Journal of Statistical Software

September 2025, Volume 114, Issue 3. doi: 10.18637/jss.v114.i03

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

Jason Willwerscheid Providence College

Peter Carbonetto University of Chicago

Matthew Stephens University of Chicago

Abstract

The empirical Bayes normal means (EBNM) model is important to many areas of statistics, including (but not limited to) multiple testing, wavelet denoising, and gene expression analysis. There are several existing software packages that can fit EBNM models under different prior assumptions and using different algorithms. However, the differences across interfaces complicate direct comparisons, and a number of important prior assumptions do not yet have implementations. Motivated by these issues, we developed the R package **ebnm**, which provides a unified interface for efficiently fitting EBNM models using a variety of prior assumptions, including nonparametric approaches. In some cases, we incorporated existing implementations into **ebnm**; in others, we implemented new fitting procedures, with an emphasis on speed and numerical stability. We illustrate the use of **ebnm** in a detailed analysis of baseball statistics. By providing a unified and easily extensible interface, **ebnm** can facilitate development of new methods in statistics, genetics, and other areas; as an example, we briefly discuss the R package **flashier**, which harnesses **ebnm** for flexible and robust matrix factorization.

Keywords: empirical Bayes, normal means, shrinkage estimation, mixture models, NPMLE, maximum likelihood.

1. Introduction

Given n observations $x_i \in \mathbb{R}$ with known standard deviations $s_i > 0$, $i = 1, \ldots, n$, the normal means model (Robbins 1951; Efron and Morris 1972; Stephens 2017; Bhadra, Datta, Polson, and Willard 2019; Johnstone 2019; Sun 2020) is

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

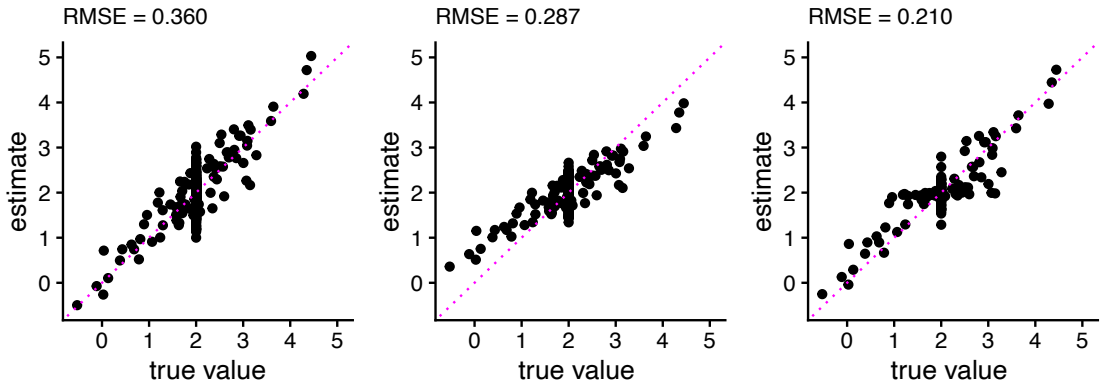

Figure 1: Illustration of shrinkage estimation using empirical Bayes normal means (EBNM). The example data set consists of 400 noisy observations x_i, with true means θ_i simulated from a "point-normal" prior. The left-hand plot shows the observed means (x_i) vs. the true means (θ_i). (The observed means are also the maximum-likelihood estimates (MLEs) of the true means.) The middle plot shows EB estimates of the true means obtained by learning a normal prior g from the data. These EB estimates "shrink" the observations toward the mode of the common prior, improving the overall root mean-squared error (RMSE) from 0.360 to 0.287. But the normal prior also appears to "overshrink" observations distant from the center (which is located near 2). Fitting an EBNM model with a family of priors that better suits the data – point-normal priors – results in more accurate EB estimates, improving the RMSE to 0.210 (see the right-hand plot). In particular, the point-normal prior avoids "overshrinking" the more extreme observations. For an expanded illustration, including the R code that generates the results shown here, see the **ebnm** package vignette, "Introduction to the empirical Bayes normal means model via shrinkage estimation."

where the unknown ("true") means $\theta_i \in \mathbb{R}$, $i = 1, \ldots, n$, are the quantities to be estimated. Here and throughout, $\mathcal{N}(\mu, \sigma^2)$ denotes the normal distribution with mean μ and variance σ^2.

The empirical Bayes (EB) approach to inferring θ_i attempts to improve upon the maximum-likelihood estimates $\hat{\theta}_i = x_i$ 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; Stephens 2017). Specifically, the empirical Bayes normal means (EBNM) approach assumes

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

where $\mathcal{G}$ is some family of probability distributions that is specified in advance, and $g \in \mathcal{G}$ is estimated using the data (typically, via maximum likelihood). Given an estimate of the prior, $\hat{g} \in \mathcal{G}$, estimates of θ_i can be obtained using, for example, posterior means:

$$\hat{\theta}_i := \mathsf{E}\left(\theta_i \mid x_i, \hat{g}\right).$$

We refer to the problem of computing $\hat{g}$ and $\hat{\theta}_i$ as "solving the empirical Bayes normal means problem." See Figure 1 for an illustration.

Applications in which the EBNM model plays an important role include wavelet denoising (Clyde and George 2000; Johnstone and Silverman 2004, 2005b), multiple testing (Efron 2010;

Stephens 2017), gene expression analysis (Love, Huber, and Anders 2014; Zhu, Ibrahim, and Love 2019; Smyth 2004), multiple linear regression (Kim, Wang, Carbonetto, and Stephens 2024; Mukherjee, Sen, and Sen 2023), and matrix factorization (Wang and Stephens 2021). This versatility has motivated the development of a number of software packages using different choices for the prior family $\mathcal{G}$; see Section 2 for a review. Still, important gaps in the software remain. For example, we are not aware of any software that fits the EBNM model in the simple case where $\mathcal{G}$ is the family of univariate normal distributions. Further, each existing package has a different interface and outputs, which complicates comparisons across packages as well as making it difficult to develop software packages that flexibly build on EBNM methods. Motivated by these issues, we developed **ebnm**, which provides a unified interface for efficiently solving the EBNM problem using a wide variety of prior families.

We developed the **ebnm** package in R (R Core Team 2025), a programming language and environment that is free, open source, and highly interoperable – for example, with Python (Van Rossum *et al.* 2011) via **rpy2** (Gautier 2023), with MATLAB (The MathWorks Inc. 2021) via **R-link** (Henson 2024), and with Julia (Bezanson, Edelman, Karpinski, and Shah 2017) via **RCall** (Bates, Lai, and Byrne 2024). **ebnm** can be downloaded from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=ebnm`. The latest development branch is available on GitHub (`https://github.com/stephenslab/ebnm`). The website, which includes detailed documentation and vignettes, is at `https://stephenslab.github.io/ebnm/`. Code for reproducing results and figures in the text is available on GitHub at `https://github.com/willwerscheid/ebnm-paper` and as supplementary materials to this paper.

The paper is organized as follows. In Section 2, we give a brief history of the EBNM problem and review existing approaches. Section 3 gives an overview of the **ebnm** package, including the unified interface and the newly implemented prior families. In Section 4, we compare different choices of prior family and illustrate how this choice can impact statistical performance. This section also includes a runtime benchmark for several implemented prior families. Section 5 illustrates usage of **ebnm** in an analysis of baseball statistics. In Section 6, we describe the matrix factorization framework implemented in the R package **flashier** (Willwerscheid, Carbonetto, Wang, and Stephens 2024), which builds on **ebnm**. In **flashier**, the EBNM problem arises as a subproblem that must be solved many times, so that the speed and flexibility provided by **ebnm** prove critical. Finally, Section 7 summarizes the key contributions of this work.

2. Background and existing software

In this section, we review existing approaches to the EBNM problem within a common modeling framework.

2.1. Normal priors

Stein (1956) famously discovered that under quadratic loss, the maximum-likelihood estimate (MLE) $\hat{\theta}_i = x_i$, $i = 1, \ldots, n$, is an inadmissible solution to the homoskedastic normal means problem,

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

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

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

where $\mathcal{G}$ is the family of zero-mean normal distributions,

$$\mathcal{G}_{\text{norm0}} := \{g : g(x) = \mathcal{N}(x; 0, \sigma^2),\ \sigma^2 \geq 0\}.$$

(Here, $\mathcal{N}(x; \mu, \sigma^2)$ denotes the normal probability density function at x with mean μ and variance σ^2.)

In many applications, the mean of the θ_i's may be non-zero, and so a natural generalization is the family of *all* normal distributions,

$$\mathcal{G}_{\text{norm}} := \{g : g(x) = \mathcal{N}(x; \mu, \sigma^2),\ \sigma^2 \geq 0, \mu \in \mathbb{R}\}.$$

Estimating $g \in \mathcal{G}_{\text{norm}}$ reduces to estimating σ^2 and μ. For the "homoskedastic" case – that is, when $s_i^2 = s^2$ for $i = 1, \ldots, n$ – the MLEs have simple, closed-form solutions:

$$\begin{aligned} \hat{\mu} &= \frac{1}{n}\sum_{i=1}^{n} x_i, \\ \hat{\sigma}^2 &= \max\left\{0,\ \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})^2 - s^2\right\}. \end{aligned} \tag{3}$$

When μ is fixed at zero, the solution in Equation 3 is similar to the one implied by the "positive-part James-Stein estimator," with the sole difference that it divides $\sum_{i=1}^n x_i^2$ by n rather than by $n - 2$ (Efron and Morris 1973). For the "heteroskedastic" case, when the σ_i^2 are not all identical, the likelihood for μ, σ^2 has a closed form but must be maximized numerically.

In both the homoskedastic and heteroskedastic cases, the posterior distributions

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

are normal distributions with closed-form expressions.

2.2. Sparse priors

Although the normal prior family has the advantage of simplicity, in practice more flexible priors are often preferred. In particular, one would often like the prior to be able to capture sparsity in $\boldsymbol{\theta} := \{\theta_1, \ldots, \theta_n\}$. One approach is to use a "spike-and-slab" prior; that is, a mixture consisting of two components, a point-mass at zero (the "spike") and a "slab" belonging to a family of continuous distributions, usually symmetric and centered at zero. A common choice is the "point-normal" family,

$$\mathcal{G}_{\text{pn}} := \{g : g(x) = \pi_0 \delta_0(x) + (1 - \pi_0)\mathcal{N}(x; 0, \sigma^2),\ 0 \leq \pi_0 \leq 1,\ \sigma^2 > 0\},$$

where $\delta_y(x)$ denotes the density function of the delta-Dirac mass centered at y. With this choice, estimating g reduces to estimating two parameters, π_0 and σ^2. Similar to the family of normal distributions, the likelihood for the point-normal prior family has a closed form, and

standard numerical optimization methods can be used to efficiently find the MLE $\hat{g} \in \mathcal{G}_{\text{pn}}$. Given $\hat{g}$, posterior distributions of θ_i are mixtures of a point-mass at zero and a normal distribution, and have closed-form expressions.

As Johnstone and Silverman (2005b) showed, replacing the normal slab with a "heavy-tailed" distribution generally improves accuracy. Their **EbayesThresh** software, available in R and S-PLUS (Johnstone and Silverman 2005a), implements two such priors: the point-Laplace prior,

$$\mathcal{G}_{\text{pl}} := \{g : g(x) = \pi_0\delta_0(x) + (1-\pi_0)\text{Laplace}(x; 0, a),\ 0 \le \pi_0 \le 1,\ a \ge 0\}, \tag{5}$$

and a family of priors in which the slab has "Cauchy-like" tails. For both priors, MLEs $\hat{g} \in \mathcal{G}$ can be found using numerical methods. In Equation 5, $\text{Laplace}(x; \mu, a)$ denotes the probability density of the Laplace distribution at x with mean μ and scale a (Gelman, Carlin, Stern, Dunson, Vehtari, and Rubin 2014).

Another parametric prior that is well suited for capturing sparse signals is the horseshoe prior, which models sparsity by having appreciable mass near zero rather than exactly at zero (Carvalho, Polson, and Scott 2010). The R package **horseshoe** (Van der Pas, Scott, Chakraborty, and Bhattacharya 2019) solves the homoskedastic EBNM problem with $\mathcal{G}$ the family of horseshoe distributions. See Bhadra *et al.* (2019) for a review of this and other software implementations of the horseshoe prior.

