Empirical Bayes shrinkage, and denoising of Poisson and heteroskedastic Gaussian signals

1 Abstract

2 Introduction

Shrinkage and sparsity play a key role in many areas of modern statistics, including, for example, high-dimensional regression [?], covariance or precision matrix estimation [?], multiple testing [?] and signal denoising [?, ?]. One attractive way to achieve shrinkage and sparsity is via Bayesian or Empirical Bayes (EB) methods (e.g. [?, ?, ?, ?, ?]). However, these methods are usually perceived to require context-specific implementations, and this overhead can limit their use in practice. Here we consider a flexible EB approach to shrinkage, which we call *adaptive shrinkage*, whose goal is to provide a generic shrinkage method that could be useful for a range of applications. We show how this single shrinkage method can produce effective results for several denoising problems, including smoothing Gaussian means in the presence of heteroskedastic variance, smoothing of Gaussian variances, and smoothing Poisson means. These are all settings that are relatively underserved by existing EB implementations, and indeed we are unaware of any existing EB implementation for smoothing either the mean or the variance in the heteroskedastic Gaussian case. Consistent with previous studies ([?], [?]) we find these EB methods to be more accurate than commonly-used thresholding rules, and, in the Poisson case, competitive with a purpose-built EB method.

Methods for smoothing the variance of Gaussian data, or even for smoothing the mean in the presence of heteroskedastic errors, are uncommon, particularly compared with the well-studied homoskedastic case. Previous non-wavelet-based work includes [?], who estimated the variance by smoothing the squared residuals using local polynomial smoothing, [?], who employed difference-based kernel estimators, and [?], who developed a Bayesian mean field variational methodology for both mean and variance estimation. The only wavelet-based method for variance estimation we have found is [?], which applied a wavelet thresholding approach to first order differences. In no case could we find publicly-available software implementations of these methods (but we did receive code by email from [?], which we use in our comparisons).

For Poisson data, variance stabilizing techniques together with normal approximations have been proposed by [?] and [?], by exploiting the mean-variance relationship of a Poisson distribution. [?] and [?] derived thresholds achieving optimal asymptotic properties in the context of wavelet transformations, similar to the thresholds in the i.i.d. Gaussian case. However, [?] is computationally inefficient due to the presence of external cycle-spinning, and threshold-based methods may not result in satisfactory finite sample performance. Furthermore, all of these methods are sensitive to the choice of the primary resolution level as subsequent simulations will show. Multiscale analysis using recursive dyadic partitions within a Bayesian framework was later developed by [?] to make use of a particular form of likelihood factorization, but a relatively inflexible conjugate prior was chosen.

The *adaptive shrinkage* (`ash`) method is described in detail in a companion paper, [?], which applies `ash` to estimate false discovery rates (FDR) in multiple testing settings. Here we apply the same method to the signal denoising problems described above. In brief, `ash` aims to provide EB shrinkage estimates of quantities $\beta_1, \dots, \beta_J$, given observed estimates of these quantities $\hat{\beta}_1, \dots, \hat{\beta}_J$, and corresponding standard errors $\hat{s}_1, \dots, \hat{s}_J$. In common with most EB methods `ash` assumes that the β_j come from some common underlying distribution, g, which is estimated from all the data (as $\hat{g}$ say), and uses the posterior mean, $E(\beta_j|\hat{\beta}_j, \hat{s}_j, \hat{g})$ as a shrinkage estimate of β_j. Key features that make `ash` attractive as a flexible and generic approach to shrinkage include: i) its flexible semi-parametric modelling of g, under the constraint that it be uni-modal at 0 (and, optionally, symmetric); ii) its incorporation of the precision of each measurement through the standard errors $\hat{s}_j$, so that the amount of shrinkage adapts to the precision of each measurement; iii) its computational simplicity and speed: e.g. our `R` implementation typically takes a fraction of a second for $J = 1000$; iv) its computational stability, with, for example, minimal problems due to convergence to local optima [?].

