ebnm: An R Package for Solving the Empirical Bayes Normal Means Problem Using a Variety of Prior Families

Jason Willwerscheid
University of Chicago

Matthew Stephens
University of Chicago

Abstract

The empirical Bayes normal means (EBNM) model plays an important role in both theoretical and applied statistics. Applications include meta-analysis and shrinkage estimation; wavelet denoising; multiple testing and false discovery rate estimation; and empirical Bayes matrix factorization. As such, several software packages have been developed that fit this model under different prior assumptions. Each package naturally has a different interface and outputs, which complicates comparison of results for different prior families. Further, there are some notable gaps in the software — for example, implementations for simple normal and point-normal priors are absent. Motivated by these issues, we developed the R package **ebnm**, which provides a unified interface for efficiently solving the EBNM problem using a wide variety of prior families, both parametric and non-parametric. Where practical we leverage core fitting procedures from existing packages, writing wrappers to create a unified interface; in other cases, we implement new core fitting procedures ourselves, with a careful focus on both speed and robustness. The result is a convenient and comprehensive package for solving the EBNM problem under a wide range of prior assumptions.

1. Introduction

Given n observations x_i with known standard deviations s_i $(i = 1, \ldots, n)$, the normal means model (Robbins 1951; Johnstone 2019) assumes that

$$x_i \overset{\text{ind.}}{\sim} \mathcal{N}(\theta_i, s_i^2), \tag{1}$$

with "true means" θ_i to be estimated. The maximum likelihood estimate of θ_i is, of course, simply x_i. The empirical Bayes (EB) approach to inferring the means θ_i attempts to improve upon these maximum likelihood estimates by "borrowing information" across observations, exploiting the fact that each observation contains information not only about its respective mean, but also about how the means are collectively distributed (Robbins 1956; Morris 1983; Efron 2010).

Specifically, the EB approach assumes that

$$\theta_i \overset{\text{ind.}}{\sim} g \in \mathcal{G}, \tag{2}$$

where the distribution g is to be estimated from among some family of distributions $\mathcal{G}$ that is specified in advance. It then fits the joint model (1)-(2) by first using *all* of the observations to estimate $g \in \mathcal{G}$ (e.g., via maximum likelihood), and then using the estimate $\hat{g}$ to compute posteriors for each mean θ_i. That is, the task is to:

1. Find $\hat{g} := \text{argmax}_{g \in \mathcal{G}} L(g)$, where

$$L(g) := p(x_1, \ldots, x_n | g, s_1, \ldots, s_n) = \prod_{i=1}^{n} \int p(x_i | \theta_i, s_i) g(\theta_i) \, d\theta_i; \tag{3}$$

2. Compute the posterior distributions

$$p(\theta_i | \hat{g}, x_i, s_i) \propto \hat{g}(\theta_i) p(x_i | \theta_i, s_i) \tag{4}$$

and/or desired posterior summaries (posterior means, variances, etc.).

We refer to this process as solving the "empirical Bayes normal means" (EBNM) problem. The difficulty of both steps naturally depends on the choice of family $\mathcal{G}$.

As we review in the following section, a number of software packages have been developed that can solve the EBNM problem, and the total available range of prior families is large. The availability of so many methods reflects the important role that the EBNM problem plays in both theoretical and applied statistics. Among the many possible applications of the EBNM problem, we cite wavelet denoising (Clyde and George 2000; Johnstone and Silverman 2004, 2005b), multiple testing (Efron 2010; Stephens 2017), and matrix factorization (Wang and Stephens 2021).

However, notable gaps in the software remain. For example, as far as we are aware no existing package solves the EBNM problem for arguably the simplest case, where $\mathcal{G}$ is the set of all normal distributions; nor does any existing package implement the point-normal family (i.e., the family of all distributions that are a mixture of a point mass at zero and a zero-centered normal distribution), which arises naturally when the means vector $\boldsymbol{\theta}$ is expected to be sparse. Further, each package has a different interface and outputs, which hinders comparison of results for different choices of $\mathcal{G}$, as well as making it difficult to develop software packages that can interface nicely with the full range of existing EBNM solvers.

Motivated by these issues, we developed the R package **ebnm**, which provides a unified interface for efficiently solving the EBNM problem using a wide variety of prior families. In some cases, we leveraged core fitting procedures from existing packages, writing wrappers to create a unified interface; in others, we implemented new core fitting procedures ourselves, with a careful focus on both speed and robustness. The result is a convenient and comprehensive package for solving the EBNM problem under a wide range of assumptions.

In Section 2, we give a brief history of the EBNM problem and review several existing software packages that can solve it. Section 3 describes our new **ebnm** package — both the unified interface and the newly implemented families. In Section 4 we compare different choices of prior family $\mathcal{G}$ via a brief numerical illustration of their performance in different settings, as well as providing timing comparisons. Finally, Section 5 illustrates two practical applications of **ebnm**, one a simple direct application to Rubin's eight schools data (Rubin 1981), and another where solving the EBNM problem arises as a key subroutine in matrix factorization.

2. Background and existing software

Stein (1956) famously discovered that under quadratic loss, the maximum likelihood estimate $\hat{\theta}_i = x_i$ is an inadmissible solution to the homoskedastic normal means problem

$$x_i \overset{\text{ind.}}{\sim} \mathcal{N}(\theta_i, s^2) \qquad i = 1, \ldots, n \tag{5}$$

when $n \geq 3$. James and Stein (1961) subsequently gave an explicit formula for a shrinkage estimator that dominates the MLE. As Efron and Morris (1973) showed, a lightly modified version of the James-Stein estimator can be derived via an EB approach that assumes

$$\theta_i \overset{\text{ind.}}{\sim} g \in \mathcal{G}, \tag{6}$$

where the prior family $\mathcal{G}$ is taken to be the family of zero-mean normal distributions

$$\mathcal{G}_{\text{norm0}} := \left\{ g : g \sim \mathcal{N}\left(0, \sigma^2\right) \text{ for some } \sigma^2 \geq 0 \right\}. \tag{7}$$

In many applications the mean of the θ_is may be non-zero, and so a natural generalization takes $\mathcal{G}$ to be the family of normal distributions

$$\mathcal{G}_{\text{norm}} := \left\{ g : g \sim \mathcal{N}\left(\mu, \sigma^2\right) \text{ for some } \sigma^2 \geq 0 \text{ and } \mu \in \mathbb{R} \right\}. \tag{8}$$

Estimating $g \in \mathcal{G}_{\text{norm}}$ boils down to estimating σ^2 and μ. For the homoskedastic case (5), the maximum likelihood estimates have closed-form solutions:

$$\hat{\mu} = \frac{1}{n} \sum_i x_i \tag{9}$$

$$\hat{\sigma}^2 = \max \left\{ 0, \frac{1}{n} \sum (x_i - \hat{\mu})^2 - s^2 \right\}. \tag{10}$$

When μ is fixed at zero this solution is similar to the one implied by the positive-part James-Stein estimator, with the difference that it divides $\sum x_i^2$ by n instead of $n - 2$ (Efron and Morris 1973). For the heteroskedastic case (1), the likelihood $L(\mu, \sigma^2)$ has a closed form but must be maximized numerically. In both cases, the posteriors (4) are normal distributions and are available analytically. Somewhat surprisingly, we were unable to find an existing software package that estimates $g \in \mathcal{G}_{\text{norm}}$, and so we provide an implementation in **ebnm**.

Although the normal family is simple, more flexible prior families are often required in practice. In particular, the means vector $\boldsymbol{\theta}$ is expected to be sparse in many applications, calling for prior families capable of capturing this sparsity. A common approach is to use "spike-and-slab" distributions — that is, mixture distributions consisting of two components, one a point mass at zero (the "spike") and the other belonging to some family of continuous distributions (the "slab", which is usually symmetric and centered at zero). In general Bayesian applications, a popular choice is the point-normal family

$$\mathcal{G}_{\text{pn}} := \left\{ g : g \sim \pi_0 \delta_0 + (1 - \pi_0)\mathcal{N}\left(0, \sigma^2\right) \text{ for some } 0 \leq \pi_0 \leq 1, \sigma^2 \geq 0 \right\}. \tag{11}$$

With this choice of $\mathcal{G}$, estimating g boils down to estimating π_0 and σ. Again, the likelihood $L(\pi_0, \sigma)$ has a closed form and can be maximized numerically (however, since the problem is not in general convex, a globally optimal solution is not guaranteed, and in general some work

is required to produce methods that reliably converge). Given $\hat{g} \in \mathcal{G}_{\mathrm{pn}}$, the posteriors (4) are mixtures of a point mass at zero and a normal distribution, and are available analytically. Again somewhat surprisingly, we were unable to find an existing implementation, and so we provide our own.