2.3. Nonparametric approaches

When $\mathcal{G}$ is the unconstrained family of all distributions, $\hat{g} \in \mathcal{G}$ is called the nonparametric maximum-likelihood estimate (NPMLE) (Kiefer and Wolfowitz 1956; Laird 1978; Lindsay 1983; Jiang and Zhang 2009; Koenker and Mizera 2014; Dicker and Zhao 2016). In practice, most nonparametric methods approximate this family, which we denote as $\mathcal{G}_{\text{npmle}}$, by a finite mixture of point masses,

$$\tilde{\mathcal{G}}_{\text{npmle}} := \left\{ g : g(x) = \sum_{k=1}^{K} \pi_k \delta_{\mu_k}(x) \,\middle|\, \pi_1, \ldots, \pi_K \ge 0,\ \sum_{k=1}^{K} \pi_k = 1 \right\},$$

where $\mu_1, \ldots, \mu_K$ is a fixed, dense grid of values spanning the range of the observations. Estimating $g \in \tilde{\mathcal{G}}_{\text{npmle}}$ involves solving a convex optimization problem,

$$\begin{aligned} \text{maximize} \quad & \mathbf{L}\boldsymbol{\pi} \\ \text{subject to} \quad & \mathbf{0} \preceq \boldsymbol{\pi} \preceq \mathbf{1}_K \\ & \boldsymbol{\pi}^\top \mathbf{1}_K = 1, \end{aligned} \tag{6}$$

where $\boldsymbol{\pi} := (\pi_1, \ldots, \pi_K)^\top$, $\mathbf{1}_K$ is a column vector of ones of length K, and $\mathbf{L} \in \mathbb{R}^{n \times K}$ is the matrix with elements $\ell_{ik} = \mathcal{N}(x_i; \mu_k, \sigma_i^2)$ (Koenker and Mizera 2014). The R package **REBayes** (Koenker and Gu 2017) implements an efficient solver for Equation 6 using interior point methods (MOSEK ApS 2019). See Kim, Carbonetto, Stephens, and Anitescu (2020) and Zhang, Cui, Sen, and Toh (2024) for other approaches.

Although the fully nonparametric approach is the most flexible, the NPMLE is a discrete distribution (Laird 1978), so posterior distributions in Equation 4 are discrete as well. For point estimation, the posterior mean from a discrete prior is usually adequate; however, interval estimates can behave poorly. To address this issue, the R package **deconvolveR**

(Narasimhan and Efron 2020) uses a natural spline basis to obtain a smoothed nonparametric estimate of g. This approach provides sensible interval estimates and can in certain respects outperform the NPMLE when the true prior is smooth (Koenker 2017).

2.4. Constrained nonparametric approaches

Constrained nonparametric approaches offer a middle ground, retaining some of the flexibility of a fully nonparametric approach while avoiding "overfitting" (Hastie, Tibshirani, and Friedman 2009). Stephens (2017) argued that the set of all distributions that are unimodal at zero can be a particularly good choice of $\mathcal{G}$ in the context of multiple testing. If it is reasonable to assume that the prior is symmetric, one can instead take $\mathcal{G}$ to be the family of scale mixtures of normals,

$$\mathcal{G}_{\text{smn}} := \{g : g(x) = \int_0^\infty \mathcal{N}(x; 0, \sigma^2)\, dh(\sigma^2) \text{ for some } h\}.$$

A slightly more flexible option is the family of all symmetric distributions that are unimodal at zero, which can be represented by scale mixtures of uniform distributions:

$$\mathcal{G}_{\text{symm-u}} := \{g : g(x) = \int_0^\infty \text{Unif}(x; -a, a)\, dh\,(a) \text{ for some } h\}.$$

(Here, $\text{Unif}(x; a, b)$ denotes the probability density function of the uniform distribution on (a, b).)

When these families are approximated by finite mixtures, estimating g reduces to the same convex optimization problem that arises for the NPMLE (Equation 6), and can again be solved using fast algorithms for convex optimization. This approach is implemented in the R package **ashr** (Stephens *et al.* 2023), which, by default, uses **mixsqp** (Kim *et al.* 2020) to solve the problem in Equation 6.

3. The ebnm package: implementation and usage

The **ebnm** package provides a unified interface for solving the EBNM problem using a wide variety of prior assumptions, including all of the choices of prior family discussed in Section 2. In addition to making existing implementations available via a shared interface, the package provides new implementations for several simple but useful prior families that, to our knowledge, have not previously been implemented; important examples include the normal and point-normal prior families ($\mathcal{G}_{\text{norm}}$ and $\mathcal{G}_{\text{pn}}$). The available prior families are summarized in Table 1. Note, also, that **ebnm** was designed to be easily extensible to other prior families. To facilitate such extensions, we have written a vignette, "Extending ebnm with custom ebnm-style functions," which is available on the **ebnm** package website.

3.1. The `ebnm()` function

The `ebnm()` function is the main interface to the EBNM methods. It has the following input arguments:

- `x`: The vector of observations, $\boldsymbol{x} = \{x_1, \ldots, x_n\}$.
- `s`: The vector of standard errors, $\boldsymbol{s} = \{s_1, \ldots, s_n\}$. (`s` may be a scalar for the homoskedastic case.)

- `prior_family`: The prior family, $\mathcal{G}$ (see Table 1).
- `mode`: For unimodal prior families, this argument specifies the mode. The mode may also be estimated with `mode = "estimate"`.
- `scale`: Either the scale parameter (for parametric priors), or the grid of parameters used to approximate the nonparametric prior. By default it is `scale = "estimate"`, which directs **ebnm** either to estimate the scale or to select the grid following the grid selection strategies described in Willwerscheid (2021).
- `g_init`: An optional initial estimate of g. A good estimate can speed up the search for an MLE.
- `fix_g`: A logical argument. When `fix_g = TRUE`, the prior is fixed to `g_init` so that the posterior distributions are computed using this fixed value of g.
- `output`: A character vector indicating which quantities should be returned.
- `optmethod`: The name of the optimization method to use. See Section 3.2.
- `control`: A list of parameters controlling the optimization.

The `ebnm()` outputs include:

- `fitted_g`: The estimated prior, $\hat{g}$.
- `log_likelihood`: The log-likelihood at $\hat{g}$, which can be used to compare the quality of the fit to the data across different priors or prior families:

$$\log p(x_1, \ldots, x_n \mid s_1, \ldots, s_n, \hat{g}) = \sum_{i=1}^{n} \log \int p(x_i \mid \theta_i, s_i)\, \hat{g}(\theta_i)\, d\theta_i. \tag{7}$$

- `posterior`: Summaries of the posterior distributions $p(\theta_i \mid x_i, s_i, \hat{g})$, including posterior means, posterior standard deviations, and *local false sign rates* (Stephens 2017), defined as

$$\mathit{lfsr}(i) := \min\left\{p(\theta_i \leq 0 \mid x_i, s_i, \hat{g}),\, p(\theta_i \geq 0 \mid x_i, s_i, \hat{g})\right\}.$$

- `posterior_sampler`: A function that can be used to generate random samples from the posterior distributions in Equation 4.

The return value is an object of class '`ebnm`'. Many of the S3 methods that are typically associated with model fits in R also work for objects of class '`ebnm`', including:

- `summary()`: Gives an overview of the fitted model.
- `plot()`: Produces a scatterplot comparing the observations x_i against posterior estimates of the true means θ_i and, optionally, a visualization of the prior cumulative density function.
- `nobs()`: Returns the number of observations used to fit the model.
- `coef()`: Returns the posterior means, $\hat{\theta}_i := \mathsf{E}(\theta_i \mid x_i, s_i, \hat{g})$.

- `vcov()`: Returns the posterior variances, $\mathsf{VAR}(\theta_i \mid x_i, s_i, \hat{g})$.
- `fitted()`: Returns a data frame containing multiple posterior summary statistics (e.g., posterior means and variances).
- `residuals()`: Returns the residuals, defined as the differences $x_i - \hat{\theta}_i$.
- `logLik()`: Returns the log-likelihood in Equation 7 at $\hat{g}$.
- `simulate()`: Generates random draws of each θ_i from their posterior distributions.
- `quantile()`: Uses the sampler to compute posterior quantiles for each θ_i.
- `confint()`: Uses the sampler to compute posterior "credible intervals" for each θ_i. We define the $(1-\alpha)\%$ credible interval for θ_i as the narrowest continuous interval $[a_i, b_i]$ such that $\theta_i \in [a_i, b_i]$ with posterior probability at least $1-\alpha$, $\alpha \in (0, 1)$. The credible intervals are estimated using Monte Carlo methods. The proportion $1-\alpha$ is determined by the `level` argument.
- `predict()`: Uses the fitted prior $\hat{g}$ to compute posterior mean estimates $\hat{\theta}_i^{\text{new}}$ for a different set of observations x_i^{new} with standard deviations s_i^{new}. This could be used, for example, to provide a more robust assessment of fit by evaluating the accuracy of the predictions on a test set.

We illustrate several of these methods in Section 5 and in the package vignettes.

3.2. Optimization methods

The `ebnm()` function involves two key computations:

1. *Estimate the prior.* Compute $\hat{g} := \text{argmax}_{g \in \mathcal{G}} L(g)$, where $L(g)$ denotes the marginal likelihood,
$$L(g) := p(\boldsymbol{x} \mid g, \boldsymbol{s}) = \prod_{i=1}^{n} \int p(x_i \mid \theta_i, s_i)\, g(\theta_i)\, d\theta_i,$$
where $\boldsymbol{x} := (x_1, \ldots, x_n)$ and $\boldsymbol{s} := (s_1, \ldots, s_n)$.
2. *Compute posterior quantities.* Compute summaries (means, variances, etc.) from the posterior distributions
$$p(\theta_i \mid x_i, s_i, \hat{g}) \propto p(x_i \mid \theta_i, s_i)\, \hat{g}(\theta_i).$$

The complexity of each step depends on the prior family $\mathcal{G}$. In most cases, estimating the prior (Step 1) is the most difficult and computationally intensive step; for all but the simplest prior families this step requires the use of numerical optimization methods.

Parametric priors

Parametric priors available in **ebnm** include the normal, point-normal, point-Laplace, point-exponential, and horseshoe prior families. For the horseshoe, we used the implementation from the **horseshoe** package (Van der Pas *et al.* 2019), which requires the prior mode to be

prior_family	Prior	Source	Support	Sym?
Parametric				
"normal"	$\mathcal{N}(x; \mu, \sigma^2)$	**ebnm**	$\pm$	Yes
"point_mass"	$\delta_\mu(x)$	**ebnm**		Yes
"point_normal"	$\pi_0\delta_\mu(x) + (1-\pi_0)\mathcal{N}(x; \mu, \sigma^2)$	**ebnm**	$\pm$	Yes
"point_laplace"	$\pi_0\delta_\mu(x) + (1-\pi_0)\text{Laplace}(x; \mu, a)$	**ebnm**	$\pm$	Yes
"point_exponential"	$\pi_0\delta_0(x) + (1-\pi_0)\text{Exp}(x; a)$	**ebnm**	$+$	No
"horseshoe"	$\text{Horseshoe}(x; \tau)$	**horseshoe**	$\pm$	Yes
Constrained nonparametric				
"normal_scale_mixture"	$\int_0^\infty \mathcal{N}(x; 0, \sigma^2)\, dh(\sigma^2)$	**ebnm**	$\pm$	Yes
"unimodal_symmetric"	$\int_0^\infty \text{Unif}(x; -a, a)\, dh(a)$	**ashr**	$\pm$	Yes
"unimodal"	$\int_{-\infty}^\infty \text{Unif}(x; 0, a)\, dh(a)$	**ashr**	$\pm$	No
"unimodal_nonnegative"	$\int_0^\infty \text{Unif}(x; 0, a)\, dh(a)$	**ashr**	$+$	No
Nonparametric				
"npmle"	$\int_{-\infty}^\infty \delta_t(x)\, dh(t)$	**ebnm**	$\pm$	No
"deconvolver"	See Narasimhan and Efron (2020)	**deconvolveR**	$\pm$	No
Other				
"flat"	$\text{Unif}(x; -\infty, \infty)$	**ebnm**	$\pm$	Yes

Table 1: Prior families implemented in **ebnm**. The "prior_family" column gives the corresponding prior_family argument to ebnm(). The "source" column gives the name of the R package that implements the core model-fitting routines. "Support" says whether the prior has support for only positive realizations of θ_i ($+$) or for all real numbers ($\pm$). A "yes" in the "sym?" column means that the prior is symmetric about its mode. Note that the "flat" prior is mainly intended for use as a point of comparison with other prior families; it always only returns $\hat{\theta}_i = x_i$. Also note that some specialized priors such as the "generalized binary prior" (Liu *et al.* 2025) are not included in this table; run help("ebnm") to obtain a full list of supported prior families.

fixed at zero. For all other prior families, we developed special implementations, and we allow the prior mode to be estimated or fixed at a specified value (commonly, zero).