The `ash` method makes assumptions, particularly conditional independence of observations, and normality of likelihood $p(\hat{\beta}_j|\beta_j, \hat{s}_j)$, that will often be violated in practice, and indeed are violated in our applications here. However, as we demonstrate later, it can nonetheless produce good shrinkage estimates. Essentially, we treat `ash` as a generic or "black box" shrinkage procedure, which inputs estimates and their corresponding standard errors and outputs shrinkage estimates, without worrying too much about the details. Of course, one must be careful about how far one takes this. And in more complex applications (e.g. large-scale regression and covariance estimation) it is unclear whether this strategy can be usefully applied. However, we believe our results here for signal denoising, together with those in [?] for FDR estimation, illustrate `ash`'s potential and flexibility.

Shrinkage methods are widely-used in signal denoising applications, because signal denoising can be accurately and conveniently achieved by shrinkage in a transformed (e.g. wavelet) domain [?]. Commonly-used shrinkage methods include both simple thresholding rules [?, ?, ?] and EB methods [?, ?]. Being an EB method, `ash` has much in common with these previous EB methods, but generalizes them in two ways:

first, `ash` allows more flexibility in the underlying distribution g than the Laplace or spike-and-slab distributions used in [?, ?]; second, `ash` allows for variations in precision in the transformed observations (e.g. wavelet coefficients). The latter property is particularly important for the Poisson and heteroskedastic Gaussian settings we consider here.

Software implementations of our methods are available in the R packages `ashr` (Adaptive SHrinkage in R) and `smash` (SMoothing by Adaptive SHrinkage), available from http://www.github.com/stephens999/ashr and http://www.github.com/stephenslab/smash respectively.

3 Methods

3.1 Adaptive shrinkage

Here we briefly outline the adaptive shrinkage method; see [?] for full details. Adaptive shrinkage (`ash`) is an EB method for estimating quantities $\beta = (\beta_1, \dots, \beta_n)$ from noisy estimates $\hat{\beta} = (\hat{\beta}_1, \dots, \hat{\beta}_n)$ and their corresponding standard errors $\hat{s} = (\hat{s}_1, \dots, \hat{s}_n)$. In its simpest form it assumes the hierarchical model

$$\beta_j \mid \hat{s}_j \sim g \tag{1}$$

$$\hat{\beta}_j \mid \beta_j, \hat{s}_j \sim N(\beta_j, \hat{s}_j^2), \tag{2}$$

where the distribution g is constrained to be unimodal and symmetric. This constraint on g can be flexibly achieved using a mixture of zero-centered normal distributions

$$g(\cdot) = \sum_{k=0}^{K} \pi_0 N(\cdot; 0, \sigma_k^2), \tag{3}$$

where the mixture weights $\pi_0, \dots, \pi_K$ are non-negative and sum to 1, and $N(\cdot; \mu, \sigma^2)$ denotes the density of a normal distribution with mean μ and variance σ^2. A key idea, which substantially simplies inference, is to take $\sigma_0, \dots, \sigma_K$ to be a fixed grid of values. ranging from very small (e.g. $\sigma_0 = 0$, in which case g includes a point mass at 0) to very large. Estimating g then boils down to estimating the mixture weights π, which is done by maximum likelihood using a simple EM algorithm. Given an estimate $\hat{g}$ for g, the conditional distributions $p(\beta_j \mid \hat{\beta}, \hat{s}, \hat{g})$ are analytically tractable, and the posterior mean $E(\beta_j \mid \hat{\beta}, \hat{s}, \hat{g})$ provides a shrinkage point estimate for β_j.

[?] also introduces various embellishments that are implemented in the `ashr` package, including generalizing the normal likelihood to a t likelihood, and dropping the symmetric constraint on g by replacing the mixture of normals with a more flexible (though less smooth) mixture of uniforms. However, we do not use these embellishments here.

3.2 Signal Denoising

A common generic signal denoising problem, sometimes known as "non-parametric regression", involves estimating a "spatially-structured" mean $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_T)$, from corresponding noisy observations $\boldsymbol{Y} = (Y_1, \ldots, Y_T)$, where $t = 1, \ldots, T$ indexes location in a one-dimensional space, such as time, or, as in our example later, location along the genome. By "spatially-structured" we mean that μ_t will often be similar to μ_t' for small $|t - t'|$, though we do not rule out occasional abrupt changes in μ. For convenience we assume that $T = 2^J$ for some integer J, as is common when using wavelet methods.

