of a subsidiary routine `isotone`, which calculates weighted least squares isotone regression; given a sequence of values  $b_i$  and of weights  $\omega_i$ , 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 a standard algorithm called the pool-adjacent-violators algorithm, though the implementation differs from most in allowing for weights  $\omega_i$ .

The maximization of  $S(w_1, \dots, w_n)$  is carried out by an iteratively reweighted isotone regression. Define  $\beta_i = \beta(x_i)$  and note that the negative log likelihood 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, we then have

$$\begin{aligned} U(w) - U(w_0) &\approx \sum_i \left\{ -(a_i^0)^{-1} (w_i - w_i^0) + \frac{1}{2} (a_i^0)^{-2} (w_i - w_i^0)^2 \right\} \\ &= \sum_i \frac{1}{2} (a_i^0)^{-2} \{w_i - (w_i^0 + a_i^0)\}^2 + C(w^0) \end{aligned}$$

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 reestimating the isotone regression is repeated to convergence, which is typically achieved quite rapidly. Setting all the  $w_i^0$  to 1 initially works well.

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. At each stage, locate all decreasing subsequences in the current sequence, by finding all local maxima and minima; each decreasing subsequence is 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, a sequence of values  $b'_i$ , of weights  $\omega'_i$ , and of 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  $j_i \leq k < j_{i+1}$ .

## References

- Johnstone, I. M. & Silverman, B. W. (2002*a*), Empirical Bayes selection of wavelet thresholds, submitted for publication.
- Johnstone, I. M. & Silverman, B. W. (2002*b*), Needles and straw in haystacks: Empirical Bayes estimates of possibly sparse sequences, submitted for publication.

EbayesThresh: R and S-PLUS software for Empirical Bayes thresholding.

by Iain M. Johnstone and Bernard W. Silverman, August 2002

## Appendix 1: Description of software and individual help pages

These documents and listings are available online at  
<http://www.statistics.bristol.ac.uk/~bernard/ebayesthresh>

This software complements the paper "Needles and straw in haystacks: Empirical Bayes approaches to thresholding a possibly sparse sequence" and "Empirical Bayes selection of wavelet thresholds" by Iain M. Johnstone and Bernard W. Silverman, submitted for publication 2002. A paper giving a general description of the software and some details both of the general methodology and of some specific technical matters is available [here](#).

To obtain the software, save the file obtained by clicking [here](#), and load it into R or S-PLUS by using the `source` function.

The master routine in the software for processing single sequences of data is the function `ebayesthresh`. Click [here](#) for instructions for its use. A fuller list of links to help files for individual functions is given below. To download all the help files and other files as a single zip file, click [here](#).

The routine [ebayesthresh.wavelet](#) applies the approach to wavelet transforms obtained within the S+WAVELETS module or using the WaveThresh package. If wavelet transforms are obtained using other software, the routine will not be applicable directly, but should still provide a model for the user to write their own wavelet smoothing routine making use of the function `ebayesthresh`.

The software may be downloaded and used freely for academic purposes, provided its use is acknowledged. Commercial use is not allowed without the permission of the [author](#). It is hoped that corresponding MATLAB functions will be available in the future. Please bring any problems or errors to the author's attention.

<a href="#">ebayesthresh</a>	Performs various empirical Bayes thresholding functions on single sequences of data
<a href="#">ebayesthresh.wavelet</a>	Processes wavelet transforms obtained from data using the S+WAVELETS module or the WaveThresh package, and reconstructs to obtain function estimates.
<a href="#">wfromx</a>	Estimates prior mixing weight from data
<a href="#">wandafromx</a>	Estimates prior mixing weight and scale factor from data

<a href="#"><u>tfromw</u></a>	Finds threshold corresponding to weight
<a href="#"><u>wfromt</u></a>	Finds weight corresponding to threshold
<a href="#"><u>tfromx</u></a>	Estimates threshold from data
<a href="#"><u>postmed</u></a>	Finds posterior median estimate from data
<a href="#"><u>postmean</u></a>	Finds posterior mean estimate from data
<a href="#"><u>threshold</u></a>	Performs hard or soft thresholding
<a href="#"><u>vecbinsolv</u></a>	Vector version of binary search equation solving algorithm
<a href="#"><u>wmonfromx</u></a>	Estimates weight sequence from data, under constraint that weight is non-increasing sequence
<a href="#"><u>isotone</u></a>	Weighted least squares isotone regression
<a href="#"><u>beta.cauchy</u></a>	Finds the function beta for the quasi-Cauchy prior
<a href="#"><u>beta.laplace</u></a>	Finds the function beta for the Laplace prior

`beta.cauchy`

## Evaluate the function beta for the quasi-Cauchy prior

### DESCRIPTION

Given a value or vector of values, find the value(s) of the function beta, defined as  $(g/\phi - 1)$ , where  $g$  is the marginal density conditional on the mean being nonzero and having a quasi-Cauchy prior.

### USAGE

`beta.cauchy(x)`

### REQUIRED ARGUMENTS

`x` a data value or a vector of data

### VALUE

A vector the same length as `x` is returned, containing the value(s) `beta(x)`.

### SEE ALSO

[beta.laplace](#)

### EXAMPLE

```
bb <- beta.cauchy(x)
```

`beta.laplace`

## Evaluate the function beta for the Laplace prior

### DESCRIPTION

Given a value or vector of values, find the value(s) of the function beta, defined as  $(g/\phi - 1)$ , where  $g$  is the marginal density conditional on the mean being nonzero and having a -Laplace prior.

### USAGE

```
beta.laplace(x, a=1)
```

### REQUIRED ARGUMENTS

`x` a data value or a vector of data

### OPTIONAL ARGUMENT

`a` the scale parameter of the Laplace prior (default value = 1)

### VALUE

A vector the same length as `x` is returned, containing the value(s) `beta(x)`.