A closed-form solution is available only for the normal prior with homoskedastic errors. In all other cases, we use numerical methods to search for an MLE. This search involves at most three parameters: the scale of the slab component, the mixture weight for the spike, and the mode (when mode = "estimate").

We found that several off-the-shelf optimizers worked well for fitting parametric priors, although care was needed in implementing the underlying objective and gradient computations in order to avoid numerical issues. In particular, we found that the quasi-Newton method nlm() from the **stats** package worked reliably in our tests across a range of parametric prior

families. Therefore, this method is the default for estimating the prior in all cases except the horseshoe, which uses `optimize()`.

Since other optimization methods might be preferred in some circumstances – say, when dealing with large or complex data sets, or to refine the estimation of the prior – we have designed the package to allow use of other optimization methods. Further, the user can choose whether to use analytical gradients and Hessians, or whether to estimate them numerically (which can be faster when the analytical calculations are complex). These options are controlled by the `optmethod` argument to `ebnm()`. The default for most parametric priors, `"nohess_nlm"`, uses `nlm()` with gradients calculated analytically and Hessians estimated numerically. Alternatives include `"nlm"` (both gradients and Hessians are calculated analytically); `"nograd_nlm"` (both gradients and Hessians are estimated numerically); `"lbfgsb"` and `"nograd_lbfgsb"`, which use the L-BFGS-B algorithm from `optim()` (L-BFGS-B always estimates Hessians numerically, so the two options use, respectively, analytical and numerical gradients); and the trust region method from the **trust** package (Geyer 2020), which requires analytical gradients and Hessians (`optmethod = "trust"`).

In our benchmarking experiments (Appendix A.1), `"nohess_nlm"` was always either the fastest method or took no more than twice as long as the fastest method. Both the other `nlm()` methods and the `trust()` method reliably converged to a solution, but they tended to be slower. L-BFGS-B did not reliably find a solution.

Constrained nonparametric priors

The constrained nonparametric families (Table 1) are implemented in package **ashr** (Stephens *et al.* 2023). **ashr** uses the mix-SQP algorithm (Kim *et al.* 2020) by default. Different optimization methods can again be chosen via the `optmethod` argument; for details on these different methods, see the **ashr** documentation. The only constrained nonparametric family that does not rely on **ashr** is the family of scale mixtures of normals. For this family, we re-implemented the **ashr** algorithm with the aim of improving speed. Our implementation improved the average runtime over **ashr** by a full order of magnitude for smaller data sets ($n \approx 1\,000$) (Appendix A.2).

Nonparametric priors

The NPMLE can, in principle, be computed using **ashr**, but this computation is cumbersome since **ashr** requires the user to specify the grid of point masses in advance. Further, we have found that, as with scale mixtures of normals, **ashr** can be slow in some scenarios. The **REBayes** package (Koenker and Gu 2017) was developed specifically for the NPMLE, and is typically very fast, but it relies on the commercial interior-point solver MOSEK (MOSEK ApS 2019). Therefore, in order to provide a fully open-source toolkit that does not require installation of commercial software, we re-implemented the NPMLE in **ebnm** using the open-source package **mixsqp** (Kim *et al.* 2020). As with the constrained nonparametric prior families, `optmethod = "mixsqp"` is the default setting. If desired, however, the **REBayes** algorithm can be used by setting `optmethod = "REBayes"`. In our tests, **mixsqp** was typically faster than **REBayes** when the number of mixture components K was small, whereas **REBayes** was usually faster than **mixsqp** when K was 80 or more (Appendix A.2).

	Relative log-likelihood			RMSE			CI coverage		
Prior family	Normal	Point-t	Tophat	Normal	Point-t	Tophat	Normal	Point-t	Tophat
flat	NA	NA	NA	1.01	1.00	0.999	0.893	0.893	0.896
normal	−4.9	−62.5	−232.9	0.900	0.663	0.978	0.893	0.923	0.897
point_normal	−4.8	−7.1	−232.9	0.900	0.531	0.978	0.892	0.871	0.895
point_laplace	−26.5	−5.5	−310.5	0.917	0.527	0.988	0.881	0.909	0.893
normal_scale_mixture	−5.3	−4.5	−239.6	0.901	0.527	0.979	0.891	0.908	0.894
unimodal_symmetric	−6.1	−4.1	−198.2	0.908	0.528	0.966	0.861	0.875	0.890
unimodal	−4.2	−1.7	−11.7	0.909	0.529	0.941	0.860	0.877	0.890
npmle	0.0	0.0	0.0	0.908	0.534	0.951	0.687	0.718	0.630
deconvolver	−3.8	−18.5	−19.9	0.903	0.582	0.953	0.848	0.898	0.860
horseshoe	−161.0	−17.1	−931.8	0.987	0.532	1.10	0.812	0.903	0.826

Figure 2: Results of fitting EBNM models with different prior families. For each prior family, the table reports evaluation metrics averaged over 10 simulated data sets with $n = 1\,000$ observations per data set. Three metrics are reported: the log-likelihood at $\hat{g}$ relative to the log-likelihood attained by the NPMLE (higher log-likelihoods are better); the root mean-squared error (lower RMSEs are better); and the proportion of 90% posterior credible intervals containing the true mean (values closer to 0.9 are better).

4. Numerical comparisons of prior families

To test our implementations and to compare the performance of different prior families, we simulated data sets from three data-generating distributions:

1. **Normal**. The simplest scenario. We simulated the means θ_i from a normal prior, $\theta_i \overset{\text{ind.}}{\sim} \mathcal{N}(0, 2^2)$.

2. **Point-*t***. A more challenging scenario. The prior was both sparse and heavy-tailed, yet still symmetric: $\theta_i \overset{\text{ind.}}{\sim} 0.8\delta_0 + 0.2t_5(0, 1.5)$, where $t_\nu(\mu, \sigma)$ denotes the Student t distribution with location μ, scale σ, and ν degrees of freedom.

3. **Asymmetric Tophat**. The most challenging scenario. Means were uniformly distributed, $\theta_i \overset{\text{ind.}}{\sim} \text{Unif}(-5, 10)$. Although this scenario is perhaps less realistic than the other simulations, it yields data sets that are best modeled with nonparametric or constrained nonparametric priors.

In all simulations, we generated the observed means as $x_i \overset{\text{ind.}}{\sim} \mathcal{N}(\theta_i, 1)$.

Figure 2 summarizes results from running EBNM analyses on 10 data sets in each of the three simulation scenarios, with $n = 1\,000$ observations in each data set. We used the following three measures to evaluate the EBNM model fits:

a. The log-likelihood, which, for ease of interpretation, is shown relative to the log-likelihood attained at the NPMLE estimate. (In principle, the NPMLE estimate should always give the highest likelihood because it includes all other prior families as proper subsets.)

b. The root mean-squared error, RMSE $:= \sqrt{\sum_{i=1}^{n}(\hat{\theta}_i - \theta_i)^2/n}$, where $\hat{\theta}_i$ denotes the posterior mean estimate, $\hat{\theta}_i := \mathsf{E}(\theta_i \mid x_i, s_i, \hat{g})$.

c. The proportion of true means θ_i contained within the 90% posterior credible intervals.

As expected, the model fit returned by `ebnm()` with `prior_family = "npmle"` always attained the largest log-likelihood. More generally, log-likelihoods usually (though not always) aligned with the orderings implied by nestings of prior families, such as, for example,

$$\mathcal{G}_{\text{norm0}} \subset \mathcal{G}_{\text{pn}} \subset \mathcal{G}_{\text{smn}} \subset \mathcal{G}_{\text{symm-u}} \subset \mathcal{G}_{\text{npmle}}.$$

Prior families that were a poor match with the distribution used to simulate the data typically had worse log-likelihoods.

The RMSE evaluated the quality of the posterior estimates $\hat{\theta}_i$ generated by an EBNM analysis. Reassuringly, nearly all prior families improved upon the $\hat{\theta}_i = x_i$ estimates returned by the "flat" prior, which we included as a baseline. However, the improvement was sometimes small, particularly when the prior family was a poor match with the true distribution (e.g., symmetric prior families in the Asymmetric Tophat setting). In general, higher log-likelihoods were indicative of better accuracy of the estimates $\hat{\theta}_i$. Results that did not follow this trend suggested overfitting; for example, the RMSE for the NPMLE was typically worse than for prior families that better matched the true distribution.

The "CI coverage" measured how well posterior credible intervals are calibrated. A known limitation of empirical Bayes methods is that they often underestimate uncertainty in the posteriors, since uncertainty in the estimate of g is not taken into account (Ignatiadis and Wager 2022). Indeed, the credible intervals tended to be too small (less than 90%) for most prior families and simulation scenarios. Still, the intervals were usually not far from the target coverage of 90%. The exception was the NPMLE, which tended to have much poorer coverage because it resulted in a discrete prior that often greatly underestimated uncertainty.

Finally, to assess the ability of our implementation to handle large data sets, we simulated additional (point-t) data sets ranging in size from $n = 100$ to $n = 1\,000\,000$. These analyses were performed in R 4.3.2 on a desktop running Windows 11 Pro with an Intel Core i9-13900KF multicore processor and 96 GB of memory. As expected, the less flexible priors with the fewest parameters tended to also be the fastest, whereas the more complex methods (e.g., unimodal prior, NPMLE) were slower than the fastest methods by several orders of magnitude (see Figure 3). Importantly, all prior families scaled well to large data sets; for each, the computational effort grew linearly or close to linearly in n.

5. An analysis of weighted on-base averages with ebnm

In this section, we illustrate the key features of **ebnm** in an analysis of baseball statistics. See the package vignette for an expanded version of this example.

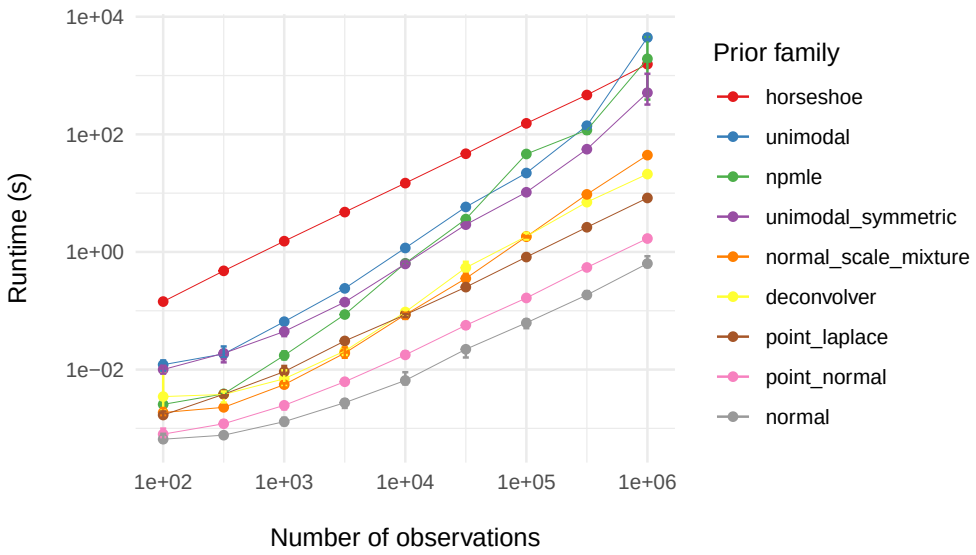

Figure 3: Runtimes for fitting EBNM models to data sets of varying sizes. For each combination of sample size (n) and prior family, an EBNM model was fit to 20 data sets simulated using the "Point-t" distribution. Each point gives the average of the 20 simulations; error bars show 10% and 90% quantiles.