As Johnstone and Silverman (2005b) showed, better theoretical guarantees can be obtained for the normal means problem when the slab component has tails that are heavier than normal (specifically, exponential or heavier). The R package **EbayesThresh** (Johnstone and Silverman 2005a) implements two cases: the point-Laplace family

$$\mathcal{G}_{\mathrm{pl}} := \{g : g \sim \pi_0 \delta_0 + (1 - \pi_0)\mathrm{Laplace}(a) \text{ for some } 0 \leq \pi_0 \leq 1, a \geq 0\} \tag{12}$$

and a family of priors in which the slab component has Cauchy-like tails. Again the optimization problem is non-convex, and can be performed numerically; but in this case the posterior distributions are slightly more complex, involving mixtures of truncated normals, and we found it necessary to pay careful attention to numerical issues when computing summaries such as posterior means.

As an alternative to spike-and-slab approaches, Carvalho et al. (2010) introduced the horseshoe prior, which has heavy, Cauchy-like tails and enjoys good theoretical guarantees, but lacks a point mass at zero. (The horseshoe prior captures sparsity by having appreciable mass near zero, rather than exactly at zero.) The R package **horseshoe** (van der Pas et al. 2019) includes an implementation that solves the homoskedastic EBNM problem with $\mathcal{G}$ the family of horseshoe distributions $\mathcal{G}_{\mathrm{horseshoe}}$.

Nonparametric approaches to solving the EBNM problem have been investigated for nearly as long as these parametric approaches. Indeed, the problem of estimating g by the full nonparametric maximum likelihood estimate (NPMLE) — which corresponds to $\mathcal{G}$ being the family of *all* distributions, which we denote as $\mathcal{G} = \mathcal{G}_{\mathrm{npmle}}$ — has been studied theoretically since Kiefer and Wolfowitz (1956); see for example Laird (1978); Lindsay (1983); Jiang and Zhang (2009); Koenker and Mizera (2014); Dicker and Zhao (2016). To find the NPMLE in practice it is convenient to approximate $\mathcal{G}_{\mathrm{npmle}}$ by a dense (but finite) mixture of point masses:

$$\tilde{\mathcal{G}}_{\mathrm{npmle}} := \left\{ g : g \sim \pi_1 \delta_{\mu_1} + \ldots + \pi_K \delta_{\mu_K} \;\middle|\; \pi_1, \ldots, \pi_K \geq 0, \; \sum_{k=1}^{K} \pi_k = 1 \right\} \tag{13}$$

where $\mu_1, \ldots, \mu_K$ are a (fixed) dense grid of values on the real line spanning the range of the observations. Estimating $g \in \tilde{\mathcal{G}}_{\mathrm{npmle}}$ then amounts to solving the optimization problem

$$\text{maximize} \quad \mathbf{L}\boldsymbol{\pi} \tag{14}$$
$$\text{subject to} \quad \mathbf{0} \preceq \boldsymbol{\pi} \preceq \mathbf{1} \tag{15}$$
$$\mathbf{1}^T \boldsymbol{\pi} = 1, \tag{16}$$

where $\mathbf{L} \in \mathbb{R}^{n \times K}$ is the matrix with elements

$$\ell_{ik} = \frac{1}{\sqrt{2\pi s_i^2}} \exp\left(-\frac{(x_i - \mu_k)^2}{2s_i^2}\right). \tag{17}$$

This is a convex optimization problem (Koenker and Mizera 2014); the R package **REBayes** (Koenker and Gu 2017) implements an efficient solution based on interior point optimization routines from the commercial MOSEK library (MOSEK ApS 2019).

Although the fully nonparametric approach is attractive in its flexibility, the NPMLE is discrete (Laird 1978), and so not at all smooth. This is one motivation for more "constrained" nonparametric approaches, which can lead to smoother — and, arguably, more plausible — estimates of g. For example, in the context of multiple testing, Stephens (2017) argues for nonparametric families $\mathcal{G}$ constrained to distributions that are unimodal at zero. This includes the family of scale mixtures of normals

$$\mathcal{G}_{\text{smn}} := \left\{ g : g \sim \int_0^\infty \mathcal{N}\left(0, \sigma^2\right) \, dh\left(\sigma^2\right) \text{ for some distribution } h \right\} \tag{18}$$

as well as the family of all symmetric distributions that are unimodal at zero. For analogy with $\mathcal{G}_{\text{smn}}$, the latter can be represented as a family of scale mixtures of uniforms:

$$\mathcal{G}_{\text{symm-u}} := \left\{ g : g \sim \int_0^\infty \text{Unif}\left[-a, a\right] \, dh\left(a\right) \text{ for some distribution } h \right\}. \tag{19}$$

Approximating these families by finite mixtures with many components reduces the estimation of g to essentially the same convex optimization problem as for the NPMLE, which can be solved efficiently using convex methods. These families and others are implemented in the R package **ashr** (Stephens et al. 2020), which, in its current version, uses the convex optimization method implemented by the R package **mixsqp** (Kim et al. 2020).

Finally, a more flexible (non-unimodal) nonparametric method is implemented by the recent R package **deconvolveR** (Narasimhan and Efron 2020), which uses a natural spline basis to obtain a smooth estimate $\hat{g}$. In cases where the true prior is indeed smooth, this approach can outperform the NPMLE: see Koenker (2017) for examples where **deconvolveR** achieves a smaller Wasserstein distance between true and estimated priors than **REBayes**.

3. Implementation

Our **ebnm** package makes two key contributions. First, it provides implementations that solve the EBNM problem using several simple but useful prior families that, to our knowledge, have not previously been implemented (e.g., normal and point-normal families). Second, it collects previously existing implementations under a common interface with shared inputs and outputs. In some cases, we provide wrappers to existing packages; in others, we provide our own implementation with the aim of improving efficiency and/or reliability.

Consequently, **ebnm** offers a unified interface for solving the EBNM problem under a wide variety of prior assumptions. Inputs are as follows:

- `x`: A vector of observations x_i.

- `s`: A vector of standard errors s_i (or a scalar for the homoskedastic EBNM problem).

- `mode`: The location of the mode (for unimodal prior families). For most families, the mode can be estimated by setting `mode = "estimate"`.

- `scale`: Either a scale parameter (for parametric prior families) or the grid of parameters used to approximate a nonparametric prior family. For example, the family of scale

mixtures of normals $\mathcal{G}_{smn}$ is approximated via the family of finite mixtures

$$\tilde{\mathcal{G}} = \left\{ g : g \sim \pi_1 \mathcal{N}\left(0, \sigma_1^2\right) + \ldots + \pi_K \mathcal{N}\left(0, \sigma_K^2\right) \;\middle|\; \pi_1, \ldots, \pi_K \geq 0, \; \sum_{k=1}^{K} \pi_k = 1 \right\}, \quad (20)$$

where $\{\sigma_1^2, \ldots, \sigma_K^2\}$ is a sufficiently dense grid of scale parameters that is fixed in advance (Stephens 2017). Similarly, the family of symmetric unimodal distributions $\mathcal{G}_{symm\text{-}u}$ can be approximated by a family of finite mixtures of uniform distributions with mean zero, while the family of all distributions $\mathcal{G}_{npmle}$ is typically approximated by finite mixtures of point masses (Jiang and Zhang 2009; Koenker and Gu 2017). By default, **ebnm** estimates the scale/grid, using strategies for grid selection discussed by Willwerscheid (2021a).

- `g_init`: An optional initial value of $\hat{g}$ used during optimization.

- `fix_g`: When an initial value of $\hat{g}$ is supplied, it can be fixed in order to return desired outputs for that value.

- `output`: The desired outputs.

Available outputs are:

- `posterior`: Summary statistics for the posteriors $p(\theta_i \mid x_i, s_i, \hat{g})$, including posterior means and standard deviations as well as local false sign rates, defined by Stephens (2017) as
$$\text{lfsr}(i) := \min\left\{ p(\theta_i \leq 0 \mid x_i, s_i, \hat{g}), p(\theta_i \geq 0 \mid x_i, s_i, \hat{g}) \right\}. \quad (21)$$

- `fitted_g`: The fitted prior $\hat{g}$.

- `log_likelihood`: The data log-likelihood
$$\log p(x_1, \ldots, x_n | \hat{g}, s_1, \ldots, s_n) = \sum_i \log p\left(x_i \mid s_i, \hat{g}\right). \quad (22)$$

- `posterior_sampler`: A function that can be used to generate samples from the posteriors
$$p(\theta_i | \hat{g}, x_i, s_i) \propto \hat{g}(\theta_i) p(x_i | \theta_i, s_i). \quad (23)$$

Table 1 lists all prior families implemented in **ebnm**. The table is partitioned into two main groups. The first, which is discussed in Section 3.1, consists of "parametric" prior families, which include spike-and-slab families as well as the single-component normal and horseshoe distributions. The second, "nonparametric" group (discussed in Section 3.2) can be further divided into unimodal families, which are useful for modeling sparse data and for controlling false discovery rates (Efron 2010; Stephens 2017), and non-unimodal families, which make no assumptions about sparsity.