The most studied denoising problem is the homoskedastic Gaussian case; that is, the data Y_t have Gaussian noise and constant variance. Here we consider the more general case of Gaussian data with spatially-structured mean *and* spatially-structured (non-constant) variance. In some settings the changes in variance may themselves be of interest, and our methods provide explicit estimates for the variance as well as the mean. In addition we consider Poisson data (where the variance depends on the mean, so a spatially-structure mean implies spatially-structure variance). To build up to these more interesting cases we start with the simplest setting: Gaussian noise with known variance.

3.2.1 Gaussian noise; estimate mean (known variance)

Suppose the Y_t are independent noisy observations of the μ_t, with Gaussian noise of known variance σ_t^2. That is,

$$\boldsymbol{Y} = \boldsymbol{\mu} + \boldsymbol{\epsilon} \tag{4}$$

where $\boldsymbol{\epsilon} \sim N(0, D)$ with D the diagonal matrix with diagonal entries $(\sigma_1^2, \ldots, \sigma_T^2)$.

Wavelet denoising involves first applying a discrete wavelet transform to the data $\boldsymbol{Y}$. This involves pre-multiplying $\boldsymbol{Y}$ by an orthogonal $n \times n$ matrix W that depends on the orthonormal wavelet basis chosen. Pre-multiplying (4) by W yields

$$W\boldsymbol{Y} = W\boldsymbol{\mu} + W\boldsymbol{\epsilon} \tag{5}$$

which, using $\tilde{}$ to denote the wavelet transform, we write

$$\tilde{\boldsymbol{Y}} = \tilde{\boldsymbol{\mu}} + \tilde{\boldsymbol{\epsilon}}, \tag{6}$$

where $\tilde{\boldsymbol{\epsilon}} \sim N(0, WDW')$.

A key feature of the wavelet transform is that if $\boldsymbol{\mu}$ is spatially structured then many elements of $\tilde{\boldsymbol{\mu}} = W\boldsymbol{\mu}$ will tend to be close to zero, and vice versa. In other words, smoothing in the data domain corresponds to shrinking in the wavelet domain. Thus, one can obtain a spatially structured estimate of mean $\boldsymbol{\mu}$ by first using shrinkage methods to estimate $\tilde{\boldsymbol{\mu}}$ and then reversing the wavelet transform.

Here we use the adaptive shrinkage method to obtain shrinkage estimates for $\tilde{\boldsymbol{\mu}}$. Focussing on the marginals of (6), we have

$$\tilde{Y}_j | \tilde{\mu}_j, s_j^2 \sim N(\tilde{\mu}_j, \omega_j^2). \tag{7}$$

where

$$\omega_j^2 := \sum_{t=1}^{T} \sigma_t^2 W_{jt}^2, \tag{8}$$

are the diagonal elements of WDW'. Thus, applying ash with $\hat{\beta}_j = \tilde{Y}_j$ and $\hat{s}_j = \omega_j$ yields shrinkage estimates for the $\tilde{\mu}_j$. (In practice it is important to group the wavelet-transformed observations j by their resolution level before shrinking; see note below.) Of course, by focusing on the marginals we are ignoring any correlations among the $\tilde{Y}_j$, and this is the primary simplification here. (We are not alone in making this simplification; see also [?] for example.)

The above outlines the basic strategy, but there are some important additional implementational details:

- Rather than use a single wavelet transform, we use the "non-decimated" wavelet transform, which treats the observations as coming from a circle, rather than a line, and averages results over all T possible rotations of the data. Although not always necessary, this is a standard trick to reduce artifacts that can occur near discontinuities in the underlying signal (eg. [?]).
- The non-decimated wavelet transform yields T wavelet coefficients (transformed values of $\boldsymbol{Y}$) at each of $J = \log_2(T)$ resolution levels. We apply ash separately to the wavelet coefficients at each resolution level, so that a different distribution g for the $(\tilde{\mu}_j)$ is estimated for each resolution. This is the usual way that EB approaches are applied in this context (e.g. [?]) and indeed is crucial because the underlying distribution g will vary with resolution (because smoothness of $\boldsymbol{\mu}$ will vary with resolution).
- Although we have presented the wavelet transform as a matrix multiplication, which is an $o(T^2)$ operation, in practice both the wavelet transform and the inverse transform are implemented using efficient algorithms [?,?], implemented in the `wavethresh` package [?], taking only $O(T \log T)$ operations.