5.1. The `wOBA` data set

We begin by loading and inspecting the `wOBA` data set, which consists of weighted on-base averages (wOBAs) and standard errors for the 2022 MLB regular season:

```
R> library("ebnm")
R> data("wOBA")
R> nrow(wOBA)

[1] 688

R> head(wOBA)

  FanGraphsID           Name Team  PA     x     s
1       19952     Khalil Lee  NYM   2 1.036 0.733
2       16953 Chadwick Tromp  ATL   4 0.852 0.258
3       19608     Otto Lopez  TOR  10 0.599 0.162
4       24770   James Outman  LAD  16 0.584 0.151
5        8090 Matt Carpenter  NYY 154 0.472 0.054
6       15640    Aaron Judge  NYY 696 0.458 0.024
```

Column `x` contains the (observed) wOBAs, which we interpret as estimates of a player's *true hitting ability*. Column `s` gives standard errors. See Appendix B for background on the wOBA statistic and an explanation of how standard errors were calculated.

Most players finished the season with a wOBA between 0.200 and 0.400. A few had very high wOBAs (>.500) while others had wOBAs at or near zero. A casual inspection of the

data suggests that players with these extreme wOBAs were very lucky (or very unlucky). For example, the 4 players with the highest wOBAs (included in the code output above) each had fewer than 20 plate appearances. (The number of plate appearances, or PAs, is the sample size, so smaller PAs generally lead to larger standard errors.)

In contrast, Aaron Judge's production – which included a record-breaking number of home runs – appears to be "real," since it was sustained over nearly 700 PAs. Other cases are more ambiguous; for example, Matt Carpenter had several exceptional seasons between 2013 and 2018, but then his performance declined in the 2019–2021 seasons. So what should we make of his remarkable improvement in 2022? An empirical Bayes analysis can help resolve some of these ambiguities.

5.2. The `ebnm()` function

Function `ebnm()` is the main interface for fitting the empirical Bayes normal means model given in Equations 1–2; it is a "Swiss army knife" that allows for various choices of prior family $\mathcal{G}$ as well as many other options for fitting and tuning models. For example, we might take $\mathcal{G}$ to be the family of normal distributions:

```
R> x <- wOBA$x
R> s <- wOBA$s
R> names(x) <- wOBA$Name
R> names(s) <- wOBA$Name
R> fit_normal <- ebnm(x, s, prior_family = "normal", mode = "estimate")
```

(Since the distribution of true hitting ability should not be centered at zero, we set `mode = "estimate"`.)

The **ebnm** package has an additional, alternative interface in which each prior family has its own function. The following call produces the same result as the previous call to `ebnm()`:

```
R> fit_normal <- ebnm_normal(x, s, mode = "estimate")
```

Overviews of results can be obtained using the `summary()` and `plot()` methods. The `plot()` method returns a '`ggplot`' object (Wickham 2016), so that the plot can be conveniently customized using **ggplot2**:

```
R> plot(fit_normal) +
+    geom_point(aes(color = sqrt(wOBA$PA))) +
+    labs(x = "wOBA", y = "EB estimate of true wOBA skill",
+       color = expression(sqrt(PA))) +
+    scale_color_gradient(low = "blue", high = "red")
```

The resulting plot, shown in Figure 4, compares the observed wOBAs against the posterior estimates of hitting ability returned by `ebnm()`. We customized the plot to vary the color of points by the number of plate appearances. The plot shows the expected shrinkage behavior of the EBNM model: wOBAs associated with fewer plate appearances (the blue points) were shrunk toward the league average (near 0.300) much more strongly than wOBAs for hitters with many plate appearances (the red points).

Let us now revisit the first 6 hitters in the data set. The `fitted()` method returns a posterior summary for each hitter (by default, the posterior mean and standard deviation):

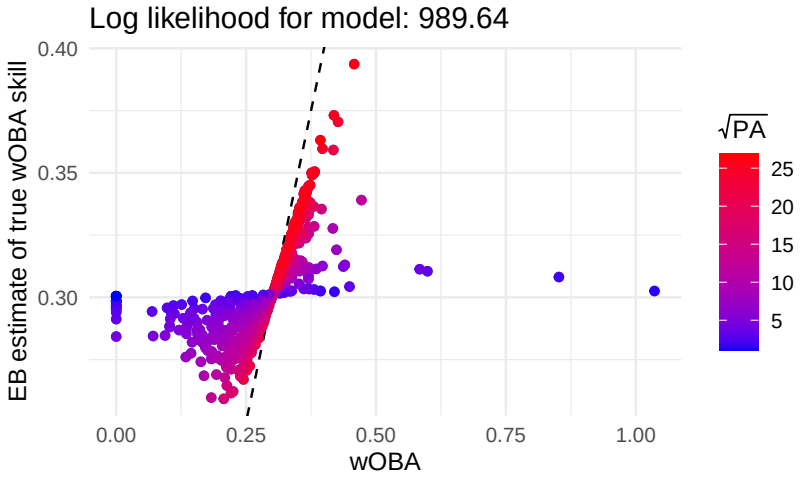

Figure 4: Initial wOBA estimates vs. posterior mean wOBA obtained by fitting a prior from the family of normal distributions. The color of the points is varied by the number of plate appearances. The dashed line shows the diagonal ($x = y$) line.

```
> print(head(fitted(fit_normal)), digits = 3)
```

```
                mean     sd
Khalil Lee     0.303 0.0287
Chadwick Tromp 0.308 0.0286
Otto Lopez     0.310 0.0283
James Outman   0.311 0.0282
Matt Carpenter 0.339 0.0254
Aaron Judge    0.394 0.0184
```

Estimates for the first four ballplayers are shrunk strongly toward the league average, reflecting the fact that these players had very few plate appearances. Carpenter had more plate appearances, but according to this model we should remain skeptical about his strong performance; after factoring in the prior, we judge his "true" talent to be much closer to the league average, downgrading an observed wOBA of 0.472 to a posterior mean estimate of 0.339.

5.3. Reanalyzing the wOBA data with a different prior

Judge's "true" talent was also estimated to be much lower (.394) than his observed wOBA (.458) despite sustaining this high level of production over a full season (696 PAs). One might ask whether a prior that is more flexible than the normal prior – in particular, a prior that can better adapt to "outliers" like Judge – might produce a different result. The **ebnm** package is well suited to answering this question. For example, to analyze the data using the family of all unimodal priors rather than a normal prior, we need only update the argument to `prior_family`:

```
R> fit_unimodal <- ebnm(x, s, prior_family = "unimodal", mode = "estimate")
```

Using this prior, estimates for players with many plate appearances and outlying performances (very high or very low wOBAs) are not adjusted quite so strongly toward the league average. In particular, Judge's estimated "true" talent is now much closer to his observed wOBA:

```
R> dat <- cbind(wOBA[, c("PA","x")], fitted(fit_normal),
+    fitted(fit_unimodal))
R> names(dat) <- c("PA", "x", "mean_n", "sd_n", "mean_u", "sd_u")
R> print(head(dat), digits = 3)

                PA     x mean_n   sd_n mean_u   sd_u
Khalil Lee       2 1.036  0.303 0.0287  0.302 0.0277
Chadwick Tromp   4 0.852  0.308 0.0286  0.307 0.0306
Otto Lopez      10 0.599  0.310 0.0283  0.310 0.0315
James Outman    16 0.584  0.311 0.0282  0.311 0.0318
Matt Carpenter 154 0.472  0.339 0.0254  0.355 0.0430
Aaron Judge    696 0.458  0.394 0.0184  0.439 0.0155
```

Carpenter's estimated "true" talent is also higher, but is appropriately adjusted much more strongly toward the league average than Judge's in light of Carpenter's smaller number of PAs. Interestingly, the unimodal prior also assigns greater uncertainty to Carpenter's estimate than the normal prior (compare the "`sd_u`" to the "`sd_n`" column).

5.4. A reanalysis using a nonparametric prior

An alternative to prior families that require specific assumptions about the data is to use the prior family that contains *all* distributions, $\mathcal{G}_{\text{npmle}}$, which is in a sense "assumption free." Although nonparametric priors require specialized computational techniques, switching to a nonparametric prior is seamless in **ebnm** as these implementation details are hidden. As before, we need only make a single change to the `prior_family` argument:

```
R> fit_npmle <- ebnm(x, s, prior_family = "npmle")
```

(Note that because $\mathcal{G}_{\text{npmle}}$ is not unimodal, the `mode = "estimate"` option is not needed here.)

We compare the three fits (normal, unimodal, and NPMLE) using `plot()`. To focus on results for Judge and other players with large numbers of PAs, we use the `subset` argument. Additionally, we set `incl_cdf = TRUE` to show the cumulative distribution functions (CDFs) of the fitted priors:

```
R> top50 <- order(wOBA$PA, decreasing = TRUE)
R> top50 <- top50[1:50]
R> plot(fit_normal, fit_unimodal, fit_npmle, incl_cdf = TRUE, subset = top50)
```

The plots generated by this call are shown in Figures 5 and 6. The estimates across the three analyses largely agree, differing mainly at the tails. Observe that the unimodal prior family and the NPMLE avoid the strong shrinkage behavior of the normal prior family at the tails.

Fits can be compared quantitatively using the `logLik()` method, which, in addition to the log-likelihood, reports the number of free parameters or "degrees of freedom" (note however that Wilks' theorem does not apply to these nonparametric comparisons):

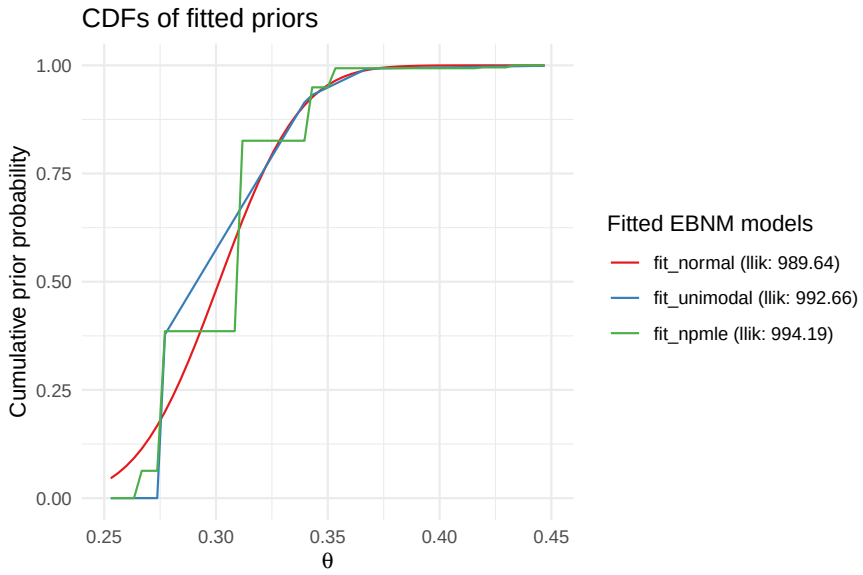

Figure 5: The CDFs for priors fitted to the 2022 MLB wOBA data: the normal prior (`prior_family = "normal"`), the unimodal prior (`prior_family = "unimodal"`), and the NPMLE (`prior_family = "npmle"`).

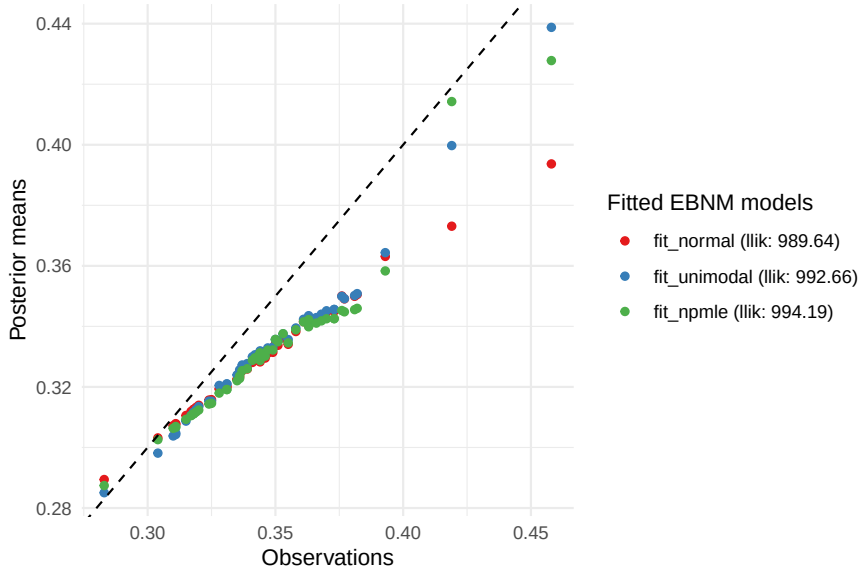

Figure 6: Initial wOBA estimates vs. posterior mean wOBA estimates for priors fitted to the 2022 MLB wOBA data: the normal prior (`prior_family = "normal"`), the unimodal prior (`prior_family = "unimodal"`), and the NPMLE (`prior_family = "npmle"`). Results are shown only for the top 50 ballplayers by number of PAs.

```
R> logLik(fit_unimodal)

'log Lik.' 992.6578 (df=40)
```

```
R> logLik(fit_npmle)

'log Lik.' 994.193 (df=94)
```

A nonparametric prior is approximated by K mixture components on a fixed grid (see Sections 2.3–2.4). We infer from the above output that the family of unimodal priors has been approximated by a family of mixtures with $K = 41$ fixed components, while $\mathcal{G}_{\text{npmle}}$ has been approximated as a family of mixtures with a grid of $K = 95$ point masses spanning the range of the data. (The number of degrees of freedom is one fewer than K because the mixture proportions $\boldsymbol{\pi}$ must always sum to 1.)