3.1. Parametric families

Parametric prior families available in **ebnm** include the normal, point-normal, point-Laplace, point-exponential, and horseshoe families. For the normal, point-normal, and point-Laplace

Function	Form of Prior	Package	Sign	Symm.
normal	$\mathcal{N}\left(\mu, \sigma^2\right)$	**ebnm**		✓
point_normal	$\pi_0 \delta_\mu + (1 - \pi_0)\mathcal{N}\left(\mu, \sigma^2\right)$	**ebnm**		✓
point_laplace	$\pi_0 \delta_\mu + (1 - \pi_0)\operatorname{Laplace}\left(\mu, a\right)$	**ebnm**		✓
point_exponential	$\pi_0 \delta_0 + (1 - \pi_0)\operatorname{Exp}\left(a\right)$	**ebnm**	+	
horseshoe	$\operatorname{Horseshoe}\left(\tau\right)$	**horseshoe**		✓
normal_scale_mixture	$\int_0^\infty \mathcal{N}\left(0, \sigma^2\right)\ dh\left(\sigma^2\right)$	**ebnm**		✓
unimodal_symmetric	$\int_0^\infty \operatorname{Unif}[-a, a]\ dh(a)$	**ashr**		✓
unimodal	$\int_{-\infty}^\infty \operatorname{Unif}[0, a]\ dh(a)$	**ashr**		
unimodal_nonnegative	$\int_0^\infty \operatorname{Unif}[0, a]\ dh(a)$	**ashr**	+	
unimodal_nonpositive	$\int_0^\infty \operatorname{Unif}[-a, 0]\ dh(a)$	**ashr**	-	
npmle	$\int_{-\infty}^\infty \delta_x\ dh(x)$	**ebnm**		
deconvolver	Narasimhan and Efron (2020)	**deconvolveR**		

Table 1: Prior families implemented in **ebnm**. "Sign" indicates the family's support (nonnegative or real-valued). "Symm." indicates whether the family is symmetric about its mode.

families, the mean can be either estimated or fixed at zero. We implemented our own solutions for all but the horseshoe family. For the latter, we supply a wrapper to R package **horseshoe** (van der Pas et al. 2019), which implements only the homoskedastic case (5). (While package **EbayesThresh** implements a solution using the point-Laplace family, it lacks key outputs such as the data log-likelihood and posterior standard deviations. Further, we were able to obtain improvements in speed and reliability in our re-implementation. For details, see the supplementary benchmarking results in Section 6.)

The only case in which a closed-form solution exists is the homoskedastic case (5) with the prior family $\mathcal{G}$ taken to be the family of normal distributions. In all other cases, no more than three parameters need to be estimated — the scale of the slab component, the spike/slab mixture proportions, and possibly the mode — so that off-the-shelf optimizers are largely sufficient provided that some attention is paid to numerical issues. For example, we found that transforming parameters to unconstrained variables typically improves stability, and that analytic calculations of Hessians, though available, can be slower than using numerical methods.

By default, **ebnm** uses `nlm`, a Newton-type method included in package **stats**, with gradients calculated analytically and Hessians estimated numerically. We also implemented approaches using the L-BFGS-B algorithm (via the **stats** function `optim`) and a trust region method (via the package **trust**; Geyer (2020)). In each of our experiments, the default was either the fastest method or differed from the fastest by less than a factor of two. Further, both `nlm` and `trust` are considerably more stable than `optim`, particularly for the null case where the "true" prior is a point mass at zero. For details, see Section 6.

3.2. Nonparametric families

The nonparametric families implemented in **ebnm** can be divided into two broad subgroups.

The first is comprised of unimodal families, including scale mixtures of normals as well as symmetric, nonnegative, and nonpositive unimodal prior families. The second subgroup is not constrained to be unimodal, and includes the NPMLE and the spline-based family implemented in R package **deconvolveR** (Narasimhan and Efron 2020).

The unimodal families are all implemented in package **ashr** (Stephens et al. 2020), and with the exception of the family of scale mixtures of normals, **ebnm** leans on these implementations to solve the EBNM problem for unimodal families. For scale mixtures of normals, we re-implemented the **ashr** algorithm with the aim of improving efficiency. By focusing only on this special case, we were able to reduce the run-time by half (again, see the supplementary benchmarking results in Section 6).

The NPMLE can also be estimated using **ashr**, but the user must specify the grid of point masses and, as with scale mixtures of normals, there are a number of inefficiencies in the implementation. Package **REBayes** (Koenker and Gu 2017) was developed with the NPMLE more particularly in mind, but it uses the commercial interior-point solver **MOSEK** (MOSEK ApS 2019). Thus we re-implemented the NPMLE in **ebnm** using the open source package **mixsqp** to solve the core optimization problem (14)-(16). When the number of mixture components is not too large, **mixsqp** tends to be faster than the interior point method; however, the R implementation of **mixsqp** — which is slower than the Julia implementation — can be outpaced by **REBayes** when there are more than 100 or so mixture components (again see Section 6). Despite the superior performance of **REBayes** for large K, we use **mixsqp** as the default optimization function for all nonparametric prior families to provide a fully open-source toolkit that avoids the need to install **MOSEK**.

3.3. Sign-constrained priors

In most applications of the EBNM problem, the means can be positive, negative, or zero. But there are also applications in which, for either physical or statistical reasons, the means are constrained to be nonnegative (or, possibly, nonpositive). Our examples in Section 5 include one setting where such a constraint may be helpful.

To accommodate such situations, we implemented two nonnegative prior families in **ebnm**: the point-exponential prior family

$$\mathcal{G}_{\mathrm{pe}} := \{g : g \sim \pi_0 \delta_0 + (1 - \pi_0)\mathrm{Exp}(a) \text{ for some } 0 \leq \pi_0 \leq 1, a \geq 0\}; \qquad (24)$$

and the family of nonnegative distributions that are unimodal at zero, which, similarly to the family of symmetric unimodal distributions $\mathcal{G}_{\mathrm{symm\text{-}u}}$, can be represented as a family of mixtures of uniforms:

$$\mathcal{G}_{\mathrm{nn}} := \left\{ g : g \sim \int_0^\infty \mathrm{Unif}\,[0, a] \; dh\,(a) \text{ for some distribution } h \right\}. \qquad (25)$$

While nonpositive constraints can be enforced by solving the EBNM problem for $-\mathbf{x}$ using a nonnegative prior family, we've included a nonpositive version of the nonparametric unimodal family (25) for convenience.

4. Prior families comparisons

This section provides some comparisons of the different prior families available in **ebnm**. The main goal of the package is to provide a convenient interface for fitting the EBNM model using a range of different prior families, and so our aim here is simply to provide some basic comments and results to help guide users. In particular, we aim neither to perform a systematic comparison across the full range of potential applications, nor to argue that any one family is uniformly superior to any other.

4.1. Flexibility

Clearly, some prior families are more flexible than others. Indeed, many of the families are nested, which provides a formal ordering. For example,

$$\mathcal{G}_{\text{norm0}} \subseteq \mathcal{G}_{\text{pn}} \subseteq \mathcal{G}_{\text{smn}} \subseteq \mathcal{G}_{\text{symm-u}} \subseteq \mathcal{G}_{\text{npmle}}, \tag{26}$$

where the prior families $\mathcal{G}$ are, respectively, the family of zero-mean normals, point-normal priors, scale mixtures of normals, symmetric unimodal priors, and the nonparametric family of all distributions. In addition, the families of point-Laplace priors $\mathcal{G}_{\text{pl}}$ and horseshoe priors $\mathcal{G}_{\text{horseshoe}}$ are each contained in $\mathcal{G}_{\text{smn}}$, and $\mathcal{G}_{\text{deconvolver}} \subseteq \mathcal{G}_{\text{npmle}}$.

Flexibility is certainly important: when an insufficiently flexible prior is applied in a setting where its assumptions are violated, inference can be poor. On the other hand, overly flexible families sometimes have the potential to "overfit." For example, the fully nonparametric family $\mathcal{G}_{\text{npmle}}$ has the feature that estimates of g are discrete distributions, which also leads to discrete posteriors (4). Whether or not this is problematic will depend on the end goal: for example, the posterior mean from a discrete prior may be perfectly adequate for point estimation even when interval estimates perform less well (see, for example, Koenker 2017).