3.2.2 Estimating spatially structured variance (known mean)

Although we assumed variances to be known above, the problem of variance estimation is itself non-trivial. A common approach is to assume that the variance is constant ($\sigma_t = \sigma$), and that changes in the adjacent means $\mu_t - \mu_{t+1}$ are negligible, and to use an unbiased estimate for σ such as

$$\hat{\sigma}^2 = (1/2T) \sum_{t=1}^{T} (Y_t - Y_{t-1})^2 \tag{9}$$

with $Y_0 := Y_T$. Here we make the more flexible assumption that the variance function is spatially structured, and use wavelet shrinkage to estimate it. Our approach is similar in spirit to [?], and also to [?] (although the latter estimates only the variance function).

Assume initially that the mean μ_t is known, and define

$$Z_t^2 = (Y_t - \mu_t)_t^2 \tag{10}$$

to be the "observations" for the unknown variance function. Note that $\mathbb{E}(Z_t^2) = \sigma_t^2$, and so we have a mean estimation problem. To tackle this we use the mean estimation procedure above (effectively treating the wavelet-transformed values WZ_t^2 as Gaussian when really they are linear combinations of χ^2 random variables). To apply this procedure we need the variance of Z_t^2, $\mathbb{V}(Z_t^2)$, which is unknown. Here we use $\frac{2}{3}Z_t^4$ as an unbiased estimator for $\mathbb{V}(Z_t^2)$. (This follows from the distributional result that, when $Z^2 \sim \sigma^2\chi^2$, then $\mathbb{E}(Z^4) = 3\sigma^4$, and $\mathbb{V}(Z^2) = 2\sigma^4$.)

Despite the approximations being made here, we have found this procedure to work well in practice in most cases, perhaps with a tendancy to oversmooth quickly-varying variance functions.

3.2.3 Estimating spatially-structured mean and variance

Having specified procedures for estimating the means with variances known, and the variances with means known, we iterate these procedures to deal with the (typical) case where both are unknown. We initialize the algorithm by estimating the variance vector $\boldsymbol{\sigma}^2$ using

$$\hat{\sigma}_t^2 = \frac{1}{2}\left((Y_t - Y_{t-1})^2 + (Y_t - Y_{t+1})^2\right) \tag{11}$$

where $Y_0 \equiv Y_n$ and $Y_{T+1} \equiv Y_1$ (equivalent to putting the observations on a circle). Then we iterate:

1 Estimate $\boldsymbol{\mu}$ as if $\boldsymbol{\sigma}^2$ is known (with the value obtained from the previous step).
2 Estimate $\boldsymbol{\sigma}^2$ as if $\boldsymbol{\mu}$ is known (with the value obtained by the previous step); return to 1.

We cannot guarantee that this procedure will converge in general. In our simulations we found that two iterations of steps 1-2 yielded accurate results (so the full procedure consists of initialize + Steps 1-2-1-2).

3.3 Smoothing Poisson data

Now assume that each Y_t has a Poisson distribution with mean μ_t. To do denoising here we apply adaptive shrinkage to the Poisson multiscale models from [?] and [?], which are an analogue of wavelet methods for Poisson data.

To explain the idea, first recall the following elementary distributional result: if Y_1, Y_2 are independent, with $Y_j \sim \text{Poi}(\mu_j)$ then

$$Y_1 + Y_2 \sim \text{Poi}(\mu_1 + \mu_2) \tag{12}$$

$$Y_1 | (Y_1 + Y_2) \sim \text{Bin}(Y_1 + Y_2, \mu_1/(\mu_1 + \mu_2)). \tag{13}$$

To extend this to $T = 4$, introduce the notation $v_{i:j}$ to denote, for any vector v, the sum $\sum_{t=i}^{j} v_t$. Then

$$Y_{1:4} \sim \text{Poi}(\mu_{1:4}) \tag{14}$$
$$Y_{1:2}|Y_{1:4} \sim \text{Bin}(Y_{1:4}, \mu_{1:2}/\mu_{1:4}) \tag{15}$$
$$Y_1|Y_{1:2} \sim \text{Bin}(Y_{1:2}, \mu_1/\mu_{1:2}) \tag{16}$$
$$Y_3|Y_{3:4} \sim \text{Bin}(Y_{3:4}, \mu_3/\mu_{3:4}). \tag{17}$$