One potential issue with the NPMLE is that, since it is discrete (as Figure 5 makes apparent), observations are variously shrunk toward one of the support points, which can result in poor interval estimates. For illustration, we calculate 80% posterior credible intervals (since credible intervals are obtained using Monte Carlo methods, we set a seed for reproducibility):

```
R> fit_npmle <- ebnm_add_sampler(fit_npmle)
R> set.seed(123)
R> print(head(confint(fit_npmle, level = 0.8)), digits = 3)

               CI.lower CI.upper
Khalil Lee        0.265    0.309
Chadwick Tromp    0.276    0.342
Otto Lopez        0.276    0.342
James Outman      0.276    0.342
Matt Carpenter    0.309    0.419
Aaron Judge       0.430    0.430
```

Each credible interval endpoint is constrained to lie at one of the support points of the NPMLE $\hat{g}$. Note in particular that the NPMLE yields a degenerate interval estimate for Judge. To address this and other issues, the **deconvolveR** package (Narasimhan and Efron 2020) uses a penalized likelihood that encourages "smooth" priors – that is, priors for which few of the mixture proportions are zero:

```
R> fit_deconv <- ebnm_deconvolver(x / s, output = ebnm_output_all())
```

Note however that since package **deconvolveR** fits a model to z scores rather than observations and standard errors, the "true" means θ_i being estimated are z scores rather than raw wOBA talent. While this may be reasonable in many settings, it does not seem appropriate for the wOBA data set:

```
R> set.seed(123)
R> print(head(confint(fit_deconv, level = 0.8) * s), digits = 3)

               CI.lower CI.upper
Khalil Lee        0.000    1.600
Chadwick Tromp    0.563    1.127
Otto Lopez        0.442    0.796
```

```
James Outman       0.412    0.742
Matt Carpenter     0.413    0.531
Aaron Judge        0.406    0.459
```

These interval estimates do not match our expectations; for example, no player has ever sustained a wOBA of 0.600 over a full season.

6. Building on ebnm for new matrix factorization methods

As outlined in our Introduction, the EBNM model underlies many other well-studied statistical problems. One example is *matrix factorization*: as Wang and Stephens (2021) showed, fitting an empirical Bayes matrix factorization (EBMF) model can be reduced to solving a sequence of EBNM problems (typically very many of them). Therefore, the aspects that we have emphasized in developing **ebnm** – the unified interface, the variety of prior families and fitting options, and the speed and robustness of the numerical optimization – have greatly facilitated the creation of a flexible software framework for EBMF in the R package **flashier** (Willwerscheid *et al.* 2024), which is available on CRAN and GitHub (https://github.com/willwerscheid/flashier/).

Matrix factorization attempts to approximate a data matrix $\mathbf{X}$ by a low-rank matrix product, $\mathbf{X} \approx \mathbf{L}\mathbf{F}^\top$. The EBMF approach introduces priors on the low-rank matrices $\mathbf{L}$ and $\mathbf{F}$:

$$\begin{aligned} \mathbf{X} &= \mathbf{L}\mathbf{F}^\top + \mathbf{E} \\ e_{ij} &\sim \mathcal{N}(0, \sigma^2) \\ \ell_{ik} &\sim g_\ell^{(k)} \in \mathcal{G}_\ell \\ f_{jk} &\sim g_f^{(k)} \in \mathcal{G}_f, \end{aligned}$$

where $\mathbf{X}$, $\mathbf{L}$, $\mathbf{F}$, and $\mathbf{E}$ are, respectively, matrices of dimension $n \times p$, $n \times K$, $p \times K$, and $n \times p$ storing real-valued elements x_{ij}, l_{ik}, f_{jk}, and e_{ij}, and $\mathcal{G}_\ell$, $\mathcal{G}_f$ are specified prior families. In brief, each iteration of the EBMF model-fitting algorithm involves solving an EBNM problem separately for each column of $\mathbf{L}$ (using the prior family $\mathcal{G}_\ell$) and each column of $\mathbf{F}$ (using the prior family $\mathcal{G}_f$). The solutions to these EBNM problems yield fitted priors $\hat{g}_\ell^{(k)}$, $\hat{g}_f^{(k)}$ and posterior estimates of ℓ_{ik} and f_{jk}. See Wang and Stephens (2021) for details.

The EBMF framework is highly flexible in that different choices of prior families $\mathcal{G}_\ell$ and $\mathcal{G}_f$ can give very different factorizations. For example, the use of normal priors yields factorizations similar to the truncated singular value decomposition (SVD) (Nakajima and Sugiyama 2011). The use of *sparse priors* (e.g., the point-normal prior family) can yield *sparse matrix factorizations* which in many settings are more interpretable than an SVD (Engelhardt and Stephens 2010; Yang, Ma, and Buja 2014; Witten, Tibshirani, and Hastie 2009). By choosing priors with nonnegative support (e.g., the point-exponential family), one can obtain *nonnegative factorizations* (Lee and Seung 1999). More novel combinations are also possible; for example, one can obtain a *semi-nonnegative matrix factorization* (Ding, Li, and Jordan 2010; Wang, Fischer, and Song 2019; He *et al.* 2020) by choosing a prior family with nonnegative support for $\mathcal{G}_\ell$ and a prior family without sign constraints for $\mathcal{G}_f$. In yet another example, Liu *et al.* (2025) recently proposed the family of "generalized binary" priors to encourage binary-valued l_{ik}.

By building on the fast and reliable methods in **ebnm**, the **flashier** package makes it straightforward to generate any of these matrix factorizations. For example, a sparse factorization can be obtained by calling the **flashier** function `flash()` with argument `ebnm_fn = ebnm_point_normal`, which specifies point-normal distributions for all priors $g_\ell^{(k)}$ and $g_f^{(k)}$. To obtain a sparse, semi-nonnegative factorization, one can set `ebnm_fn = c(ebnm_point_exponential, ebnm_point_normal)`, which specifies point-normal priors for all $g_f^{(k)}$ and point-exponential priors for all $g_\ell^{(k)}$. In general, any of the prior families discussed above (Table 1) can be used, and if some other option is desired, it is not difficult to implement a new "ebnm-style" function (see the **ebnm** package vignette for details).

We provide a detailed illustration of these ideas in the **flashier** package vignette, "Introduction to **flashier**", which is available on the package's website (https://willwerscheid.github.io/flashier/).

7. Summary

The **ebnm** package provides a comprehensive toolkit for solving the empirical Bayes normal means (EBNM) problem under a variety of prior assumptions. In many situations – as in our analysis of baseball statistics in Section 5 – the "best" choice of prior family is not known in advance. The **ebnm** package is especially well suited to handling such situations by providing: (i) a large set of prior families to choose from (Table 1); and (ii) an interface that allows for convenient comparison of prior families. When deciding which prior family to use, our general recommendation is to weigh prior assumptions about the data against empirical measures of fit. The best prior will often depend on the context, and for this reason we have designed **ebnm** to be easily extensible so that researchers are not limited by the existing options. Our ultimate hope is that experts in other research areas will consider contributing to our package and help expand the use of EBNM methods to other domains.

Acknowledgments

We thank William Denault, Andrew Goldstein, Yusha Liu and Zihao Wang for their input and contributions to the software. This work was supported by the NHGRI at the National Institutes of Health under award number 5R01HG002585.

References

Bates D, Lai R, Byrne S (2024). ***RCall.jl***. Julia package version 0.14.1, URL https://github.com/JuliaInterop/RCall.jl.

Bezanson J, Edelman A, Karpinski S, Shah VB (2017). "Julia: A Fresh Approach to Numerical Computing." *SIAM Review*, **59**(1), 65–98. doi:10.1137/141000671.

Bhadra A, Datta J, Polson NG, Willard B (2019). "Lasso Meets Horseshoe: A Survey." *Statistical Science*, **34**(3), 405–427. doi:10.1214/19-STS700.

Brown LD (2008). "In-Season Prediction of Batting Averages: A Field Test of Empirical Bayes and Bayes Methodologies." *The Annals of Applied Statistics*, **2**(1), 113–152. doi:10.1214/07-AOAS138.

Carvalho CM, Polson NG, Scott JG (2010). "The Horseshoe Estimator for Sparse Signals." *Biometrika*, **97**(2), 465–480. doi:10.1093/biomet/asq017.

Clyde M, George EI (2000). "Flexible Empirical Bayes Estimation for Wavelets." *Journal of the Royal Statistical Society B*, **62**(4), 681–698. doi:10.1111/1467-9868.00257.

Dicker LH, Zhao SD (2016). "High-Dimensional Classification via Nonparametric Empirical Bayes and Maximum Likelihood Inference." *Biometrika*, **103**(1), 21–34. doi:10.1093/biomet/asv067.

Ding CH, Li T, Jordan MI (2010). "Convex and Semi-Nonnegative Matrix Factorizations." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **32**(1), 45–55. doi:10.1109/TPAMI.2008.277.

Efron B (2010). *Large-Scale Inference: Empirical Bayes Methods for Estimation, Testing, and Prediction*, volume 1 of *Institute of Mathematical Statistics Monographs*. Cambridge University Press, Cambridge. doi:10.1017/CBO9780511761362.

Efron B, Morris C (1972). "Limiting the Risk of Bayes and Empirical Bayes Estimators – Part II: The Empirical Bayes Case." *Journal of the American Statistical Association*, **67**(337), 130–139. doi:10.1080/01621459.1972.10481215.

Efron B, Morris C (1973). "Stein's Estimation Rule and Its Competitors – An Empirical Bayes Approach." *Journal of the American Statistical Association*, **68**(341), 117–130. doi:10.1080/01621459.1973.10481350.

Engelhardt BE, Stephens M (2010). "Analysis of Population Structure: A Unifying Framework and Novel Methods Based on Sparse Factor Analysis." *PLOS Genetics*, **6**(9), 1–12. doi:10.1371/journal.pgen.1001117.

FanGraphs (2023). "Guts!" URL https://www.fangraphs.com/guts.aspx.

Gautier L (2023). ***rpy2**: Bridge between Python and R*. Python package version 3.5.15, URL https://rpy2.github.io/.

Gelman A, Carlin JB, Stern HS, Dunson DB, Vehtari A, Rubin DB (2014). *Bayesian Data Analysis*. 3rd edition. CRC Press, Boca Raton. URL https://www.stat.columbia.edu/~gelman/book/.

Geyer CJ (2020). ***trust**: Trust Region Optimization*. doi:10.32614/CRAN.package.trust. R package version 0.1-8.

Gu J, Koenker R (2017). "Empirical Bayesball Remixed: Empirical Bayes Methods for Longitudinal Data." *Journal of Applied Econometrics*, **32**(3), 575–599. doi:10.1002/jae.2530.

Hastie T, Tibshirani R, Friedman J (2009). *The Elements of Statistical Learning*. 2nd edition. Springer-Verlag, New York. doi:10.1007/978-0-387-84858-7.

He Y, Chhetri SB, Arvanitis M, Srinivasan K, Aguet F, Ardlie KG, Barbeira AN, Bonazzola R, Im HK, GTEx Consortium, Brown CD, Battle A (2020). "sn-spMF: Matrix Factorization Informs Tissue-Specific Genetic Regulation of Gene Expression." *Genome Biology*, **21**(235). doi:10.1186/s13059-020-02129-6.