Ideally, it would be possible to use the data to identify an appropriate prior family. In this respect, the log-likelihoods of fits computed by **ebnm** for different families may be helpful. For example, comparing the log-likelihood for a parametric family with a nonparametric alternative may help identify gross deviations from the parametric assumptions. However, formal testing is difficult; in particular, Wilks's theorem does not apply to these nonparametric comparisons.

4.2. Numerical illustrations

To illustrate some of these issues we simulated data under three different data-generating distributions:

- **Point-normal**. A point-normal prior with 90% sparsity:

$$0.9\delta_0 + 0.1\mathcal{N}\left(0, 2^2\right). \tag{27}$$

- **Point-t**. A 80-20 mixture of a point mass at zero and a scaled t_5 distribution:

$$0.8\delta_0 + 0.2(1.5t_5). \tag{28}$$

- **Asymmetric tophat**. A 50-50 mixture of a point mass at zero and a uniform distribution on $[-5, 10]$

$$0.5\delta_0 + 0.5\text{Unif}[-5, 10]. \tag{29}$$

Function	Point-normal			Point-t			Asymmetric tophat		
	LogLik	RMSE	ClCov	LogLik	RMSE	ClCov	LogLik	RMSE	ClCov
ebnm_normal	–28.2	0.536	0.942	–62.5	0.663	0.927	–277.1	0.971	0.898
ebnm_point_normal	–3.4	0.443	0.947	–7.1	0.531	0.896	–122.5	0.841	0.914
ebnm_point_laplace	–3.9	0.443	0.954	–5.5	0.527	0.920	–146.4	0.849	0.922
ebnm_normal_scale_mixture	–3.3	0.442	0.951	–4.5	0.527	0.922	–126.0	0.841	0.914
ebnm_unimodal_symmetric	–2.8	0.444	0.942	–4.1	0.528	0.900	–103.0	0.833	0.906
ebnm_unimodal	–1.0	0.445	0.942	–1.7	0.529	0.902	–5.4	0.820	0.907
ebnm_npmle	0.0	0.450	0.892	0.0	0.534	0.789	0.0	0.830	0.801
ebnm_horseshoe	–14.1	0.446	0.946	–17.1	0.532	0.917	–269.3	0.864	0.911
ebnm_deconvolver	–8.0	0.476	0.951	–18.5	0.582	0.928	–36.9	0.881	0.899

Figure 1: Simulation results using three different data-generating distributions. Shown for each distribution are the average data log-likelihood relative to the NPMLE, the root mean-squared error, and the proportion of 90% credible intervals that contain the true mean.

For each of the three data-generating distributions listed above, we ran ten simulations in which we simulated $n = 1000$ "true means" θ_i, added $\mathcal{N}(0, 1)$ noise, and then estimated $\boldsymbol{\theta}$ using various prior families. In Figure 1, we show the mean log-likelihood attained by each prior family relative to the `ebnm_npmle` log-likelihood; the root mean-squared error

$$\text{RMSE} := \sqrt{\sum_{i=1}^{n} \left(\hat{\theta}_i - \theta_i \right)^2}, \tag{30}$$

where $\hat{\theta}_i$ is the posterior mean $\mathbb{E}\left(\theta_i \mid x_i, s_i, \hat{g}\right)$; and the proportion of true means that are covered by 90% credible intervals (which were obtained via the posterior samplers returned by **ebnm**). (Although the NPMLE posterior has discrete support, we took a 90% interval to be the continuous region between the estimated 5% and 95% quantiles of the distribution.)

We focus first on the log-likelihoods. In every case, `ebnm_npmle` attains the maximum log-likelihood, which it should do because $\mathcal{G}_{\text{npmle}}$ includes every other prior family as a subfamily. Therefore we show all other log-likelihoods as differences relative to this optimum. Note also that the log-likelihoods generally obey the inequalities implied by the nesting (26), which is a helpful check that the optimizers are behaving as expected. (Note that **deconvolveR** sometimes shows worse log-likelihoods than less flexible parametric models; this is because it optimizes a penalized log-likelihood, and so there is no guarantee that the unpenalized log-likelihood it achieves will exceed that of simpler models.) Gross deviations from the underlying model are consistently reflected in large average log-likelihood differences (say, more than ten log-likelihood units). For example, the normal model is clearly inadequate in all three cases, whereas the point-normal model shows gross deviations only for the last example (and, indeed, all the symmetric models show poor log-likelihoods in this asymmetric case).

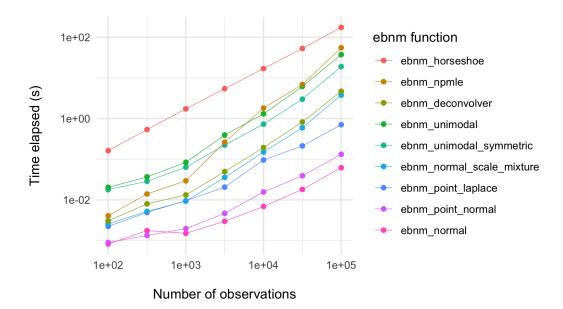Turning to RMSE, we note that all methods provide an improvement relative to the maximum

Figure 2: run-time for variously sized datasets and for different choices of prior family.

likelihood estimate (which would have RMSE of 1 here), even when there is a gross misfit between the model and the simulation scenario. As suggested above, one might expect the simpler parametric models to outperform the more flexible models when their assumptions hold, and to under-perform them when their assumptions do not hold. Although the trend is in this direction, overall the differences in RMSE are often small, with the most striking pattern being that all methods consistently outperform the simple normal prior in these non-normal settings.

As for the coverage of credible intervals, we also see generally good robustness to modeling assumptions: even the normal prior shows reasonable coverage in all three settings. Only the NPMLE shows substantial "under-coverage" in any setting, presumably because it estimates a discrete prior that leads to understatements of uncertainty.

4.3. Timing comparisons

When the EBNM problem only needs to be solved once, run-time is not usually a major concern, but in applications that must iteratively solve a large number of EBNM problems — such as the matrix factorization application discussed in Section 5.2 — speed can be important. Therefore we provide a brief assessment of run-time for different prior families. We fit **ebnm** to datasets simulated from a point-t prior (28) with the number of observations ranging from 10^2 to 10^5 (similar trends were observed for different data-generating mechanisms). All trials were performed on a 2018 MacBook Pro with a 2.6 GHz Intel Core i7 processor and 16 GB of DDR4-2400 RAM.

Results are displayed in Figure 2. For any given number of observations, the run-times for the different prior families span 2-3 orders of magnitude, with the normal being fastest and the horseshoe being slowest. The point-normal family is nearly as fast as the normal, and several times faster than the next fastest method, the point-Laplace. Despite the fact that the nonparametric prior families (other than the **deconvolveR** family) are fit by solving a convex
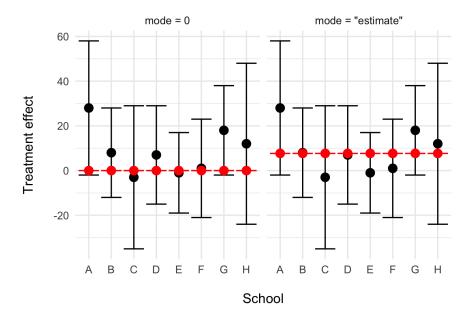
Figure 3: Results for the eight schools example (Rubin 1981) with mode fixed at zero (left) or estimated (right). The data are shown in black: points indicate treatment effect estimates and error bars are ± 2 standard errors. The `ebnm_point_normal` estimates of the "true" means and 95% credible intervals are in red. Since, in both cases, $\hat{g}$ is essentially a point mass, the credible intervals are very narrow.

optimization problem using relatively sophisticated convex solvers, they remain appreciably slower than the point-normal family.

5. Examples

5.1. Eight schools

We first consider the "eight schools" dataset from Rubin (1981), which is also analyzed in Gelman et al. (2014). In each of eight US high schools, different randomized experiments were conducted in order to estimate the effect of coaching programs on SAT scores. The pooling of results leads to an example of "meta-analysis," which draws quantitative conclusions from a review of previous experiments, usually performed in parallel and often using different protocols (Glass 1976). The approach has long been used in psychology and the social sciences, and has since become widely used in medical research as well (Cooper 1982; Rosenthal 1984; Haidich 2010).

The eight schools dataset includes estimates of treatment effects and standard errors. We use a point-normal prior family to fit the data with, first, the mode fixed at zero and, second, the mode to be estimated. Since the first prior family models sparsity it can be used to assess whether treatment effects are non-null, whereas the second prior family is potentially more useful for modelling variation across schools.