Together these models are exactly equivalent to $Y_j \sim \text{Poi}(\mu_j)$, and they decompose the overall distribution $Y_1, \dots, Y_4$ into parts involving aspects of the data at increasing resolution: (14) represents the coarsest resolution (the sum of all data points), whereas (16) and (17) represent the finest resolution, with (15) in between. Further, this representation suggests a reparameterization, from $\boldsymbol{\mu} = (\mu_1, \mu_2, \mu_3, \mu_4)$ to $\mu_{1:4}$ plus the binomial parameters $\boldsymbol{p} = (\mu_{1:2}/\mu_{1:4}, \mu_1/\mu_{1:2}, \mu_3/\mu_{3:4})$, where p_1 controls lower-resolution changes in the mean vector μ and p_2, p_3 control higher resolution changes.

This idea extends naturally to $T = 2^J$ for any J, reparameterizing $\boldsymbol{\mu}$ into its sum $\mu_{1:T}$ and a vector $\boldsymbol{p}$ of $T - 1$ binomial probabilities that capture features of $\boldsymbol{\mu}$ at different resolutions. This can be thought of as an analogue of the wavelet transform of $\boldsymbol{\mu}$ for Poisson data.

Note that, in this reparameterization, $p_j = 0.5 \, \forall j$ corresponds to the case of a constant mean vector, and values of p_j far from 0.5 correspond to large changes in μ (at some scale). Thus estimating a spatially-structured $\boldsymbol{\mu}$ can be achieved by shrinkage estimation of $\boldsymbol{p}$, with shrinkage towards $p_j = 0.5$. Both [**?**] and [**?**] use purpose-built Bayesian models to achieve this shrinkage, by introducing a prior distribution on elements of $\boldsymbol{p}$ that is a mixture of a point mass at 0.5 (creating shrinkage toward 0.5) and a Beta distribution. Here we take a different approach, reparameterizing $\alpha_j = \log(p_j/(1 - p_j))$, and then using adaptive shrinkage to shrink α_j towards 0.

To apply adaptive shrinkage we need an estimate $\hat{\alpha}_j$ and corresponding standard error $\hat{s}_j$ for each j. This boils down to estimating a log-odds ratio, and its standard error, which is a well-studied problem (e.g. [**?**]). The main challenge is in dealing satisfactorily with cases where the maximum likelihood estimator (MLE) for α_j is infinite. Our choice of estimates, based on results from [**?**], are described in Appendix **??**.

The output of adaptive shrinkage is a posterior distribution on each α_j. The simplest approach to obtain estimates of $\boldsymbol{\mu}$ would be to estimate α_j by its posterior mean, and then reverse the reparameterization. The resulting estimate of μ_t would be the exponential of the posterior mean for $\log(\mu_t)$ (because each $\log(\mu_t)$ is a linear combination of α_j). Alternatively we can estimate μ_t by approximating its posterior mean using the Delta method; see Appendix B. Both methods are implemented in our software; for the results here we use the latter method to be more comparable with previous approaches that estimate μ on the raw scale rather than log scale.

4 Results

4.1 Illustration

Figure 1 illustrates the main features of adaptive shrinkage (ash) applied to a denoising problem. The data (panel a) is first transformed into wavelet coefficients (WCs) at different scales. Each such "observed" WC can be thought of as a noisy estimate of some "true" WC for the (unknown) mean that we wish to estimate. We also compute, for each WC, an associated standard error that depends on the variance of the data about the mean. The idea behind wavelet denoising is to "shrink" the observed WCs towards 0, which produces a smoother estimate of the mean than the observed data.

A crucial question is, of course, how much to shrink. A key idea behind ash is that the shrinkage is "adaptive", in that it determined by the data, in two distinct ways. First, if at a particular scale many observed WCs are "large" (compared with their standard errors) then ash infers that, at this scale, many of the true WCs may also be large - that is, the estimated distribution g in (**??**) will have a long tail. Consequently ash will shrink less at this scale than at scales where few observed WCs are large, for which the estimated g will have a short tail. This is illustrated in panels b-c: at scale 1, many observed WCs are large (b), and so very little shrinkage is applied to these estimates (c). In contrast, at scale 7, few observed WCs are large (b), and so stronger shrinkage is applied (c). Second, the shrinkage is adaptive to the standard error for each WC: at a given scale, WCs with larger standard error are shrunk more strongly than WCs with small standard error. This is illustrated in panel d). (The standard errors vary among WCs here because of the hetereskedastic variance.)