### SEE ALSO

[beta.cauchy](#)

### EXAMPLE

```
bb <- beta.laplace(x)
```

ebayesthresh

## Performs Empirical Bayes thresholding on a sequence

### DESCRIPTION

Given a sequence of data, performs Empirical Bayes thresholding, as discussed in Johnstone and Silverman (2002a).

### USAGE

```
ebayesthresh(x, prior = "laplace", a = 0.5, bayesfac = F, sdev = 1,  
verbose = F, threshrule = "median")
```

### REQUIRED ARGUMENT

`x` a data value or a vector of data

### OPTIONAL ARGUMENT

`prior` the prior to be used conditional on the mean being nonzero. Possible values are "laplace" for the Laplace prior and "cauchy" for the quasi-Cauchy prior.

`a` the scale factor if the Laplace prior is used. If, on entry, `a=NA` and `prior="laplace"`, then the scale parameter `a` will be estimated by marginal maximum likelihood. If the quasi-Cauchy prior is used then this argument is ignored.

`bayesfac` if `bayesfac=T` then whenever a threshold is explicitly calculated, the Bayes factor threshold will be used, viz the value of the data such that the posterior probability that the underlying mean is zero is exactly 0.5. Otherwise the posterior median threshold will be used.

`sdev` the sampling standard deviation of the data `x`. If, on entry, `sdev=NA`, then the standard deviation will be estimated using the median absolute deviation from zero, as `mad(x, center=0)`.

`verbose` controls the level of output. See below.

`threshrule` specifies the thresholding rule to be applied to the data. Possible values are "median" (use the posterior median); "mean" (use the posterior mean); "hard" (carry out hard thresholding); "soft" (carry out soft thresholding); "none" (find various parameters, but do not carry out any thresholding).

### VALUE

If `verbose=F`, a vector giving the values of the estimates of the underlying mean vector.

If `verbose=T`, a list with the following elements:

`muhat` the estimated mean vector (omitted if `threshrule="none"`)

`x` the data vector as supplied

`threshold.sdevscale` the threshold as a multiple of the standard deviation `sdev`

`threshold.origscale` the threshold measured on the original scale of the data

`prior` the prior that was used

`w` the mixing weight as estimated by marginal maximum likelihood

`a` (only present if Laplace prior used) the scale factor as supplied or estimated

`bayesfac` the value of the parameter `bayesfac`, determining whether Bayes factor or posterior median thresholds are used.

`sdev` the standard deviation of the data as supplied or estimated

`threshrule` the thresholding rule used, as specified above.

## DETAILS

The appropriate routines elsewhere in the package are called to perform the actual calculations.

## SEE ALSO

[wfromx](#), [wandafromx](#), [tfromw](#), [postmean](#), [postmed](#), [threshold](#)

`ebayesthresh.wavelet`

## **Performs Empirical Bayes thresholding on the levels of a wavelet transform**

### DESCRIPTION

Given a wavelet transform obtained using one of the routines (either `dwt` or `nd.dwt`) in S+WAVELETS, or the routine `wd` in WaveThresh, apply an Empirical Bayes thresholding approach to the detail coefficients in the transform, as discussed in Johnstone and Silverman (2002a,b). Either the standard or the nondecimated wavelet transform can be processed. After the thresholding, the wavelet transform is inverted to obtain the reconstruction of the estimated regression function underlying the original data.

### USAGE

```
ebayesthresh.wavelet(x.dwt, vscale = "independent", smooth.levels =  
Inf, prior = "laplace", a = 0.5, bayesfac = F, threshrule = "median",  
package="spluswavelets")
```

### REQUIRED ARGUMENT

`x.dwt` The wavelet transform of a vector of data. The transform is obtained using one of the wavelet transform routines in S+WAVELETS. If the standard discrete wavelet transform is required, use the routine `dwt`. If the nondecimated transform is wanted, use the routine `nd.dwt`. Any choice of wavelet, boundary condition, etc provided by these routines can be used. Note that the wavelet transform must be calculated before the routine `ebayesthresh.wavelet` is called.

### OPTIONAL ARGUMENTS

`vscale` Controls the scale used at different levels of the transform. If `vscale` is a scalar quantity, then it will be assumed that the wavelet coefficients at every level have this standard deviation. If `vscale = "independent"`, the standard deviation will be estimated from the highest level of the wavelet transform and will then be used for all levels processed. If `vscale="level"`, then the standard deviation will be estimated separately for each level processed, allowing standard deviation that is level-dependent.

`smooth.levels` the number of levels to be processed, if less than the number of levels of detail calculated by the wavelet transform.

`prior` the prior to be used conditional on the mean being nonzero. Possible values are `"laplace"` for the Laplace prior and `"cauchy"` for the quasi-Cauchy prior.

`a` the scale factor if the Laplace prior is used. If, on entry, `a=NA` and `prior="laplace"`, then the scale parameter `a` will be estimated by marginal maximum likelihood, separately for each level of the transform. If the quasi-Cauchy prior is used then this argument is ignored.

`bayesfac` if `bayesfac=T` then whenever a threshold is explicitly calculated, the Bayes factor threshold will be used, viz the value of the data such that the posterior probability that the underlying mean is zero is exactly 0.5. Otherwise the posterior median threshold will be used.

`threshrule` specifies the thresholding rule to be applied to the data. Possible values are "median" (use the posterior median); "mean" (use the posterior mean); "hard" (carry out hard thresholding); "soft" (carry out soft thresholding).

`package` specifies which package has been used to construct the wavelet transform, and will be used to perform the reconstruction. Possible values are "spluswavelets" (for the S+Wavelets module) or "wavethresh" (for the [WaveThresh](#) package).