Results are shown in Figure 3. In both cases, the prior g is estimated to be a point mass

(either δ_0 or δ_μ with $\mu \approx 7.7$). This simple result is immediately useful, because it already suggests that the data are consistent with no variation in treatment effect across schools. The difference in log-likelihoods between the two models is about 1.8, so the prior δ_μ achieves a likelihood about $e^{1.8} \approx 6$ times higher than the prior δ_0. This represents, at best, modest support for a non-zero average treatment effect.

A known limitation of EB methods is that they do not account for uncertainty in the estimate of g when computing posterior distributions for each mean, which can lead them to under-represent uncertainty in these means. In this case, where the estimated g is a point mass, this effect is especially pronounced, as the EB credible intervals are also single points and thus useless for practical purposes. If one really wants sensible credible intervals for each mean here, then a fully Bayesian method — perhaps using MCMC to integrate over uncertainty in g — would be necessary.

Figure 3 can be generated as follows:

```
library(tidyverse)

set.seed(8) # For reproducibility of sampler results.

x <- c(28, 8, -3, 7, -1, 1, 18, 12) # Observations.
s <- c(15, 10, 16, 11, 9, 11, 10, 18) # Standard errors.
n <- length(x)

# By default, a posterior sampler isn't returned, so we set
#   output = output_all().
ebnm_res_mode0 <- ebnm::ebnm_point_normal(
  x = x, s = s, mode = 0, output = output_all()
)
nsamp <- 10000
mode0_samp <- ebnm_res_mode0$posterior_sampler(nsamp)
mode0_CI <- apply(mode0_samp, 2, quantile, c(.025, .975))

ebnm_res_estmode <- ebnm::ebnm_point_normal(
  x = x, s = s, mode = "estimate", output = output_all()
)
estmode_samp <- ebnm_res_estmode$posterior_sampler(nsamp)
estmode_CI <- apply(estmode_samp, 2, quantile, c(.025, .975))

modes <- paste("mode =", c("0", "\"estimate\""))
tib <- tibble(
  idx = factor(rep(LETTERS[1:n], 2)),
  x = rep(x, 2),
  s = rep(s, 2),
  theta_hat = c(
    ebnm_res_mode0$posterior$mean, ebnm_res_estmode$posterior$mean
  ),
  CI_lower = c(mode0_CI[1, ], estmode_CI[1, ]),
```

```
  CI_upper = c(mode0_CI[2, ], estmode_CI[2, ]),
  mode = factor(rep(modes, each = n), levels = modes)
)

ggplot(tib, aes(x = idx, y = x)) +
  geom_point(size = 3) +
  geom_errorbar(aes(ymin = x - 2 * s, ymax = x + 2 * s)) +
  geom_point(aes(x = idx, y = theta_hat), color = "red", size = 3) +
  geom_errorbar(aes(ymin = CI_lower, ymax = CI_upper), color = "red") +
  theme_minimal() +
  labs(x = "\nSchool", y = "Treatment effect\n") +
  facet_wrap(~mode)
```

5.2. Flexible empirical Bayes matrix factorization

In our second example we aim to illustrate some of the benefits that accrue by having a common interface to solve the EBNM problem for different prior families. The motivation here is that solving the EBNM problem sometimes arises as a subroutine within another method — here, matrix factorization — and that by plugging in different versions of the EBNM solver one can quickly and easily experiment with different versions of the method. (Another setting where this arises, and where the common interface of **ebnm** could be exploited, is wavelet denoising: one could easily experiment with different sparse priors and compare the resulting shrinkage of wavelet coefficients.)

Our example is based on the empirical Bayes matrix factorization (EBMF) framework introduced by Wang and Stephens (2021), which considers the following model for a data matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$:

$$\mathbf{X} = \mathbf{L}\mathbf{F}' + \mathbf{E} \tag{31}$$

$$e_{ij} \sim \mathcal{N}\left(0, \sigma^2\right) \tag{32}$$

$$\ell_{ik} \sim g_\ell^{(k)} \in \mathcal{G}_\ell \tag{33}$$

$$f_{jk} \sim g_f^{(k)} \in \mathcal{G}_f. \tag{34}$$

We refer to $\mathbf{L} \in \mathbb{R}^{n \times K}$ as the matrix of "loadings" and $\mathbf{F} \in \mathbb{R}^{p \times K}$ as the matrix of "factors." Typically, K is much smaller than either n or p. Fitting this EB model involves estimating the priors $g_\ell^{(k)}, g_f^{(k)}$ $(k = 1, \ldots, K)$ as well as the loadings and factors $\mathbf{L}, \mathbf{F}$. Different choices of prior families $\mathcal{G}_\ell$ and $\mathcal{G}_f$ lead to different types of factorization. For example, if both these families are assumed to be normal ($\mathcal{G}_\ell = \mathcal{G}_f = \mathcal{G}_{\mathrm{norm0}}$) then the resulting estimates for $\mathbf{L}$ and $\mathbf{F}$ will be closely related to the truncated singular value decomposition (SVD) (Nakajima and Sugiyama 2011). Alternatively, point-normal priors lead to EB versions of sparse SVD or sparse factor analysis (Wang and Stephens 2021).

Wang and Stephens (2021) showed how, by using variational approximations, the EBMF model (31) can be fit by solving a series of EBNM problems. In brief, for each $k = 1, \ldots, K$ the algorithm iterates between solving an EBNM problem to estimate the prior $g_\ell^{(k)}$ and corresponding loadings, and solving another EBNM problem to estimate the prior $g_f^{(k)}$ and corresponding factors. This iteration is repeated until convergence for each k before moving

to the next k, which means that the EBNM problem may be solved hundreds of times for each fit. Thus EBMF requires highly robust and fast implementations of EBNM solvers, such as those provided in **ebnm**. We've developed an R package **flashier** (Willwerscheid 2021a,b) that fits EBMF models using this strategy; the user simply specifies which EBNM solver to use for each loading and which to use for each factor. (If desired, different EBNM solvers can be specified for each value of k, generalizing the model (31).) The result is a highly flexible EB approach to matrix factorization that includes many previous implementations as special cases, but also many novel combinations — in the example below, we use nonnegative priors for **L** ($\mathcal{G}_\ell = \mathcal{G}_{\mathrm{pe}}$) and point-normal priors for **F**, which yields a semi-nonnegative matrix factorization (Ding et al. 2010; Wang et al. 2019; He et al. 2020). As far as we know, ours is the first implementation of an EB approach to semi-nonnegative matrix factorization.

We illustrate these methods on the GTEx dataset that also serves as the primary example in Wang and Stephens (2021). This dataset is derived from data made available by the Genotype Tissue Expression (GTEx) project (Lonsdale et al. 2013), which provides z-scores for assessing the significance of effects of genetic variants ("single nucleotide polymorphisms" or SNPs) on gene expression across 44 human tissues. To reduce the data to a more manageable size, Urbut et al. (2019) choose the "top" SNP for each gene — that is, the SNP associated with the largest (absolute) z-score over all 44 tissues. This yields a $44 \times 16{,}069$ matrix of z-scores, with rows corresponding to tissues and columns corresponding to SNP-gene pairs.

The goal of applying the EBMF model (31) to these data is to help elucidate patterns of sharing of significant genetic effects among the different biological tissues. For example, some SNPs may show strong effects across all tissues, whereas other SNPs may show strong effects only across a subset of tissues (e.g., a subset of brain tissues). The hope is that each factor/loading may capture some such pattern of sharing, and perhaps reveal common underlying biology among tissues. Some of these patterns may be sparse, both in the sense of involving only a small fraction of all tissues and involving only a fraction of the SNPs. This fact motivated Wang and Stephens (2021) to analyze these data with sparse (point-normal) priors on both factors and loadings. Here we note that there is also reason to expect that shared patterns may tend to have the same sign: that is, when a genetic variant has an effect in several tissues, it is biologically likely that the sign of the effect in different tissues will be the same. This motivates the use of semi-nonnegative matrix factorization, in which the priors on the loadings are nonnegative (here, point-exponential), but the factors are not so constrained (because the shared effects at each gene/SNP may be either all positive or all negative). Fitting this (sparse) semi-nonnegative matrix factorization model is made possible — indeed, straightforward — by our implementation of the point-exponential prior in **ebnm**.