The end result is that i) data that are consistent with a smooth signal, get smoothed more strongly; ii) smoothing is stronger in areas of the signal with larger variance. In this example the smoothed signal from ash is visually (and also quantitatively) more accurate than using TI-thresholding (with variance estimated by running median absolute deviation (RMAD); see [**?**]) (panel e).

4.2 Simulations

We conducted extensive simulation studies to compare the performance of our method with existing approaches.

4.2.1 Gaussian mean estimation

For the Gaussian case, we focus on mean estimation, initially with homoskedastic errors. We modelled our simulation study after [**?**], using many of the same test functions, a variety of sample sizes, two different signal to noise ratios (SNRs), and including the best-performing methods from their comparison. In particular their results showed that Translation Invariant (TI) thresholding ([**?**]) performed the best for most of the test signals, where performance is measured by mean squared error

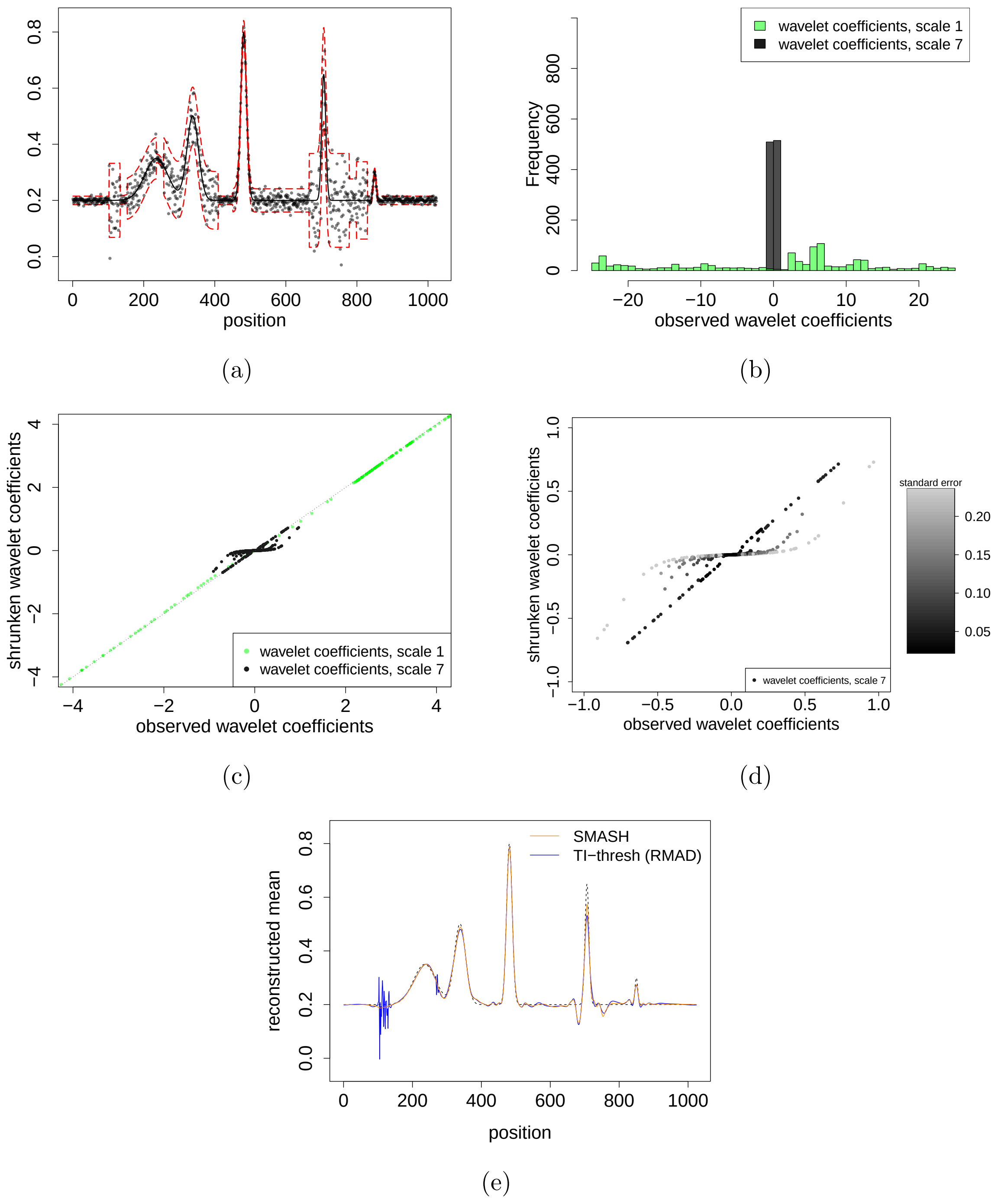


Figure 1: Shows both ASH and SMASH at work. (a) shows the mean function ± 2 standard deviations, as well as one realized dataset. (b) contrasts the difference in the distribution of the true wavelet coefficients at two different scales - one coarse (scale 1) and one fine (scale 7). (c) demonstrates the shrinkage properties of ASH for the two scales of interest (scales 1 and 7): the coefficients at the fine scale (scale 7) are substantially shrunk, but barely so for the coarse scale. (d) shows the shrinkage effect of ASH on the fine scale (scale 7) wavelet coefficients - wavelet coefficients are shrunk adaptively based on their precision. (e) plots the estimated mean functions from SMASH and TI-thresh against the true mean function; we see that TI-thresh has noticeable artifacts.