#### VALUE

A vector giving the values of the estimated regression function underlying the original data

#### DETAILS

Apart from some housekeeping to estimate the standard deviation if necessary, and to determine the number of levels to be processed, the main part of the routine is a call, for each level, to the smoothing routine `ebayesthresh`. The final step is a call to the S+WAVELETS routine `reconstruct` to invert the transform. The basic notion of processing each level of detail coefficients is easily transferred to transforms constructed using other wavelet software. Similarly, it is straightforward to modify the routine to give other details of the wavelet transform, if necessary using the option `verbose = T` in the calls to `ebayesthresh`.

#### SEE ALSO

[ebayesthresh](#)

postmean

## Find posterior mean estimator given data

### DESCRIPTION

Given a data value or a vector of data, find the posterior mean estimate(s) of the underlying signal value(s)

### USAGE

```
postmean(x,w, prior="laplace", a=0.5)
```

### REQUIRED ARGUMENTS

`x` a data value or a vector of data

`w` the value of the prior probability that the signal is nonzero

### OPTIONAL ARGUMENTS

`prior` specification of the prior to be used for the nonzero part of the prior; can be "cauchy" or "laplace"

`a` If the Laplace prior is used, `a` is the scale factor

### VALUE

A value or vector of values of the estimate(s) of the mean(s) of the distribution(s) from which the `x` are drawn.

### DETAILS

The appropriate one of `postmean.laplace` or `postmean.cauchy` is called.

### SEE ALSO

[postmean.laplace](#), [postmean.cauchy](#), [postmed](#)

### EXAMPLE

```
muhat <- postmean(x,w=0.2)
```

`postmean.cauchy`

## **Find posterior mean estimator given data, using quasi-Cauchy prior for the nonzero component**

### DESCRIPTION

Given a data value or a vector of data, find the posterior mean estimate(s) of the underlying signal value(s). The quasi-Cauchy distribution is used for the nonzero component of the prior

### USAGE

```
postmean.cauchy(x, w)
```

### REQUIRED ARGUMENTS

`x` a data value or a vector of data

`w` the value of the prior probability that the signal is nonzero

### VALUE

A value or vector of values of the estimate(s) of the mean(s) of the distribution(s) from which the `x` are drawn.

### DETAILS

The posterior mean is found by expressing the quasi-Cauchy prior as a mixture. The mean conditional on the mixing parameter is found and is then averaged over the posterior distribution of the mixing parameter, including the atom of probability at zero variance.

### SEE ALSO

[postmean](#), [postmean.laplace](#)

### EXAMPLE

```
muhat <- postmean.cauchy(x)
```

`postmean.laplace`

## **Find posterior mean estimator given data, using Laplace prior for the nonzero component**

### DESCRIPTION

Given a data value or a vector of data, find the posterior mean estimate(s) of the underlying signal value(s). The Laplace distribution is used for the nonzero component of the prior

### USAGE

```
postmean.laplace(x, w, a=0.5)
```

### REQUIRED ARGUMENTS

`x` a data value or a vector of data

`w` the value of the prior probability that the signal is nonzero

### OPTIONAL ARGUMENT

`a` the scale factor

### VALUE

A value or vector of values of the estimate(s) of the mean(s) of the distribution(s) from which the `x` are drawn.

### DETAILS

The posterior mean is found explicitly. The posterior probability that the parameter is nonzero is found, as is the posterior mean conditional on not being zero; the product of these is the posterior mean.

### SEE ALSO

[postmean](#), [postmean.cauchy](#)

### EXAMPLE

```
muhat <- postmean.laplace(x)
```

postmed

## Find posterior median estimator given data

### DESCRIPTION

Given a data value or a vector of data, find the posterior median estimate(s) of the underlying signal value(s)

### USAGE

```
postmed(x,w, prior="laplace", a=0.5)
```

### REQUIRED ARGUMENTS

`x` a data value or a vector of data

`w` the value of the prior probability that the signal is nonzero

### OPTIONAL ARGUMENTS

`prior` specification of the prior to be used for the nonzero part of the prior; can be "cauchy" or "laplace"

`a` If the Laplace prior is used, `a` is the scale factor

### VALUE

A value or vector of values of the estimate(s) of the mean(s) of the distribution(s) from which the `x` are drawn.

### DETAILS

The appropriate one of `postmed.laplace` or `postmed.cauchy` is called.

### SEE ALSO

[postmed.laplace](#), [postmed.cauchy](#), [postmean](#)

### EXAMPLE

```
muhat <- postmed(x)
```

`postmed.cauchy`

## **Find posterior median estimator given data, using quasi-Cauchy prior for the nonzero component**

### DESCRIPTION

Given a data value or a vector of data, find the posterior median estimate(s) of the underlying signal value(s). The quasi-Cauchy distribution is used for the nonzero component of the prior

### USAGE

```
postmed.cauchy(x, w)
```

### REQUIRED ARGUMENTS

`x` a data value or a vector of data

`w` the value of the prior probability that the signal is nonzero

### VALUE

A value or vector of values of the estimate(s) of the mean(s) of the distribution(s) from which the `x` are drawn.

### DETAILS

The posterior median is found by finding the zero of the function `cauchy.medzero`.

### SEE ALSO

[postmed](#), [postmed.laplace](#), [cauchy.medzero](#)

### EXAMPLE

```
muhat <- postmed.cauchy(x)
```

`postmed.laplace`

## **Find posterior median estimator given data, using Laplace prior for the nonzero component**

### DESCRIPTION

Given a data value or a vector of data, find the posterior median estimate(s) of the underlying signal value(s). The Laplace distribution is used for the nonzero component of the prior

### USAGE

```
postmed.laplace(x, w, a=0.5)
```

### REQUIRED ARGUMENTS

`x` a data value or a vector of data

`w` the value of the prior probability that the signal is nonzero

### OPTIONAL ARGUMENTS

`a` the scale factor

### VALUE

A value or vector of values of the estimate(s) of the mean(s) of the distribution(s) from which the `x` are drawn.

### DETAILS

The posterior median is found explicitly, without any need for the numerical solution of an equation.