Henson R (2024). *MATLAB R-link*. URL https://www.mathworks.com/matlabcentral/fileexchange/5051-matlab-r-link.

Ignatiadis N, Wager S (2022). "Confidence Intervals for Nonparametric Empirical Bayes Analysis." *Journal of the American Statistical Association*, **117**(539), 1149–1166. doi:10.1080/01621459.2021.2008403.

James W, Stein C (1961). "Estimation with Quadratic Loss." In *Berkeley Symposium on Mathematical Statistics and Probability, 1961*, pp. 361–379. University of California Press, Berkeley. doi:10.1007/978-1-4612-0919-5_30.

Jiang W, Zhang CH (2009). "General Maximum Likelihood Empirical Bayes Estimation of Normal Means." *The Annals of Statistics*, **37**(4), 1647–1684. doi:10.1214/08-AOS638.

Jiang W, Zhang CH (2010). "Empirical Bayes In-Season Prediction of Baseball Batting Averages." In *Borrowing Strength: Theory Powering Applications – A Festschrift for Lawrence D. Brown*, volume 6 of *Institute of Mathematical Statistics Collections*, pp. 263–273. Institute of Mathematical Statistics, Beachwood. doi:10.1214/10-IMSCOLL618.

Johnstone I (2019). "Gaussian Estimation: Sequence and Wavelet Models." URL https://imjohnstone.su.domains/GE_08_09_17.pdf.

Johnstone I, Silverman BW (2005a). "**EbayesThresh**: R Programs for Empirical Bayes Thresholding." *Journal of Statistical Software*, **12**(8), 1–38. doi:10.18637/jss.v012.i08.

Johnstone IM, Silverman BW (2004). "Needles and Straw in Haystacks: Empirical Bayes Estimates of Possibly Sparse Sequences." *The Annals of Statistics*, **32**(4), 1594–1649. doi:10.1214/009053604000000030.

Johnstone IM, Silverman BW (2005b). "Empirical Bayes Selection of Wavelet Thresholds." *The Annals of Statistics*, **33**(4), 1700–1752. doi:10.1214/009053605000000345.

Judge J (2019). "Entirely Beyond WOWY: A Breakdown of DRC+." *Baseball Prospectus*. URL https://www.baseballprospectus.com/news/article/48293/entirely-beyond-wowy-a-breakdown-of-drc/.

Kiefer J, Wolfowitz J (1956). "Consistency of the Maximum Likelihood Estimator in the Presence of Infinitely Many Incidental Parameters." *The Annals of Mathematical Statistics*, **27**(4), 887–906. doi:10.1214/aoms/1177728066.

Kim Y, Carbonetto P, Stephens M, 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. doi:10.1080/10618600.2019.1689985.

Kim Y, Wang W, Carbonetto P, Stephens M (2024). "A Flexible Empirical Bayes Approach to Multiple Linear Regression and Connections with Penalized Regression." *Journal of Machine Learning Research*, **25**(185), 1–59. URL https://jmlr.org/papers/v25/22-0953.html.

Koenker R (2017). "Bayesian Deconvolution: An R Vinaigrette." *Working Paper CWP38/17*, Centre for Microdata Methods and Practice, London. URL https://hdl.handle.net/10419/189756.

Koenker R, Gu J (2017). "**REBayes**: An R Package for Empirical Bayes Mixture Methods." *Journal of Statistical Software*, **82**(8), 1–26. doi:10.18637/jss.v082.i08.

Koenker R, Mizera I (2014). "Convex Optimization, Shape Constraints, Compound Decisions, and Empirical Bayes Rules." *Journal of the American Statistical Association*, **109**(506), 674–685. doi:10.1080/01621459.2013.869224.

Laird N (1978). "Nonparametric Maximum Likelihood Estimation of a Mixing Distribution." *Journal of the American Statistical Association*, **73**(364), 805–811. doi:10.1080/01621459.1978.10480103.

Lee DD, Seung HS (1999). "Learning the Parts of Objects by Non-Negative Matrix Factorization." *Nature*, **401**(6755), 788–791. doi:10.1038/44565.

Lindsay BG (1983). "The Geometry of Mixture Likelihoods: A General Theory." *The Annals of Statistics*, **11**(1), 86–94. doi:10.1214/aos/1176346059.

Liu Y, Carbonetto P, Willwerscheid J, Oakes SA, Macleod KF, Stephens M (2025). "Dissecting Tumor Transcriptional Heterogeneity from Single-Cell RNA-seq Data by Generalized Binary Covariance Decomposition." *Nature Genetics*, **57**(1), 263–273. doi:10.1038/s41588-024-01997-z.

Love MI, Huber W, Anders S (2014). "Moderated Estimation of Fold Change and Dispersion for RNA-seq Data with **DESeq2**." *Genome Biology*, **15**, 550. doi:10.1186/s13059-014-0550-8.

Mersmann O (2019). ***microbenchmark**: Accurate Timing Functions*. doi:10.32614/CRAN.package.microbenchmark. R package version 1.4.10.

Morris CN (1983). "Parametric Empirical Bayes Inference: Theory and Applications." *Journal of the American Statistical Association*, **78**(381), 47–55. doi:10.1080/01621459.1983.10477920.

MOSEK ApS (2019). ***Rmosek**: The R to MOSEK Optimization Interface*. doi:10.32614/CRAN.package.Rmosek. R package version 1.3.5.

Mukherjee S, Sen B, Sen S (2023). "A Mean Field Approach to Empirical Bayes Estimation in High-Dimensional Linear Regression." *arXiv 2309.16843*, arXiv.org E-Print Archive. doi:10.48550/arXiv.2309.16843.

Nakajima S, Sugiyama M (2011). "Theoretical Analysis of Bayesian Matrix Factorization." *Journal of Machine Learning Research*, **12**, 2583–2648. URL https://jmlr.org/papers/v12/nakajima11a.html.

Narasimhan B, Efron B (2020). "**deconvolveR**: A *g*-Modeling Program for Deconvolution and Empirical Bayes Estimation." *Journal of Statistical Software*, **94**(11), 1–20. doi:10.18637/jss.v094.i11.

R Core Team (2025). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. doi:10.32614/R.manuals. URL https://www.R-project.org/.

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*, pp. 131–149. University of California Press, Berkeley. doi:10.1525/9780520411586-011.

Robbins H (1956). "An Empirical Bayes Approach to Statistics." In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, 1956, Vol. I*, pp. 157–163. University of California Press, Berkeley. doi:10.1007/978-1-4612-0919-5_26.

Robinson D (2017). *Introduction to Empirical Bayes: Examples from Baseball Statistics*. Self-published. URL https://github.com/dgrtwo/empirical-bayes-book.

Sharpe S (2019). "An Introduction to Expected Weighted On-Base Average (xwOBA)." *MLB Technology Blog*. URL https://technology.mlblogs.com/an-introduction-to-expected-weighted-on-base-average-xwoba-29d6070ba52b.

Silverman BW, Evers L, Xu K, Carbonetto P, Stephens M (2017). ***EbayesThresh**: Empirical Bayes Thresholding and Related Methods*. doi:10.32614/CRAN.package.EbayesThresh. R package version 1.4-12.

Smyth GK (2004). "Linear Models and Empirical Bayes Methods for Assessing Differential Expression in Microarray Experiments." *Statistical Applications in Genetics and Molecular Biology*, **3**(1). doi:10.2202/1544-6115.1027.

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*, pp. 197–206. University of California Press, Berkeley. doi:10.1525/9780520313880-018.

Stephens M (2017). "False Discovery Rates: A New Deal." *Biostatistics*, **18**(2), 275–294. doi:10.1093/biostatistics/kxw041.

Stephens M, Carbonetto P, Gerard D, Lu M, Sun L, Willwerscheid J, Xiao N (2023). ***ashr**: Methods for Adaptive Shrinkage, Using Empirical Bayes*. doi:10.32614/CRAN.package.ashr. R package version 2.2-63.

Sun L (2020). *Topics on Empirical Bayes Normal Means*. Ph.D. thesis, University of Chicago, Chicago.

Tango TM, Lichtman MG, Dolphin AE (2006). *The Book: Playing the Percentages in Baseball*. TMA Press.

The MathWorks Inc (2021). *MATLAB – The Language of Technical Computing, Version R2021a*. Natick. URL https://www.mathworks.com/products/matlab/.

Van der Pas S, Scott J, Chakraborty A, Bhattacharya A (2019). ***horseshoe**: Implementation of the Horseshoe Prior*. doi:10.32614/CRAN.package.horseshoe. R package version 0.2.0.

Van Rossum G, *et al.* (2011). *Python Programming Language*. URL https://www.python.org/.

Wang M, Fischer J, Song YS (2019). "Three-Way Clustering of Multi-Tissue Multi-Individual Gene Expression Data Using Semi-Nonnegative Tensor Decomposition." *The Annals of Applied Statistics*, **13**(2), 1103–1127. doi:10.1214/18-AOAS1228.

Wang W, Stephens M (2021). "Empirical Bayes Matrix Factorization." *Journal of Machine Learning Research*, **22**(120), 1–40. URL https://jmlr.org/papers/v22/20-589.html.

Wickham H (2016). ***ggplot2**: Elegant Graphics for Data Analysis*. Springer-Verlag, New York. URL https://ggplot2-book.org/.

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

Willwerscheid J, Carbonetto P, Wang W, Stephens M (2024). ***flashier**: Empirical Bayes Matrix Factorization*. doi:10.32614/CRAN.package.flashier. R package version 1.0.7.

Witten DM, Tibshirani R, Hastie T (2009). "A Penalized Matrix Decomposition, with Applications to Sparse Principal Components and Canonical Correlation Analysis." *Biostatistics*, **10**(3), 515–534. doi:10.1093/biostatistics/kxp008.

Yang D, Ma Z, Buja A (2014). "A Sparse Singular Value Decomposition Method for High-Dimensional Data." *Journal of Computational and Graphical Statistics*, **23**(4), 923–942. doi:10.1080/10618600.2013.858632.

Zhang Y, Cui Y, Sen B, Toh KC (2024). "On Efficient and Scalable Computation of the Nonparametric Maximum Likelihood Estimator in Mixture Models." *Journal of Machine Learning Research*, **25**(8), 1–46. URL https://www.jmlr.org/papers/v25/22-1120.html.

Zhu A, Ibrahim JG, Love MI (2019). "Heavy-Tailed Prior Distributions for Sequence Count Data: Removing the Noise and Preserving Large Differences." *Bioinformatics*, **35**(12), 2084–2092. doi:10.1093/bioinformatics/bty895.

A. Supplementary benchmarking results

A.1. Optimization methods for parametric families

We compared the performance of 6 optimization methods implemented in **ebnm** and controlled via the `optmethod` argument: 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"`). In detail, the 6 methods compared were as follows.

- Three of the methods call `nlm()`, a Newton-type algorithm. Gradient and Hessian functions can be provided; if they are not, `nlm()` estimates them numerically. Option `optmethod = "nlm"` provides both the gradient and Hessian, while the option `optmethod = "nohess_nlm"` provides the gradient but not the Hessian; `optmethod = "nograd_nlm"` provides neither.
- `optmethod = "lbfgsb"` and `optmethod = "nograd_lbfgsb"` call `optim()` with `method = "L-BFGS-B"`. The former provides the gradient function and the latter does not. L-BFGS-B does not accept a Hessian.
- `optmethod = "trust"` calls function `trust()`, a trust-region method implemented by the **trust** package (Geyer 2020). `trust()` requires both a gradient and Hessian.

For all methods, the parameters being optimized were transformed so that the optimization problem was unconstrained. (We used a log transformation for scale parameters, which are nonnegatively constrained, and a logit transformation for proportions, which are constrained to lie between zero and one.)