To illustrate these ideas, we ran EBMF with two different sets of priors: first, point-normal priors for both loadings and factors (the "point-normal" factorization), and second, point-exponential priors for loadings and point-normal priors for factors (the semi-nonnegative factorization). Results are shown in Figure 4. We sorted semi-nonnegative results by descending proportion of variance explained and then "matched" semi-nonnegative factors with point-normal factors for ease of comparison. While results are similar, many of the semi-nonnegative factors are noticeably sparser (2, 6, 7, 8, 10, 12, 15, 17), while only a small number of factors get somewhat busier (11, 16). (We note, however, that it is possible that "busyness" is biologically relevant in some cases.) The greatest advantage to SNMF, we would argue, is the lack of anti-correlations, which are somewhat difficult to motivate biologically in this particular setting. Consider Factor 10, for example: the semi-nonnegative factorization
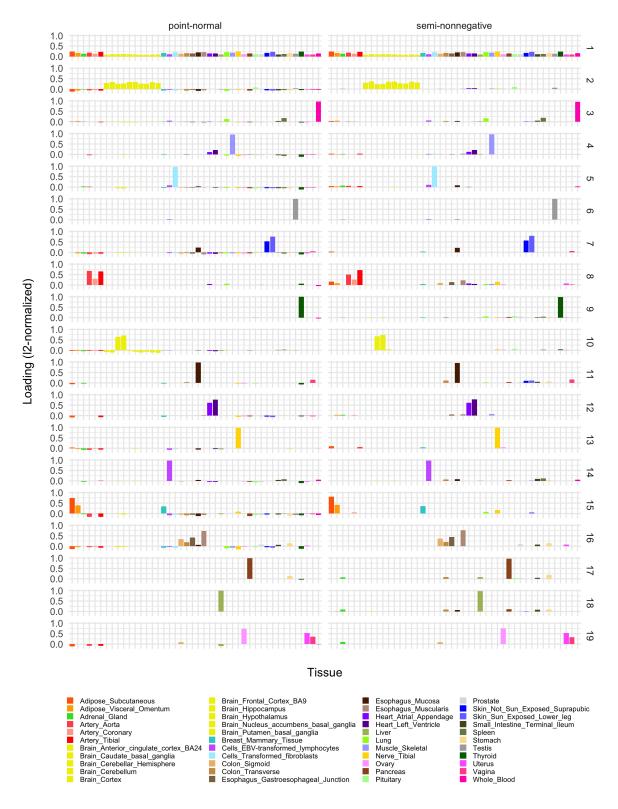
Figure 4: Results for the GTEx example. Semi-nonnegative factors are sorted by descending proportion of variance explained and then point-normal factors are "matched" to their semi-nonnegative counterparts.

suggests only that there is correlation of eQTL effect sizes in cerebellar hemisphere and cerebellum tissues, while the point-normal factorization suggests that eQTLs that increase gene activity in cerebellar tissues also *decrease* it in other brain tissues, which is biologically less plausible.

The code used to generate Figure 4 is reproduced below.

```
library(tidyverse)
library(flashier)

url_prefix <- "https://github.com/stephenslab/gtexresults/blob/master/data/"
gtex_url <- paste0(url_prefix, "MatrixEQTLSumStats.Portable.Z.rds?raw=TRUE")
colors_url <- paste0(url_prefix, "GTExColors.txt?raw=TRUE")

gtex <- readRDS(gzcon(url(gtex_url)))
strong <- t(gtex$strong.z) # Dataset used by Urbut et al. and Wang & Stephens.

gtex.colors <- read_tsv(colors_url, col_names = c("Tissue", "Hex", "RGB")) %>%
  mutate(Tissue = str_remove_all(Tissue, "[\\(\\)\\-]")) %>%
  mutate(Tissue = str_replace_all(Tissue, " +", "_")) %>%
  pull(Hex, name = Tissue)
gtex.colors <- gtex.colors[rownames(strong)]

# Point-normal factorization.
pn_res <- flash(
  strong,
  S = 1,
  greedy.Kmax = 50,
  prior.family = as.prior(ebnm::ebnm_point_normal),
  var.type = 2,
  backfit = TRUE
)

# Semi-nonnegative factorization.
snn_res <- flash(
  strong,
  S = 1,
  greedy.Kmax = 50,
  prior.family = c(
    as.prior(ebnm::ebnm_point_exponential, sign = 1),
    as.prior(ebnm::ebnm_point_normal)
  ),
  var.type = 2,
  backfit = TRUE
)

# Extract loadings (tissues).
pn_LL <- pn_res$loadings.pm[[1]][, order(-pn_res$pve)]
```

```
snn_LL <- snn_res$loadings.pm[[1]][, order(-snn_res$pve)]

# Flip loadings so that the largest loadings are positive.
pn_sign <- 2L * (apply(pn_LL, 2, max) > -apply(pn_LL, 2, min)) - 1
pn_LL <- t(t(pn_LL) * pn_sign)

# Rearrange factors to make side-by-side comparison easier.
LL_cor <- abs(cor(pn_LL, snn_LL))
pn_LL <- pn_LL[, apply(LL_cor, 2, which.max)]

colnames(pn_LL) <- paste0("Factor", 1:ncol(pn_LL))
colnames(snn_LL) <- paste0("Factor", 1:ncol(snn_LL))

pn_tib <- as_tibble(pn_LL) %>% add_column(PriorFamily = "point-normal")
snn_tib <- as_tibble(snn_LL) %>% add_column(PriorFamily = "semi-nonnegative")
tib <- pn_tib %>% bind_rows(snn_tib)

tib <- tib %>%
  mutate(Tissue = rep(rownames(strong), 2)) %>%
  pivot_longer(
    -c(PriorFamily, Tissue), names_to = "Factor", values_to = "Loading"
  ) %>%
  mutate(Factor = as.numeric(str_remove(Factor, "Factor")))

plt <- ggplot(tib, aes(x = Tissue, y = Loading, fill = Tissue)) +
  geom_col() +
  facet_grid(rows = vars(Factor), cols = vars(PriorFamily)) +
  scale_fill_manual(values = gtex.colors) +
  scale_y_continuous(breaks = c(0, 0.5, 1)) +
  theme_minimal() +
  labs(x = "\nTissue", y = "Loading (l2-normalized)\n") +
  guides(fill = guide_legend(ncol = 4)) +
  theme(legend.position = "bottom",
        legend.key.size = unit(0.5, "lines"),
        legend.text = element_text(size = 6.5),
        legend.title = element_blank(),
        axis.text.x = element_blank())
```

References

Carvalho, C. M., Polson, N. G., and Scott, J. G. (2010). The horseshoe estimator for sparse signals. *Biometrika*, 97(2):465–480.

Clyde, M. and George, E. I. (2000). Flexible empirical bayes estimation for wavelets. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 62(4):681–698.

Cooper, H. M. (1982). Scientific guidelines for conducting integrative research reviews. *Review of Educational Research*, 52(2):291–302.

Dicker, L. H. and Zhao, S. D. (2016). High-dimensional classification via nonparametric empirical Bayes and maximum likelihood inference. *Biometrika*, 103(1):21–34.

Ding, C. H., Li, T., and Jordan, M. I. (2010). Convex and semi-nonnegative matrix factorizations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):45–55.

Efron, B. (2010). *Large-scale inference*, volume 1 of *Institute of Mathematical Statistics (IMS) Monographs*. Cambridge University Press, Cambridge.

Efron, B. and Morris, C. (1973). Stein's estimation rule and its competitors—an empirical Bayes approach. *Journal of the American Statistical Association*, 68:117–130.

Gelman, A., Carlin, J. B., Stern, H. S., Dunson, D. B., Vehtari, A., and Rubin, D. B. (2014). *Bayesian data analysis*. Texts in Statistical Science Series. CRC Press, Boca Raton, FL, third edition.

Geyer, C. J. (2020). *trust: Trust Region Optimization*. R package version 0.1-8.

Glass, G. V. (1976). Primary, secondary, and meta-analysis of research. *Educational Researcher*, 5:3–8.

Haidich, A. B. (2010). Meta-analysis in medical research. *Hippokratia*, 14(Suppl 1):29–37.

He, Y., Chhetri, S. B., Arvanitis, M., Srinivasan, K., Aguet, F., Ardlie, K. G., Barbeira, A. N., Bonazzola, R., Im, H. K., Brown, C. D., et al. (2020). sn-spmf: matrix factorization informs tissue-specific genetic regulation of gene expression. *Genome biology*, 21(1):1–25.

James, W. and Stein, C. (1961). Estimation with quadratic loss. In *Berkeley Symposium on Mathematical Statistics and Probability, 1961*, pages 361–379, Berkeley. University of California Press.

Jiang, W. and Zhang, C.-H. (2009). General maximum likelihood empirical Bayes estimation of normal means. *Annals of Statistics*, 37(4):1647–1684.

Johnstone, I. (2019). Gaussian estimation: Sequence and wavelet models. Unpublished manuscript.

Johnstone, I. and Silverman, B. W. (2005a). Ebayesthresh: R programs for empirical bayes thresholding. *Journal of Statistical Software*, 12(8):1–38.

Johnstone, I. M. and Silverman, B. W. (2004). Needles and straw in haystacks: empirical Bayes estimates of possibly sparse sequences. *Annals of Statistics*, 32(4):1594–1649.