### SEE ALSO

[postmed](#), [postmed.cauchy](#)

### EXAMPLE

```
muhat <- postmed.laplace(x)
```

tfromw

## Find Empirical Bayes threshold from mixing weight

### DESCRIPTION

Given a weight, find the corresponding threshold.

### USAGE

```
tfromw(w, prior="laplace", bayesfac=F, a=0.5)
```

### REQUIRED ARGUMENTS

w a mixing weight

### OPTIONAL ARGUMENTS

prior specification of the prior to be used; can be "cauchy" or "laplace"

bayesfac logical variable. If T the threshold is found using the Bayes factor method, so that the posterior probability of zero is exactly half if the data are equal to the threshold. If F (the default), the threshold is that of the posterior median function given the data value

a If the Laplace prior is used, a is the scale factor. If the Cauchy prior is used, a is ignored.

### VALUE

The numerical value of the estimated threshold is returned.

### DETAILS

The search is over an appropriate function using a binary search. The function to be zeroed is `beta.laplace` or `beta.cauchy` if the Bayes factor threshold is to be found, and `laplace.threshzero` or `cauchy.threshzero` for the posterior median threshold.

### SEE ALSO

[wfromx](#), [tfromx](#), [beta.laplace](#), [beta.cauchy](#)

### EXAMPLE

```
threshold <- tfromw(w)
```

tfromx

## Find threshold from data

### DESCRIPTION

Given a vector of data, find the threshold corresponding to the marginal maximum likelihood choice of weight.

### USAGE

```
tfromx(x, prior="laplace", bayesfac=F, a=0.5)
```

### REQUIRED ARGUMENTS

x a vector of data

### OPTIONAL ARGUMENTS

prior specification of the prior to be used; can be "cauchy" or "laplace"

bayesfac logical variable. If T the threshold is found using the Bayes factor method, so that the posterior probability of zero is exactly half if the data are equal to the threshold. If F (the default), the threshold is that of the posterior median function given the data value

a If the Laplace prior is used, a is the scale factor

### VALUE

The numerical value of the threshold is returned.

### DETAILS

First, the routine `wfromx` is called to find the estimated weight. Then the routine `tfromw` is used to find the threshold.

### SEE ALSO

[tfromw](#), [wfromx](#).

### EXAMPLE

```
threshold <- tfromx(x)
```

threshold

## **Apply hard or soft thresholding to data**

### DESCRIPTION

Given a data value or a vector of data, threshold the data at a specified value

### USAGE

```
threshold(x, t, hard=T)
```

### REQUIRED ARGUMENTS

`x` a data value or a vector of data

`t` the value of the threshold to be used

### OPTIONAL ARGUMENTS

`hard` if `T` hard thresholding is applied, otherwise soft thresholding is used.

### VALUE

A value or vector of values the same length as `x`, containing the result of the relevant thresholding rule applied to `x`.

vecbinsolv

## **Solve a nonlinear equation or a vector of nonlinear equations based on an increasing function in a specified interval**

### DESCRIPTION

Solve nonlinear equation(s) by binary search. The equation(s) must be based on increasing function(s).

### USAGE

```
vecbinsolv((zf, fun, tlo, thi, nits = 30, ...))
```

### REQUIRED ARGUMENTS

`zf` a value or vector of values giving the right hand sides of the equations to be solved

`fun` . . . . a function of a scalar or vector argument, that returns the vector of values of the function at the values of its individual components. Additional arguments to `fun` may be passed through . . .

`tlo` the lower limit of the interval over which the solution is sought

`thi` the upper limit of the interval over which the solution is sought

### OPTIONAL ARGUMENTS

`nits` the number of binary subdivisions carried out (so that if `nits = 30` the solutions are each accurate to about 9 orders of magnitude less than `thi - tlo`).

### VALUE

A vector `t` the same length as `x` such that, component-wise if necessary, `fun(t) = x` where this is possible within the interval `(tlo, thi)`. The relevant value returned is the nearer extreme to the solution if there is no solution in the specified range.

### DETAILS

The search is carried out using a binary search, or a vector of binary searches. There is some inefficiency in early iterations because the function is evaluated at the midpoint of every interval under consideration, not taking cognisance of the fact that many of these midpoints will be identical at least initially. For a scalar argument the function `uniroot` may be more efficient, but `vecbinsolv` is more stable and avoids loops when solving vectors of equations.

wandafromx

## Find Empirical Bayes weight and scale factor from data if Laplace prior is used

### DESCRIPTION

Given a vector of data, find the marginal maximum likelihood choice of weight and scale factor under the Laplace prior.

### USAGE

```
wfromx(x)
```

### REQUIRED ARGUMENTS

`x` a vector of data

### VALUE

Returns a list with the following values

`w` The estimated weight

`a` The estimated scale factor

### DETAILS

The parameters are found by marginal maximum likelihood. The search is over weights corresponding to thresholds in the range  $[0, \sqrt{2 \log n}]$ , where  $n = \text{length}(x)$ . The search uses a nonlinear optimization routine (`nlminb` in S-PLUS, `optim` in R) to minimize the auxiliary function `negloglik.laplace`. The range over which the scale factor is searched is  $(0.04, 3)$ .

### SEE ALSO

[wfromx](#)

### EXAMPLE

```
weight <- wandafromx(x)
```

wfromt

## Find mixing weight from posterior median threshold

### DESCRIPTION

Given a threshold value, find the mixing weight for which this is the threshold of the posterior median estimator.

### USAGE

```
wfromt(w, prior="laplace", a=0.5)
```

### REQUIRED ARGUMENTS

`t` a threshold value

### OPTIONAL ARGUMENTS

`prior` specification of the prior to be used; can be "cauchy" or "laplace"

`a` If the Laplace prior is used, `a` is the scale factor. If the Cauchy prior is used, `a` is ignored.