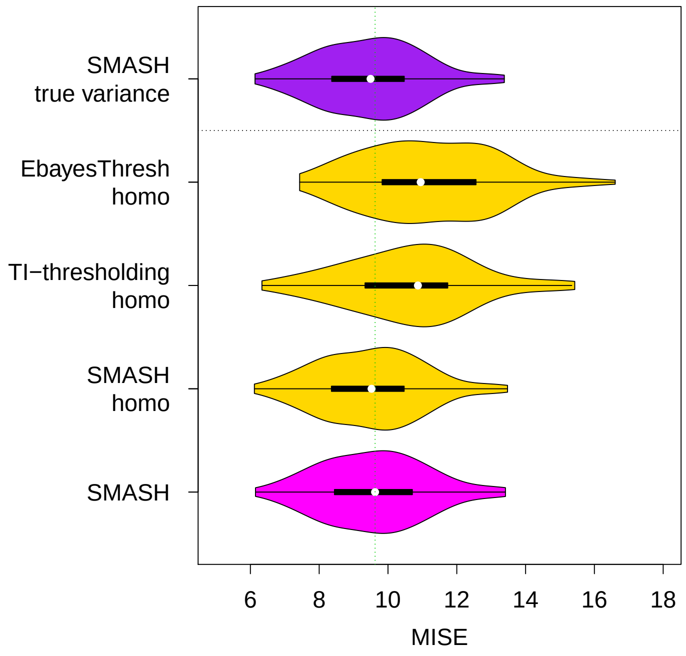


Figure 2: Comparison of wavelet-based methods for mean estimation with homoskedastic Gaussian errors for the "Spikes" mean function. Violin plots of MISEs for (from bottom to top) SMASH, SMASH with homoskedastic assumption, TI-thresholding with homoskedastic assumption, Ebayesthresh with homoskedastic assumption, and SMASH with known true variance are shown in the figure. Smaller MISE implies better performance; dashed green line indicates the median MISE for SMASH. SMASH outperforms both TI-thresholding and Ebayesthresh, and SMASH with estimated variance performs nearly as well as with true variance.

(MSE). We also considered the Empirical Bayes shrinkage procedure ([**?**]), named Ebayesthresh. All the methods were applied using the Symmlet8 wavelet basis [].

Figure 2 compares the mean integrated squared errors (MISEs) of TI-thresholding and Ebayesthresh to SMASH for the "Spikes" mean function, with a signal to noise ratio of 3 and sample size 1024. We applied SMASH in three ways, the first (SMASH) estimating the variance function allowing for heteroskedasticity, the second estimating the variance assuming homoskedasticity (SMASH-homo) and the third using the true variance function (SMASH true variance), which could be viewed as a "gold standard". The results show that all three versions of SMASH outperform both EbayesThresh and TI-thresholding. Notably all three SMASH versions perform very similarly, demonstrating that in this case there is little cost in allowing for heterskedasticity when the truth is homoskedastic.