Johnstone, I. M. and Silverman, B. W. (2005b). Empirical Bayes selection of wavelet thresholds. *Annals of Statistics*, 33(4):1700–1752.

Kiefer, J. and Wolfowitz, J. (1956). Consistency of the maximum likelihood estimator in the presence of infinitely many incidental parameters. *Annals of Mathematical Statistics*, 27:887–906.

Kim, Y., Carbonetto, P., Stephens, M., and Anitescu, M. (2020). A fast algorithm for maximum likelihood estimation of mixture proportions using sequential quadratic programming. *Journal of Computational and Graphical Statistics*, 29(2):261–273.

Koenker, R. (2017). Bayesian deconvolution: an R vinaigrette. Technical report, cemmap working paper.

Koenker, R. and Gu, J. (2017). REBayes: An R package for empirical Bayes mixture methods. *Journal of Statistical Software*, 82(8):1–26.

Koenker, R. and Mizera, I. (2014). Convex optimization, shape constraints, compound decisions, and empirical Bayes rules. *Journal of the American Statistical Association*, 109(506):674–685.

Laird, N. (1978). Nonparametric maximum likelihood estimation of a mixing distribution. *Journal of the American Statistical Association*, 73(364):805–811.

Lindsay, B. G. (1983). The geometry of mixture likelihoods: a general theory. *Annals of Statistics*, pages 86–94.

Lonsdale, J., Thomas, J., Salvatore, M., et al. (2013). The genotype-tissue expression (GTEx) project. *Nature Genetics*, 45(6):580–585.

Mersmann, O. (2019). *microbenchmark: Accurate Timing Functions*. R package version 1.4-7.

Morris, C. N. (1983). Parametric empirical bayes inference: Theory and applications. *Journal of the American Statistical Association*, 78(381):47–55.

MOSEK ApS (2019). *Rmosek: The R to MOSEK Optimization Interface*. R package version 9.1.0.

Nakajima, S. and Sugiyama, M. (2011). Theoretical analysis of Bayesian matrix factorization. *Journal of Machine Learning Research*, 12:2583–2648.

Narasimhan, B. and Efron, B. (2020). deconvolver: A g-modeling program for deconvolution and empirical Bayes estimation. *Journal of Statistical Software*, 94(11):1–20.

Robbins, H. (1951). Asymptotically subminimax solutions of compound statistical decision problems. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, 1951, vol. II*, pages 131–149. University of California Press, Berkeley and Los Angeles.

Robbins, H. (1956). An empirical Bayes approach to statistics. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, 1956, vol. I*, pages 157–163. University of California Press, Berkeley and Los Angeles.

Rosenthal, R. (1984). *Meta-Analytic Procedures for Social Science Research*. Sage Publications, Beverly Hills, CA.

Rubin, D. B. (1981). Estimation in parallel randomized experiments. *Journal of Educational Statistics*, 6(4):377–401.

Silverman, B. W., Evers, L., Xu, K., Carbonetto, P., and Stephens, M. (2017). *EbayesThresh: Empirical Bayes Thresholding and Related Methods*. R package version 1.4-12.

Stein, C. (1956). Inadmissibility of the usual estimator for the mean of a multivariate normal distribution. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, 1954–1955, vol. I*, pages 197–206. University of California Press, Berkeley and Los Angeles.

Stephens, M. (2017). False discovery rates: a new deal. *Biostatistics*, 18(2):275–294.

Stephens, M., Carbonetto, P., Gerard, D., Lu, M., Sun, L., Willwerscheid, J., and Xiao, N. (2020). *ashr: Methods for Adaptive Shrinkage, using Empirical Bayes*. R package version 2.2-51.

Urbut, S. M., Wang, G., Carbonetto, P., and Stephens, M. (2019). Flexible statistical methods for estimating and testing effects in genomic studies with multiple conditions. *Nature Genetics*, 51(1):187–195.

van der Pas, S., Scott, J., Chakraborty, A., and Bhattacharya, A. (2019). *horseshoe: Implementation of the Horseshoe Prior*. R package version 0.2.0.

Wang, M., Fischer, J., and Song, Y. S. (2019). Three-way clustering of multi-tissue multi-individual gene expression data using semi-nonnegative tensor decomposition. *The annals of applied statistics*, 13(2):1103.

Wang, W. and Stephens, M. (2021). Empirical Bayes matrix factorization. *Journal of Machine Learning Research*, 22(120):1–40.

Willwerscheid, J. (2021a). *Empirical Bayes Matrix Factorization: Methods and Applications*. PhD thesis, University of Chicago, Chicago.

Willwerscheid, J. (2021b). *flashier: Empirical Bayes Matrix Factorization*. R package version 0.2.9.

6. Supplementary Benchmarking Results

6.1. Optimization Methods for Parametric Families

We first compare the performance of six optimization methods, all of which are implemented in **ebnm** via parameter `optmethod`. In each case, parameters are transformed so that the optimization problem is unconstrained (specifically, a log transformation is used for scale parameters, which are constrained to be nonnegative, while a logit transformation is used for mixture proportions, which are constrained to lie between zero and one).

Three choices of `optmethod` call into function `nlm`, a Newton-type algorithm included in the base **stats** package. Gradient and Hessian functions can be passed into `nlm`; if not, they are estimated numerically. Option `optmethod = "nlm"` provides both the gradient and Hessian functions; `optmethod = "nohess_nlm"` provides the gradient but not the Hessian; `optmethod = "nograd_nlm"` provides neither. Options `optmethod = "lbfgsb"` and `optmethod = "nograd_lbfgsb"` call into function `optim`, also in the **stats** package, with argument `method = "L-BFGS-B"`. The former provides the gradient function; the second does not. By definition, L-BFGS-B does not accept a Hessian. Finally, `optmethod = "trust"` calls into function `trust`, a trust-region algorithm implemented in package **trust** (Geyer 2020). Since `trust` requires both a gradient and Hessian function, there is only one corresponding `optmethod`. In sum, then, there are two methods that use both gradients and Hessians (`nlm` and `trust`); two that use only gradients (`nohess_nlm` and `lbfgsb`); and two that estimate all derivatives numerically (`nograd_nlm` and `nograd_lbfgsb`).
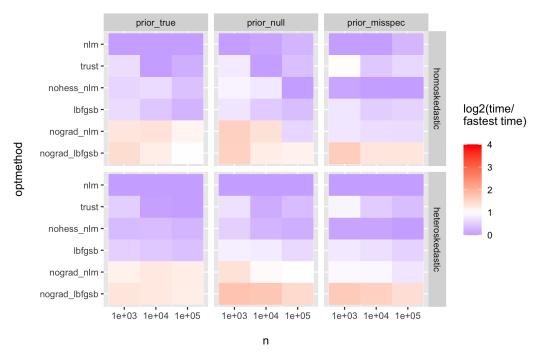
For both `ebnm_point_normal` and `ebnm_point_laplace`, we ran tests for $2 \times 3 \times 2 \times 3 = 36$ scenarios:

- The mode is either fixed at zero via argument `mode = 0` or estimated (via `mode = "estimate"`).

- The data-generating prior distribution g is: i) a true member of the prior family, $\pi_0 \delta_\mu + (1-\pi_0)N(\mu, a^2)$ or $\pi_0 \delta_\mu + (1-\pi_0)\text{Laplace}(\mu, a)$, with $\pi_0 \sim \text{Beta}(10, 2)$, $a \sim \text{Gamma}(4, 1)$, and either $\mu = 0$ or $\mu \sim \text{Unif}[-10, 10]$; ii) the null distribution δ_0; or iii) a distribution not in the prior family, so that $\mathcal{G}$ is misspecified. When `mode = 0`, the misspecified prior is a point-normal prior as above but with $\mu \sim \text{Unif}[-10, 10]$; when `mode = "estimate"`, the misspecified prior is the point-t_5 distribution $\pi_0 \delta_0 + (1 - \pi_0)at_5$, with π_0 and a distributed as above.

- The $N(0, s_i^2)$ noise added to the true means $\theta_i \sim g$ is either homoskedastic, with $s_i = 1$ for all i, or heteroskedastic, with $s_i^2 \sim \text{Exp}(1)$.

- The number of observations n is 1000, 10000, or 100000.

For each scenario, we ran $10^6/n$ simulations (so, depending on n, 1000, 100, or 10 simulations) and compared run-times using package **microbenchmark** (Mersmann 2019). All experiments were performed on a 2018 MacBook Pro with a 2.6 GHz Intel Core i7 processor and 16 GB of DDR4-2400 RAM. Results are displayed in Figures 5 and 6.