### VALUE

The numerical value of the corresponding weight is returned.

### DETAILS

Explicit formulas are used; there is no need for a binary search in this case.

### SEE ALSO

[wfromx](#), [tfromx](#), [tfromw](#)

### EXAMPLE

```
w <- wfromt(threshold)
```

wfromx

## Find Empirical Bayes weight from data

### DESCRIPTION

Given a vector of data, find the marginal maximum likelihood choice of weight.

### USAGE

```
wfromx(x, prior="laplace", a=0.5)
```

### REQUIRED ARGUMENTS

`x` a vector of data

### OPTIONAL ARGUMENTS

`prior` specification of the prior to be used; can be "cauchy" or "laplace"

`a` If the Laplace prior is used, `a` is the scale factor. If the Cauchy prior is used, `a` is ignored.

### VALUE

The numerical value of the estimated weight is returned.

### DETAILS

The weight is found by marginal maximum likelihood. The search is over weights corresponding to thresholds in the range  $[0, \sqrt{2 \cdot \log(n)}]$ , where  $n = \text{length}(x)$ . The search is by binary search for a solution to the equation  $S(w)=0$ , where  $S$  is the score function. The binary search is on a logarithmic scale in  $w$ .

### SEE ALSO

[tfromw](#), [tfromx](#)

### EXAMPLE

```
weight <- wfromx(x)
```

wmonfromx

## Find monotone decreasing Empirical Bayes weights from data

### DESCRIPTION

Given a vector of data, find the marginal maximum likelihood choice of weight sequence subject to the constraints that the weights are monotone decreasing.

### USAGE

```
wmonfromx(x, prior="laplace", a=0.5)
```

### REQUIRED ARGUMENTS

`x` a vector of data

### OPTIONAL ARGUMENTS

`prior` specification of the prior to be used; can be "cauchy" or "laplace"

`a` If the Laplace prior is used, `a` is the scale factor. If the Cauchy prior is used, `a` is ignored.

### VALUE

The vector of estimated weights is returned.

### DETAILS

The weights is found by marginal maximum likelihood. The search is over weights corresponding to thresholds in the range  $[0, \sqrt{2 \cdot \log(n)}]$ , where  $n = \text{length}(x)$ . An iterated least squares monotone regression algorithm is used to maximize the log likelihood. The weighted least squares monotone regression routine `isotone` is used.

### SEE ALSO

[isotone](#)

## Appendix 2: R/S-PLUS program listing

```

"ebayesthresh"<-
function(x, prior = "laplace", a = 0.5, bayesfac = F, sdev = 1, verbose = F,
        threshrule = "median")
{
# Given a vector of data x, find the marginal maximum likelihood estimator
# of the mixing weight w, and apply an appropriate thresholding rule using
# this weight.
# If the prior is laplace and a=NA, then the scale factor is also found by MML.
# The thresholding rules allowed are "median", "mean", "hard", "soft" and "none";
# if "none" is used, then only the parameters are worked out.
# If hard or soft thresholding is used, the argument "bayesfac" specifies
# whether to use the bayes factor threshold or the posterior median threshold.
# If verbose=T then the routine returns a list with several arguments, including
# muhat which is the result of the thresholding.
# If verbose=F then only muhat is returned.
# It is assumed that the standard deviation of the data is sdev; if sdev=NA, then
# it is estimated using the function mad(x).
#
# find the standard deviation if necessary and estimate the parameters
  if(is.na(sdev)) sdev <- mad(x, center = 0)
  x <- x/sdev
  pr <- substring(prior, 1, 1)
  if(pr == "l" & is.na(a)) {
    pp <- wandafromx(x)
    w <- pp$w
    a <- pp$a
  }
  else w <- wfromx(x, prior = prior, a = a)      #
  if(pr != "m" | verbose) {
    tt <- tfromw(w, prior = prior, bayesfac = bayesfac, a = a)
    tcor <- sdev * tt
  }
  if(threshrule == "median")
    muhat <- postmed(x, w, prior = prior, a = a)
  if(threshrule == "mean")
    muhat <- postmean(x, w, prior = prior, a = a)
  if(threshrule == "hard")
    muhat <- threshold(x, tt)
  if(threshrule == "soft")
    muhat <- threshold(x, tt, hard = F)
  if(threshrule == "none") muhat <- NA #
# Now return desired output
#
  muhat <- sdev * muhat
  if(!verbose)
    return(muhat)
  retlist <- list(muhat = muhat, x = x, threshold.sdevscale = tt,
                 threshold.origscale = tcor, prior = prior, w = w, a = a,
                 bayesfac = bayesfac, sdev = sdev, threshrule = threshrule)
  if(pr == "c")
    retlist <- retlist[-7]
  if(threshrule == "none")
    retlist <- retlist[-1]
  return(retlist)
}
"ebayesthresh.wavelet" <-
function(x.dwt, vscale = "independent", smooth.levels = Inf, prior = "laplace",
        a = 0.5, bayesfac = F, threshrule = "median", package = "spluswavelets"
)