We ran `ebnm()` on simulated data sets in $2 \times 2 \times 3 \times 2 \times 3 = 72$ combinations of the following settings:

- The prior family was `point_normal` or `point_laplace`.
- The mode was either fixed at zero via (`mode = 0`) or estimated (`mode = "estimate"`).
- The prior distribution used to simulate the true means θ_i was: (i) a true member of the prior family – i.e., point-normal, $\pi_0\delta_\mu + (1-\pi_0)N(\mu, a^2)$, or point-Laplace, $\pi_0\delta_\mu + (1-\pi_0)\text{Laplace}(\mu, a)$, with $\pi_0 \sim \text{Beta}(10, 2)$, $a \sim \text{Gamma}(4, 1)$, and $\mu = 0$ (when the mode was fixed) or $\mu \sim \text{Unif}(-10, 10)$ (when the mode was estimated); (ii) the "null" distribution, δ_0; or (iii) a prior from outside the specified prior family – when `mode = 0`, we used a point-normal or point-Laplace prior but with $\mu \sim \text{Unif}(-10, 10)$; when `mode = "estimate"`, we used a point-t_5 distribution, $\pi_0\delta_0 + (1-\pi_0)t_5(0, a)$, with π_0 and a randomly generated as above.
- Either homoskedastic or heteroskedastic noise was added to the true means θ_i: $x_i \sim N(\theta_i, 1)$ or $x_i \sim N(\theta_i, s_i^2)$, $s_i^2 - 1 \sim \text{Exp}(1)$, respectively.
- The number of observations, n, was 1 000, 10 000, or 100 000.

For each scenario, we ran $10^6/n$ simulations and recorded runtimes using the **microbenchmark** R package (Mersmann 2019). All experiments were performed on a 2021 MacBook Pro with an Apple M1 Max processor and 64 GB memory. Results are shown in Figures 7 and 8.

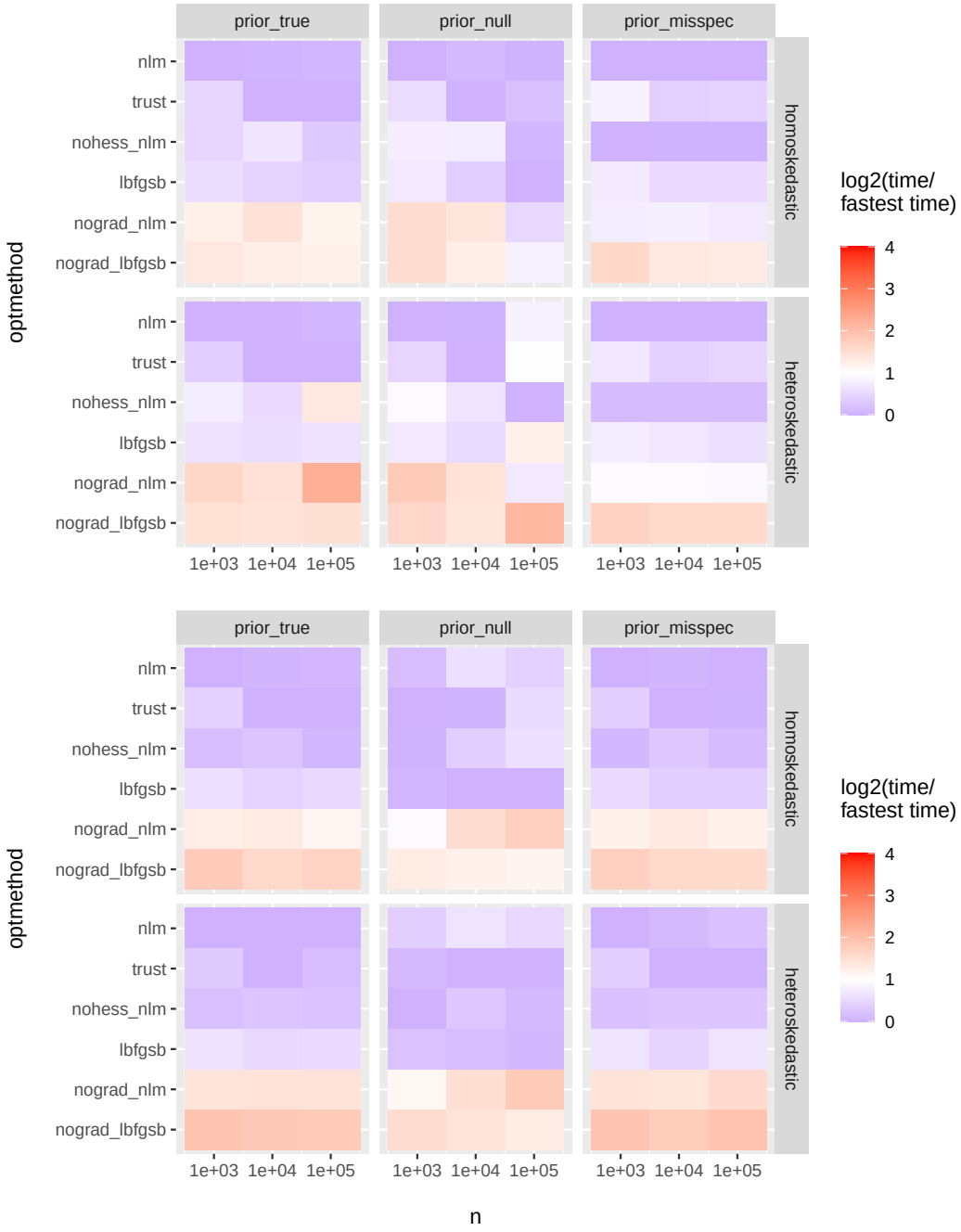

Figure 7: Runtimes for EBNM analyses with `prior_family = "point_normal"`. The three panels left-to-right correspond to different ways of simulating the true means θ_i: using a prior from the specified family (here, point-normal); the null distribution $\theta_i \sim \delta_0$; or a prior from outside the specified prior family (see text for details). For each plot, the two panels top-to-bottom give results for homoskedastic and heteroskedastic noise. Shown is the runtime as a multiple of the fastest runtime for a given simulation setting (i.e., a single column). Shades of red indicate optimization methods that are at least 50% slower than the fastest method; shades of blue indicate faster methods.

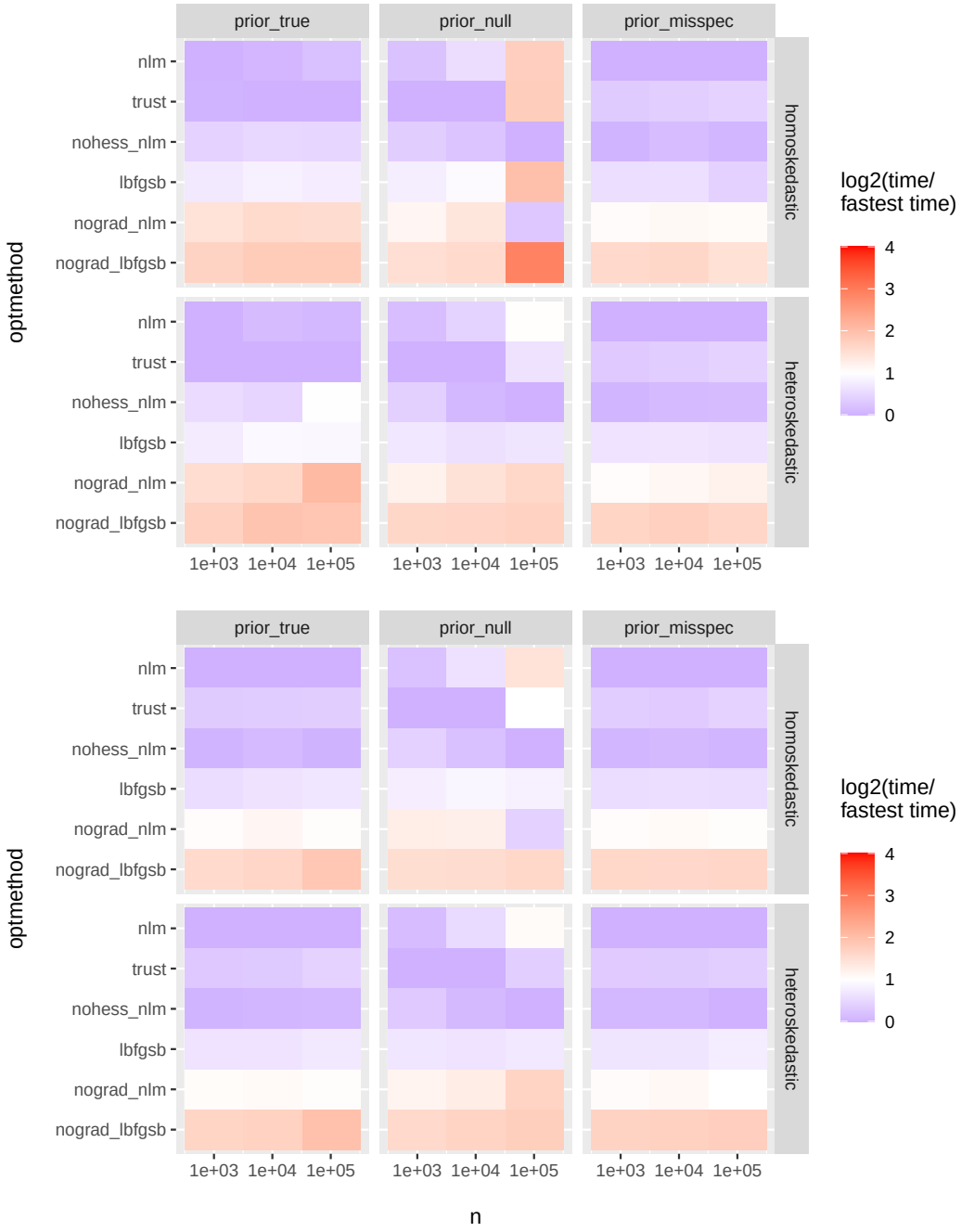

Figure 8: Runtimes for EBNM analyses with `prior_family = "point_laplace"`. See Figure 7 caption and the text for details.

Broadly, methods that supplied gradients outperformed methods that estimated all derivatives numerically. Therefore, our general recommendation is to provide gradient calculations whenever possible. (While automatic differentiation methods would facilitate this, as of this writing automatic differentiation methods are not as well supported in R as they are in other programming languages such as Python and Julia.)

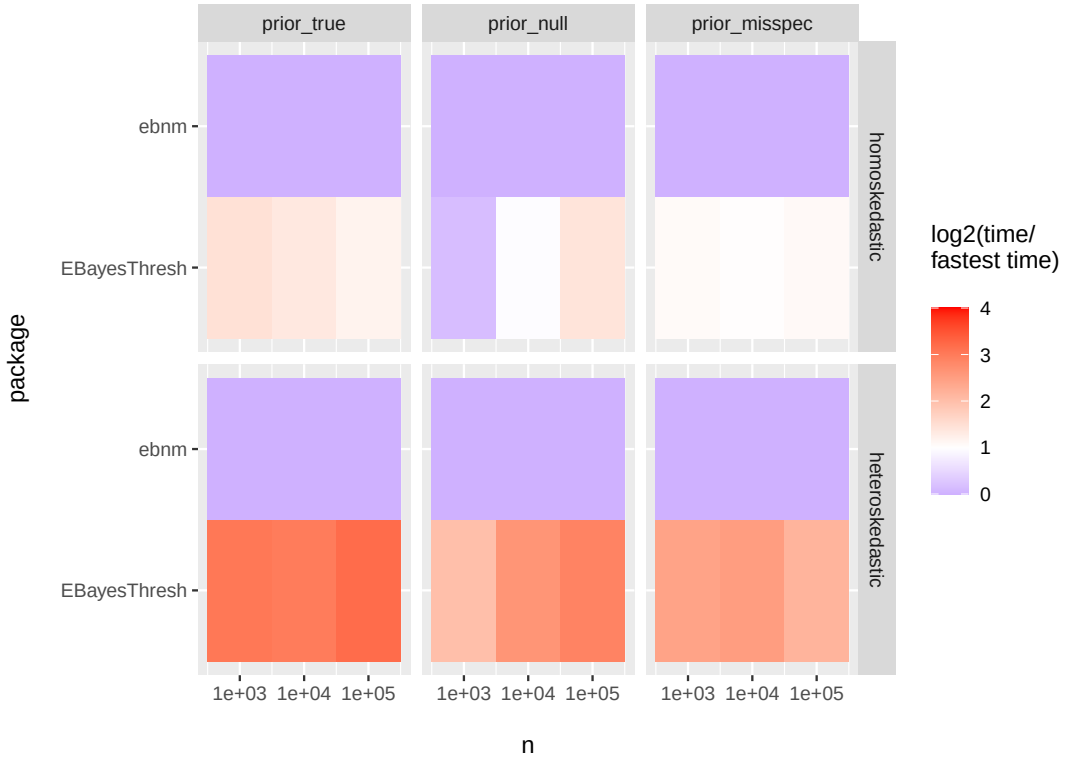

Figure 9: `ebnm_point_laplace()` vs. `ebayesthresh()` runtimes. See Figure 7 caption and the text for details.