In general, the methods that supplied the gradient function outperformed the methods that required derivatives to be estimated numerically. Of the four methods that do supply the

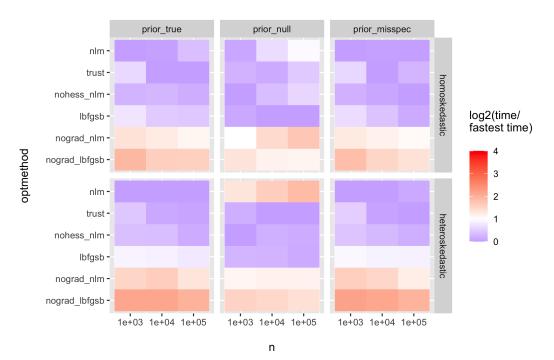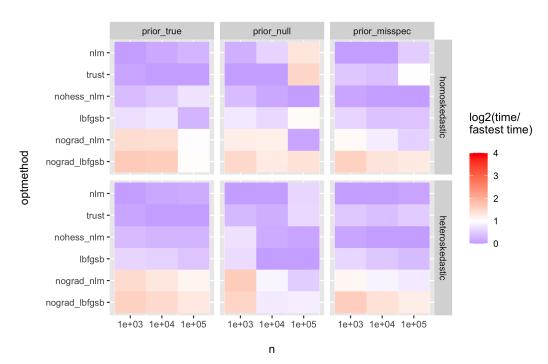Mode: fixed at zero

Mode: estimated

Figure 5: Timing comparisons for `ebnm_point_normal`. Tests in the top figure fix the mode at zero (by setting parameter `mode = 0`) while those in the bottom estimate the mode (by setting `mode = "estimate"`). Different columns correspond to different data-generating priors: a true member of the point-normal prior family; the null distribution δ_0; or a distribution from outside the prior family. Different rows correspond to different noise models, with the noise added to the "true" observations either homoskedastic (with $s_i = 1$ for all i) or heteroskedastic (with $s_i^2 \sim \text{Exp}(1)$). Plotted is the time as a multiple of the fastest time for a given combination of mode, prior, noise, and value of n. Shades of blue indicate optimization methods that are within a factor of 2 of the fastest method for that mode, prior, noise, and value of n, while shades of red indicate slower methods.
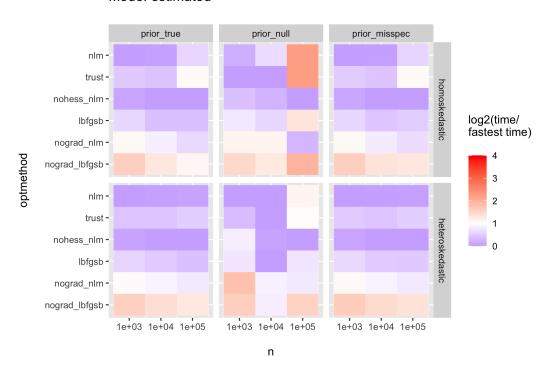
Mode: fixed at zero

Mode: estimated

Figure 6: Timing comparisons for `ebnm_point_laplace`. See Figure 5 caption and text for details.

gradient, exactly one, `nohess_nlm`, was "blue" across the board — that is, it was within a factor of 2 of the fastest method for every scenario. The methods that supply Hessians (`nlm` and `trust`) can struggle when the data-generating prior is the null distribution δ_0. Finally, method `lbfgsb` did nearly as well as `nohess_nlm` in terms of run-time but failed to converge in several of the `prior_null` simulations (in some scenarios, up to 10% of simulations resulted in an error). We thus feel confident in recommending the default setting `optmethod = "nohess_nlm"`.

6.2. Comparisons with Existing Packages

Next we compare the performance of **ebnm** against three packages with directly comparable functions: function `ebnm_point_laplace` is closely related to function `ebayesthresh` in package **EbayesThresh** (Silverman et al. 2017); function `ebnm_normal_scale_mixture` is modelled on function `ash` in package **ashr** (with option `mixcompdist = "normal"`; Stephens et al. (2020)) but, as mentioned above, is implemented in a much simpler manner; and function `ebnm_npmle` performs a similar task to function `GLmix` in package **REBayes** (Koenker and Gu 2017).

We ran tests for the same scenarios as the previous subsection, with the difference that the mode is always fixed at zero (mode estimation is not possible with **EbayesThresh**). Further, since it's not possible to "misspecify" the prior for the family of all distributions $\mathcal{G}_{\mathrm{npmle}}$, we only considered a single data-generating distribution (the point-Laplace), but we varied the number of grid points (mixture components) from 10 to 300. The number of simulations, `microbenchmark` settings, and hardware were as described in the previous section. We set parameters to make outputs as similar as possible. For **EbayesThresh**, we set `threshrule = "mean"` and `universalthresh = FALSE`; for `ash`, we set `prior = "uniform"`. Results are given in Figures 7-9.

In most scenarios, package **ebnm** outperformed both **EbayesThresh** and **ashr**. `ash` was usually slower than `ebnm_normal_scale_mixture` by a factor of around 2 to 4. In some scenarios, **EbayesThresh** was nearly as fast as `ebnm_point_laplace`, but in others it was outperformed by a full order of magnitude. Further, **ebnm** regularly found significantly better solutions than **EbayesThresh** (in terms of the final objective attained) except when the data-generating prior was the null distribution, in which cases the packages found solutions of similar quality.

Results in the comparison between `ebnm_npmle` and **REBayes** were mixed. **ebnm** was regularly faster when the number of mixture components was small (fewer than 100), while **REBayes** was consistently faster when a dense grid was used (more than 100 components). According to the theory developed in Willwerscheid (2021a), 100 components should be "good enough" for homoskedastic observations when

$$n^{1/4}\left(\frac{\mathrm{range}(x)}{s}\right) \le 400. \tag{35}$$

For example, if the number of observations $n = 10^6$, then 100 components should suffice as long as

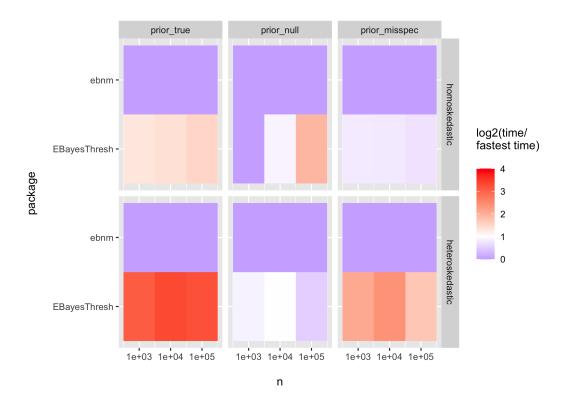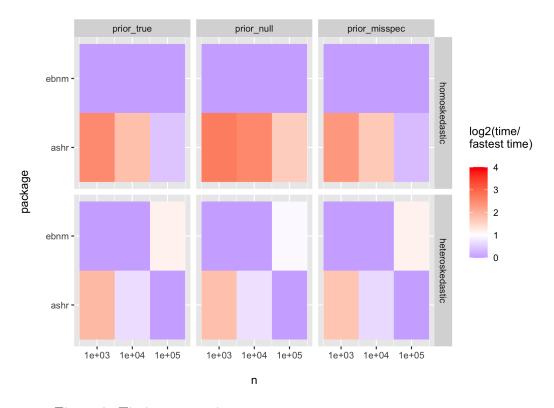$$\frac{\max(x) - \min(x)}{s} \le \frac{40}{\sqrt{10}}. \tag{36}$$

Figure 7: Timing comparisons: `ebnm_point_laplace` vs. `ebayesthresh`.

Figure 8: Timing comparisons: `ebnm_normal_scale_mixture` vs. `ash`.
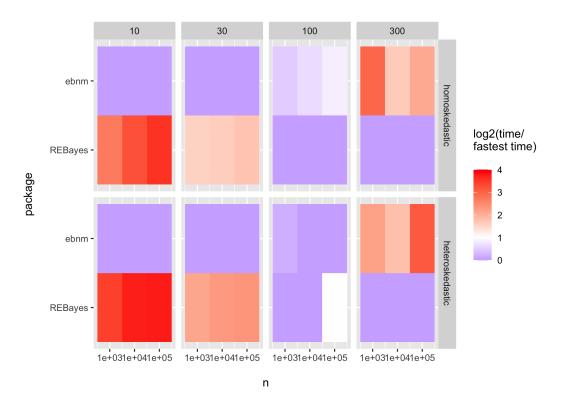
Figure 9: Timing comparisons: `ebnm_npmle` vs. **REBayes**.

Affiliation:

Jason Willwerscheid
Department of Statistics
University of Chicago
Chicago, Illinois, United States of America
E-mail: willwerscheid@uchicago.edu

Matthew Stephens
Department of Statistics
University of Chicago
Chicago, Illinois, United States of America
E-mail: mstephens@uchicago.edu