```

```

{
#
# Carry out the empirical Bayes smoothing approach for the detail
# coefficients of the wavelet transform xdwt, produced using the
# S+Wavelets module or WaveThresh package
#
# If smooth.levels is set to a smaller number than n.levels then
# only smooth.levels levels are processed.
#
# The parameter vscale controls the estimation of the variances,
# as follows:
#   if vscale is a scalar, it is used as the standard deviation
#     of the wavelet coefficients at every level.
#   if vscale = "independent", the standard deviation is estimated
#     using the coefficients at the highest level, and the same
#     value is used at every level
#   if vscale = "level" then level-dependent standard deviations
#     are estimated
#
# The other arguments (prior, a, bayesfac, threshrule) have the
# same meaning as for ebayesthresh. If prior="laplace" and a=0,
# then a is estimated separately at each level
#
# To translate this routine to work with other wavelet software,
# bear in mind the following features of the S+WAVELETS wavelet
# transform:
#
# nlevs is the number of levels of detail computed within the
# transform. The detail is then held in vectors which are
# accessible as x.dwt[[2]], x.dwt[[3]], ... , x.dwt[[nlevs+1]],
# with x.dwt[[2]] being the coarsest level and x.dwt[[nlevs+1]]
# the finest.
#
# The routine reconstruct inverts the transform, automatically
# doing the right thing if the nondecimated transform is being used
#
# In the present version, set package="wavethresh" to use the wavethresh
# option
#
# Grateful acknowledgements to Stuart Barber for the wavethresh implementation
#
pkg <- casefold(substring(package, 1, 1))
if(pkg == "s") {
  nlevs <- attributes(x.dwt)$n.levels
  slevs <- min(nlevs, smooth.levels)
  if(is.character(vscale)) {
    vs <- substring(vscale, 1, 1)
    if(vs == "i")
      vscale <- mad(x.dwt[[nlevs + 1]])
    if(vs == "l")
      vscale <- NA
  }
  for(j in ((nlevs - slevs + 2):(nlevs + 1)))
    x.dwt[[j]] <- ebayesthresh(as.vector(x.dwt[[j]]), prior,
                              a, bayesfac, vscale, F, threshrule)
  x <- reconstruct(x.dwt)
}
else {
  print("Using WaveThresh")
  nlevs <- x.dwt$nlevels
  slevs <- min(nlevs - 1, smooth.levels)
  if(is.character(vscale)) {

```

```

        vs <- substring(vscale, 1, 1)
        if(vs == "i")
            vscale <- mad(accessD(x.dwt, level = nlevs -
                            1))
        if(vs == "l")
            vscale <- NA
    }
    for(j in (nlevs - slevs):(nlevs - 1)) {
        x.dwt <- putD(x.dwt, level = j, v = ebayesthresh(
            accessD(x.dwt, level = j), prior, a, bayesfac,
            vscale, F, threshrule))
        print(paste("Done level", j))
    }
    if(x.dwt$type == "station")
        x <- AvBasis(convert(x.dwt))
    else x <- wr(x.dwt)
}
return(x)
}
"tfromw"<-
function(w, prior = "laplace", bayesfac = F, a = 0.5)
{
# given the vector of weights w, find the threshold or vector of
# thresholds corresponding to these weights, under the specified prior.
#
# if bayesfac=T the Bayes factor thresholds are found, otherwise the posterior median
# thresholds are found.
#
# if the Laplace prior is used, a gives the value of the scale factor
#
    pr <- substring(prior, 1, 1)
    if(bayesfac) {
        z <- 1/w - 2
        if(pr == "l")
            tt <- vecbinsolv(z, beta.laplace, 0, 10, a = a)
        if(pr == "c")
            tt <- vecbinsolv(z, beta.cauchy, 0, 10)
    }
    else {
        zz <- rep(0, length(w))
        if(pr == "l")
            tt <- vecbinsolv(zz, laplace.threshzero, 0, 10, w = w,
                            a = a)
        if(pr == "c")
            tt <- vecbinsolv(zz, cauchy.threshzero, 0, 10, w = w)
    }
    return(tt)
}
"tfromx"<-
function(x, prior = "laplace", bayesfac = F, a = 0.5)
{
# given the data x, the prior, and any other parameters,
# find the threshold
# corresponding to the marginal maximum likelihood estimator
# of the mixing weight.
#
    if ( prior=="laplace" && is.na(a) ) { wa _ wandafromx( x)
w _ wa$w
a _ wa$a
} else w <- wfromx(x, prior, a = a)
t <- tfromw(w, prior, a = a, bayesfac = bayesfac)
return(t)
}

```

```

}
"wfromt"<-
function(tt, prior = "laplace", a = 0.5)
{
# find the weight that has posterior median threshold tt,
#
  pr <- substring(prior, 1, 1)
  if(pr == "l")
    wi <- (a * pnorm(tt - a))/dnorm(tt - a) - beta.laplace(tt, a)
  if(pr == "c") {
    dnz <- dnorm(tt)
    wi <- 1 + (pnorm(tt) - tt * dnz - 1/2)/(sqrt(pi/2) * dnz * tt^2)
    wi[!is.finite(wi)] <- 1
  }
  return(1/wi)
}
}
"wfromx"<-
function(x, prior = "laplace", a = 0.5)
{
# given the vector of data x and the function betaf
# which finds beta(x),
# find the value of w that zeroes S(w) in the
# range
#
# additional arguments to betaf can be passed through ...
#
# works by successive bisection, carrying out nits harmonic bisections
# of the original interval between wlo and 1.
#
#
  pr <- substring(prior, 1, 1)
  tuniv <- sqrt(2 * log(length(x)))
  wlo <- wfromt(tuniv, prior, a)
  if(pr == "l") {
    beta <- beta.laplace(x, a)
  }
  if(pr == "c") {
    beta <- beta.cauchy(x)
  }
  whi <- 1
  beta <- pmin(beta, 1e20)
  shi <- sum(beta/(1 + beta))
  if(shi >= 0)
    return(w = 1)
  slo <- sum(beta/(1 + wlo * beta))
  if(slo <= 0)
    return(w = wlo)
  for(j in (1:30)) {
    wmid <- sqrt(wlo * whi)
    smid <- sum(beta/(1 + wmid * beta))
    if(smid == 0)
      return(w = wmid)
    if(smid > 0) {
      wlo <- wmid
    }
    else {
      whi <- wmid
    }
  }
  return(w = sqrt(wlo * whi))
}
}