The runtimes were similar among the four methods that supplied gradients. However, `"lbfgsb"` failed to converge in several of the simulations where the true prior was the "null" distribution δ_0. (In some settings, up to 7% of the simulations resulted in an error.) The methods that supplied Hessians (`"nlm"`, `"trust"`) also occasionally struggled to find solutions in "null" settings. Based on these results, we set the default to `optmethod = "nohess_nlm"`.

A.2. Comparisons with existing packages

We also compared **ebnm** with three packages that have similar functions: `ebnm_point_laplace()` is closely related to `ebayesthresh()` from the **EbayesThresh** package (Silverman, Evers, Xu, Carbonetto, and Stephens 2017); the function `ebnm_normal_scale_mixture()` is modeled after `ash()` in **ashr**, with `mixcompdist = "normal"`; and `ebnm_npmle()` is similar to `GLmix()` in the **REBayes** package (Koenker and Gu 2017). We considered the same combinations of settings as before, with the following changes: (i) the mode was always fixed at zero; (ii) we always simulated the true means from the point-Laplace distribution; and (iii) the number of grid points (mixture components) ranged from 10 to 300. We set the arguments to each of the functions to make outputs as similar as possible: for **EbayesThresh**, we set `threshrule = "mean"` and `universalthresh = FALSE`; and for `ash()`, we set `prior = "uniform"`. Results are shown in Figures 9–11.

In some scenarios, **EbayesThresh** was nearly as fast as `ebnm_point_laplace()`, but in others **ebnm** was an order of magnitude faster than **EbayesThresh** (Figure 9). Further, **ebnm** usually

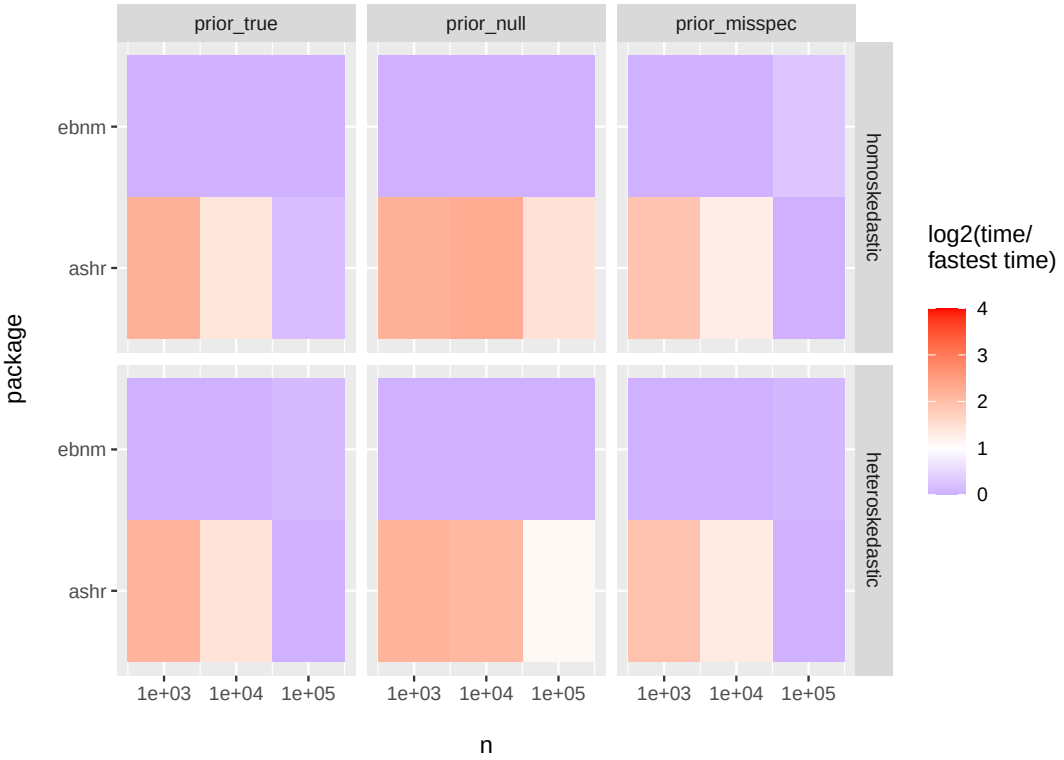

Figure 10: `ebnm_normal_scale_mixture()` vs. `ash()` runtimes. See Figure 7 caption and the text for details.

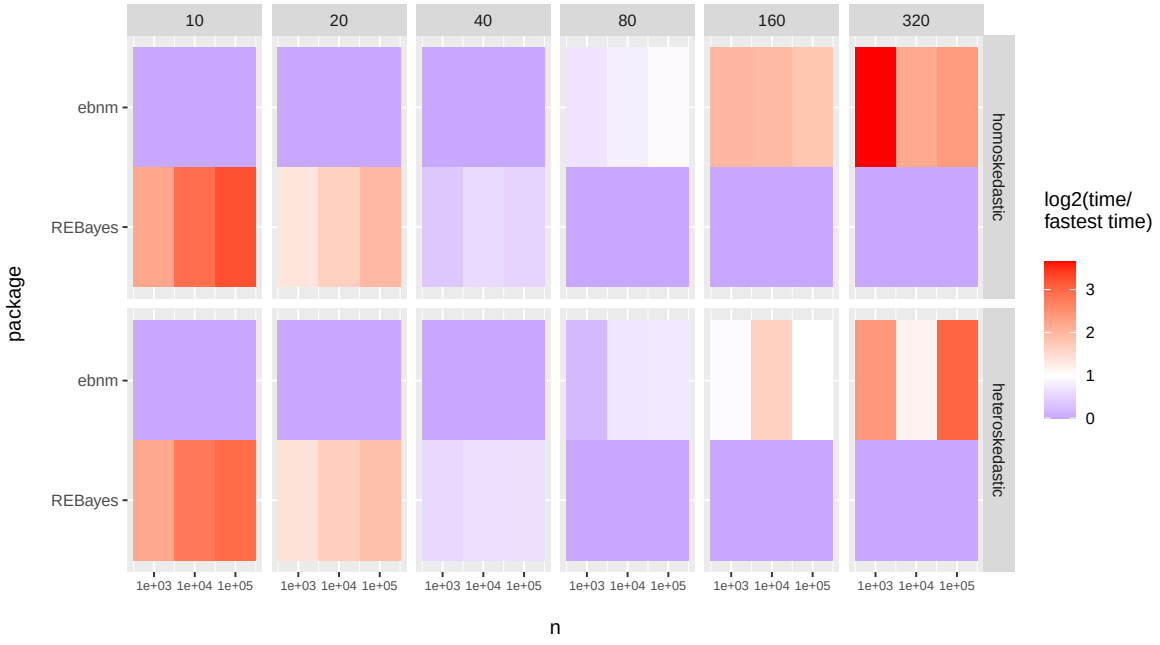

Figure 11: `ebnm_npmle()` vs. **REBayes** runtimes. Panels left-to-right show results for different numbers of NPMLE grid points. See Figure 7 caption and the text for details.

found substantially better solutions than **EbayesThresh** (in terms of the log-likelihood) except in the null setting, in which case the methods found solutions of similar quality.

For smaller data sets, **ebnm** was 2 to 4 times faster than **ashr**, but **ashr** had comparable speed to the **ebnm** function `ebnm_normal_scale_mixture()` for larger data sets (Figure 10).

The comparisons of `ebnm_npmle()` and **REBayes** were more mixed. **ebnm** was often faster when the number of mixture components was small (less than 80), while **REBayes** was consistently faster when a denser grid was used (80 or more components). From the theoretical results contained in Willwerscheid (2021), 80 components should be "good enough" for homoskedastic observations when

$$n^{1/4}\left(\frac{\text{range}(\boldsymbol{x})}{s}\right) \leq 80\sqrt{8}.$$

For example, if $n = 10\,000$, 80 components should suffice as long as

$$\frac{\max(\boldsymbol{x}) - \min(\boldsymbol{x})}{s} \leq 8\sqrt{8} \approx 22.6,$$

where $\frac{\max(\boldsymbol{x})-\min(\boldsymbol{x})}{s}$ is the "studentized range" of the data.

B. Background on weighted on-base averages

A longstanding tradition in empirical Bayes research is to analyze Major League Baseball batting averages (Brown 2008; Jiang and Zhang 2010; Gu and Koenker 2017). Until recently, batting averages were the most important measure of a hitter's performance, with the prestigious yearly "batting title" going to the hitter with the highest average. However, with the rise of baseball analytics, metrics that better correlate to teams' overall run production have become increasingly preferred. One such metric is wOBA ("weighted on-base average"), which, unlike competing metrics such as xwOBA (Sharpe 2019) and DRC+ (Judge 2019), can be calculated using publicly available data and methods.

Proposed by Tango, Lichtman, and Dolphin (2006), wOBA assigns values ("weights") to hitting outcomes according to how much the outcome contributes to average run production. For example, while batting average treats singles identically to home runs, wOBA assigns more than twice as much value to a home run. Although the weights are updated from year to year, the weight for singles has remained near 0.9 for the last several decades, and the weight for home runs has consistently been near 2.0 (FanGraphs 2023).

Given a vector of wOBA weights $\mathbf{w}$, hitter i's wOBA is the weighted average

$$x_i := \mathbf{w}^\top \mathbf{z}^{(i)}/n_i,$$

where $\mathbf{z}^{(i)} = (z_1^{(i)}, \ldots, z_7^{(i)})$ tallies outcomes (singles, doubles, triples, home runs, walks, hit-by-pitches, and outs) over the hitter's n_i plate appearances (PAs). Modeling hitting outcomes as

$$\mathbf{z}^{(i)} \overset{\text{ind.}}{\sim} \text{Multinomial}(n_i, \boldsymbol{\pi}^{(i)}), \tag{8}$$

where $\boldsymbol{\pi}^{(i)} = (\pi_1^{(i)}, \ldots, \pi_7^{(i)})$ is the vector of "true" outcome probabilities for hitter i, we interpret x_i as a point estimate for the hitter's "true hitting ability,"

$$\theta_i := \mathbf{w}^\top \boldsymbol{\pi}^{(i)}.$$

Standard errors for the x_i's are estimated as

$$s_i^2 = \mathbf{w}^\top \hat{\mathbf{\Sigma}}^{(i)} \mathbf{w} / n_i,$$

where $\hat{\mathbf{\Sigma}}^{(i)}$ is the estimate of the covariance matrix for the multinomial model in Equation 8 with $\boldsymbol{\pi} = \hat{\boldsymbol{\pi}}^{(i)}$, where $\hat{\boldsymbol{\pi}}^{(i)}$ is the vector of observed proportions, $\hat{\boldsymbol{\pi}}^{(i)} := \mathbf{z}^{(i)}/n_i$. To deal with small sample sizes, we (conservatively) lower bound each standard error by the standard error that would be obtained by plugging in league-average probabilities, $\hat{\boldsymbol{\pi}}_{\text{LA}} := \sum_{i=1}^N \mathbf{z}^{(i)} / \sum_{i=1}^N n_i$, where N is the total number of players.

The relative complexity of wOBA makes it well suited for **ebnm**. With batting averages, a common approach is to obtain empirical Bayes estimates using a Beta-binomial model (Robinson 2017). With wOBAs, one can estimate hitting outcome probabilities using a Dirichlet-multinomial model; alternatively, one can approximate the likelihood as normal and fit an EBNM model directly to the observed wOBAs. We take the latter approach.

Affiliation:

Jason Willwerscheid
Providence College
Department of Mathematics and Computer Science
Providence, Rhode Island, United States of America
E-mail: jwillwer@providence.edu

Peter Carbonetto
University of Chicago
Department of Human Genetics
Chicago, Illinois, United States of America
E-mail: pcarbo@uchicago.edu

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

Journal of Statistical Software https://www.jstatsoft.org/
published by the Foundation for Open Access Statistics https://www.foastat.org/
September 2025, Volume 114, Issue 3 *Submitted:* 2023-10-23
doi:10.18637/jss.v114.i03 *Accepted:* 2024-07-16