```

```

"wandafromx"<-
function(x)
{
# finds the marginal max lik estimators of w and a, using a bivariate optimization
#
# The threshold is constrained to lie between 0 and sqrt ( 2 log (n))
#
# If running R, the routine optim is used; in S-PLUS the routine is nlminb
#
  thi <- sqrt(2 * log(length(x)))
  lo_c(0,0.04)
  hi_c(thi,3)
  startpar_c(1,0.5)
  if (exists("optim")) {
    uu <- optim(startpar, negloglik.laplace, method="L-BFGS-B",
               lower = lo, upper = hi, xx = x)
  }
  uu <- uu$par
  }
  else {uu <- nlminb(startpar, negloglik.laplace, lower = lo, upper = hi, xx = x)
  uu <- uu$parameters}
  a <- uu[2]
  w <- wfromt(uu[1], a = a)
  return(w, a)
}
"postmean"<-
function(x, w, prior = "laplace", a = 0.5)
{
#
# find the posterior mean for the appropriate prior for
# given x and w and a, assuming the error variance
# is 1.
#
  pr <- substring(prior, 1, 1)
  if(pr == "l")
    mutilde <- postmean.laplace(x, w, a = a)
  if(pr == "c")
    mutilde <- postmean.cauchy(x, w)
  return(mutilde)
}
"postmean.cauchy"<-
function(x, w)
{
#
# Find the posterior mean for the quasi-Cauchy prior with mixing weight w
# given data x, which may be a scalar or a vector.
#
  muhat <- x
  ind <- (x == 0)
  x <- x[!ind]
  ex <- exp( - x^2/2)
  z <- w * (x - (2 * (1 - ex))/x)
  z <- z/(w * (1 - ex) + (1 - w) * ex * x^2)
  muhat[!ind] <- z
  return(muhat)
}
"postmean.laplace"<-
function(x, w, a = 0.5)
{
#
# find the posterior mean for the double exponential prior for
# given x, w and a, assuming the error variance
# is 1.

```

```

#
# only allow a < 20.
  a <- min(a, 20) #
# First find the odds of zero and the shrinkage factor
#
  wpost <- 1 - (1 - w)/(1 + w * beta.laplace(x, a))
  # now find the posterior mean conditional on being non-zero
#
  sx <- sign(x)
  x <- abs(x)
  cp1 <- pnorm(x - a)
  dp1 <- dnorm(x - a)
  cp2 <- pnorm(-x - a)
  dp2 <- dnorm(x + a)
  ef <- exp(pmin(2 * a * x, 100))
  postmeancond <- ((x - a) * cp1 + dp1 + ef * ((x + a) * cp2 - dp2))/(cp1 +
    ef * cp2) #
# calculate posterior mean and return
#
  mutilde <- sx * wpost * postmeancond
  return(mutilde)
}
"postmed"<-
function(x, w, prior = "laplace", a = 0.5)
{
#
# find the posterior median for the appropriate prior for
# given x and w and a, assuming the error variance
# is 1.
#
  pr <- substring(prior, 1, 1)
  if(pr == "l")
    muhat <- postmed.laplace(x, w, a = a)
  if(pr == "c")
    muhat <- postmed.cauchy(x, w)
  return(muhat)
}
"postmed.cauchy"<-
function(x, w)
{
#
# find the posterior median of the Cauchy prior with
# mixing weight w, pointwise for each of the data points x
#
  nx <- length(x)
  zest <- rep(NA, length(x))
  w <- rep(w, length.out = nx)
  ax <- abs(x)
  j <- (ax < 20)
  zest[!j] <- x[!j] - 2/x[!j]
  if(sum(j) > 0) {
    zest[j] <- vecbinsolv(zf = rep(0, sum(j)), fun = cauchy.medzero,
      tlo = 0, thi = max(ax[j]), z = ax[j], w = w[j])
  }
  zest[zest < 1e-007] <- 0
  zest <- sign(x) * zest
  return(zest)
}
"postmed.laplace"<-
function(x, w, a = 0.5)
{
#

```

```

# find the posterior median for the Laplace prior for
# given x (possibly vector), w and a, assuming the error variance
# is 1.
#
# only allow a < 20.
  a <- min(a, 20) #
# Work with the absolute value of x, and for x > 25 use the approximation
# to dnorm(x-a)*beta.laplace(x, a)
#
  sx <- sign(x)
  x <- abs(x)
  xma <- x - a
  zz <- (dnorm(x - a) * (1/w + beta.laplace(x, a)))/a
  zz[x > 25] <- 0.5
  mucor <- qnorm(pmin(zz, 1))
  muhat <- sx * pmax(0, x - a - mucor)
  return(muhat)
}
"threshold"<-
function(x, t, hard = T)
{
#
# threshold the data x using threshold t
# if hard=T use hard thresholding
# if hard=F use soft thresholding
  if(hard) z <- x * (abs(x) >= t) else {
    z <- sign(x) * pmax(0, abs(x) - t)
  }
  return(z)
}
"vecbinsolv"<-
function(zf, fun, tlo, thi, nits = 30, ...)
{
# given a monotone function fun, and a vector of values
# zf find a vector of numbers t such that f(t) = zf.
# The solution is constrained to lie on the interval (tlo, thi)
#
# The function fun may be a vector of increasing functions
#
# Present version is inefficient because separate calculations
# are done for each element of z, and because bisections are done even
# if the solution is outside the range supplied
#
# It is important that fun should work for vector arguments.
# Additional arguments to fun can be passed through ...
#
# Works by successive bisection, carrying out nits harmonic bisections
# of the interval between tlo and thi
#
  nz <- length(zf)
  tlo <- rep(tlo, nz)
  thi <- rep(thi, nz) #
# carry out nits bisections
#
  for(jj in (1:nits)) {
    tmid <- (tlo + thi)/2
    fmid <- fun(tmid, ...)
    indt <- (fmid <= zf)
    tlo[indt] <- tmid[indt]
    thi[!indt] <- tmid[!indt]
  }
  tsol <- (tlo + thi)/2
}

```

```

        return(tsol)
    }
}
"negloglik.laplace"<-
function(xpar, xx)
{
#
# marginal negative log likelihood function for laplace prior
#
# xx data
# xpar vector of two parameters:
#   xpar[1] : threshold
#   xpar[2] : scale factor a
#
    a <- xpar[2]
    w <- wfromt(xpar[1], a = a)
    loglik <- sum(log(1 + w * beta.laplace(xx, a)))
    return( - loglik)
}
"beta.cauchy"<-
function(x)
{
#
# Find the function beta
# for the mixed normal prior with Cauchy tails. It is assumed that the
# noise variance is equal to one.
#
    phix <- dnorm(x)
    j <- (x != 0)
    beta <- x
    beta[!j] <- -1/2
    beta[j] <- (dnorm(0)/phix[j] - 1)/x[j]^2 - 1
    return(beta)
}
"beta.laplace"<-
function(x, a = 0.5)
{
#
# The function beta for the Laplace prior with parameter a
#
    x <- abs(x)
    a <- min(a, 35)
    xpa <- x + a
    xma <- x - a
    rat1 <- 1/xpa
    rat1[xpa < 35] <- pnorm(- xpa[xpa < 35])/dnorm(xpa[xpa < 35])
    rat2 <- pnorm(xma)/dnorm(xma)
    beta <- (a/2) * (rat1 + rat2) - 1
    return(beta)
}
"laplace.threshzero"<-
function(x, w, a = 0.5)
{
#
# the function that has to be zeroed to find the threshold with the Laplace
# prior.
# only allow a < 20.
    a <- min(a, 20) #
    z <- pnorm(x - a) - (dnorm(x - a) * (1/w + beta.laplace(x, a)))/a
    return(z)
}
"cauchy.medzero"<-
function(x, z, w)

```

```

{
# the objective function that has to be zeroed, component by component,
# to find the
# posterior median when the quasi-Cauchy prior is used.
# x is the parameter vector, z is the data vector, w
# is the weight
# x and z may be scalars
#
      hh <- z - x
      dnhh <- dnorm(hh)
      yleft <- pnorm(hh) - z * dnhh + ((z * x - 1) * dnhh * pnorm(-x))/
        dnorm(x)
      yright2 <- 1 + exp(-z^2/2) * (z^2 * (1/w - 1) - 1)
      return(yright2/2 - yleft)
}
"cauchy.threshzero"<-
function(z, w)
{
# the objective function that has to be zeroed
# to find the Cauchy
# threshold. z is the putative threshold vector, w
# is the weight
# w can be a vector
#
      y <- pnorm(z) - z * dnorm(z) - 1/2 - (z^2 * exp(-z^2/2) * (1/w - 1))/
        2
      return(y)
}
"wmonfromx"<-
function(xd, prior = "laplace", a = 0.5, tol = 1e-008, maxits = 20)
{
#
# Find the monotone marginal maximum likelihood estimate of the mixing weights
# for the Laplace prior with parameter a. It is assumed that the
# noise variance is equal to one.
#
# Find the beta values and the minimum weight
#
      pr <- substring(prior, 1, 1)
      nx <- length(xd)
      wmin <- wfromt(sqrt(2 * log(length(xd))), prior, a)
      winit <- 1
      if(pr == "l")
        beta <- beta.laplace(xd, a)
      if(pr == "c") beta <- beta.cauchy(xd) #
# now conduct iterated weighted least squares isotone regression
#
      w <- rep(winit, length(beta))
      for(j in 1:maxits) {
        aa <- w + 1/beta
        ps <- w + aa
        ww <- 1/aa^2
        wnew <- isotone(ps, ww, increasing = F)
        wnew <- pmax(wmin, wnew)
        wnew <- pmin(1, wnew)
        zinc <- max(abs(range(wnew - w)))
        w <- wnew
        if(zinc < tol)
          return(w)
      }
      cat("Warning: more iterations required to achieve convergence \n")
      return(w)
}

```

```

}
"isotone"<-
function(x, wt = rep(1, length(x)), increasing = F)
{
#
# find the weighted least squares isotone fit to the
# sequence x, the weights given by the sequence wt
#
# if increasing == T the curve is set to be increasing,
# otherwise to be decreasing
#
# the vector ip contains the indices on the original scale of the
# breaks in the regression at each stage
#
  nn <- length(x)
  if(nn == 1)
    return(x)
  if(!increasing)
    x <- - x
  ip <- (1:nn)
  dx <- diff(x)
  nx <- length(x)
  while((nx > 1) && (min(dx) < 0)) {
#
# do single pool-adjacent-violators step
#
# find all local minima and maxima
#
    jmax <- (1:nx)[c(dx <= 0, F) & c(T, dx > 0)]
    jmin <- (1:nx)[c(dx > 0, T) & c(F, dx <= 0)] #
# do pav step for each pair of maxima and minima
#
# add up weights within subsequence that is pooled
# set first element of subsequence to the weighted average
# the first weight to the sum of the weights within the subsequence
# and remainder of the subsequence to NA
#
    for(jb in (1:length(jmax))) {
      ind <- (jmax[jb]:jmin[jb])
      wtn <- sum(wt[ind])
      x[jmax[jb]] <- sum(wt[ind] * x[ind])/wtn
      wt[jmax[jb]] <- wtn
      x[(jmax[jb] + 1):jmin[jb]] <- NA
    }
#
# clean up within iteration, eliminating the parts of sequences that were set
# to NA
#
    ind <- !is.na(x)
    x <- x[ind]
    wt <- wt[ind]
    ip <- ip[ind]
    dx <- diff(x)
    nx <- length(x)
  }
#
# final cleanup: reconstruct z at all points by repeating the pooled values
# the appropriate number of times
#
  jj <- rep(0, nn)
  jj[ip] <- 1
  z <- x[cumsum(jj)]
}

```

```
    if(!increasing)
      z <- -z
    return(z)
  }
```