We obtained similar results for other mean functions, SNRs and sample sizes (see Supplementary Materials).

Turning now to heteroskedastic errors, we compare results from SMASH (run three different ways, as above) with EbayesThresh (which assumes homoskedastic variance) and TI-thresh. For TI-thresh we considered three different ways of estimating the heteroskedastic variance: RMAD ([**?**]), the SMASH estimated variance, and the true variance. (TI-thresh with homoskedastic variance performed very poorly; Supplementary Results.) Figure 3 shows results for two sets of test functions: the "Spikes" mean function with the "Clipped Blocks" variance function and the "Corner" mean function with "Doppler" variance function, both with SNRs of 3 and sample sizes of 1024.

To summarize the main patterns in Figure 3:

1. SMASH outperforms all TI-thresh variants (including, here, TI with the true variance).
2. SMASH performs almost as well for mean estimation when estimating the variance as when given the true variance.
3. Allowing for heteroskedasticity within SMASH can substantially improve accuracy of mean estimation (compare SMASH with SMASH-homo and EbayesThresh).
4. TI-thresh performs considerably better when used with the SMASH variance estimate than with the RMAD variance estimate.

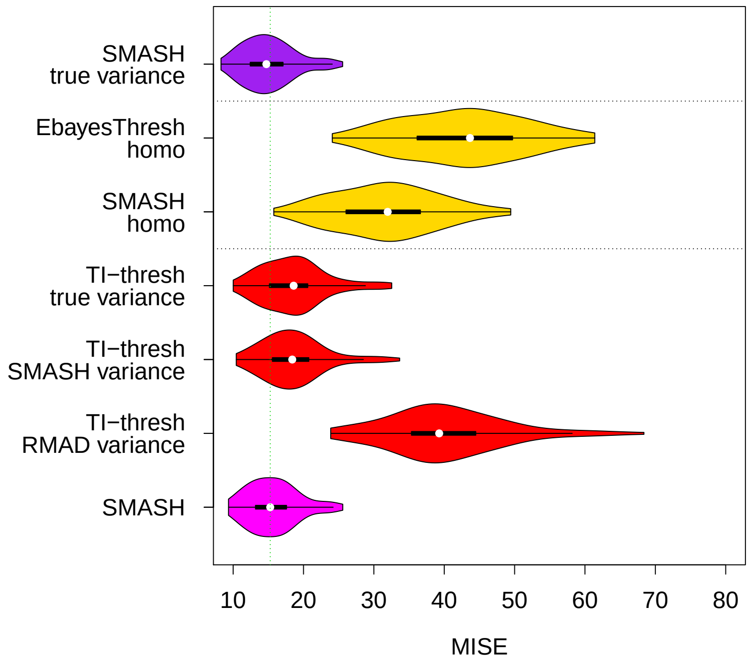
These main patterns hold for a variety of different mean and variance functions, SNRs, and sample sizes (see Supplementary Materials). Some variance functions are harder to estimate than others (e.g. the "Bumps" function), and in these cases providing methods the true variance can greatly increase accuracy compared with estimating the variance. As might be expected, the gain in allowing for heteroskedastic variance tends to be greatest when the variance functions are more volatile.

4.2.2 Gaussian variance estimation

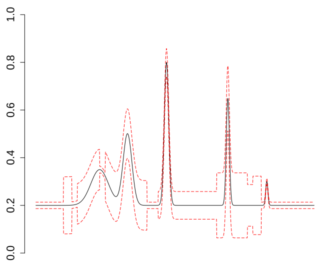
One unusual feature of SMASH is that it performs joint mean and variance estimation. Indeed, we found no existing R packages aimed at doing this. However we were able to obtain code implementing the Mean Field Variational Bayes (MFVB) method for heteroskedastic Gaussian regression [?] (M. Menictas, personal communication). This method is based on penalized splines, and so is not well suited to many standard test functions in the wavelet literature, which often contain "spiky" local features not well captured by splines. Hence, we compared SMASH and MFVB on some smoother mean and variance (standard deviation) functions, specifically scenario A in Figure 5 from [?] (Figure 4), using scripts kindly provided by M. Menictas.

We simulated data under two different scenarios:

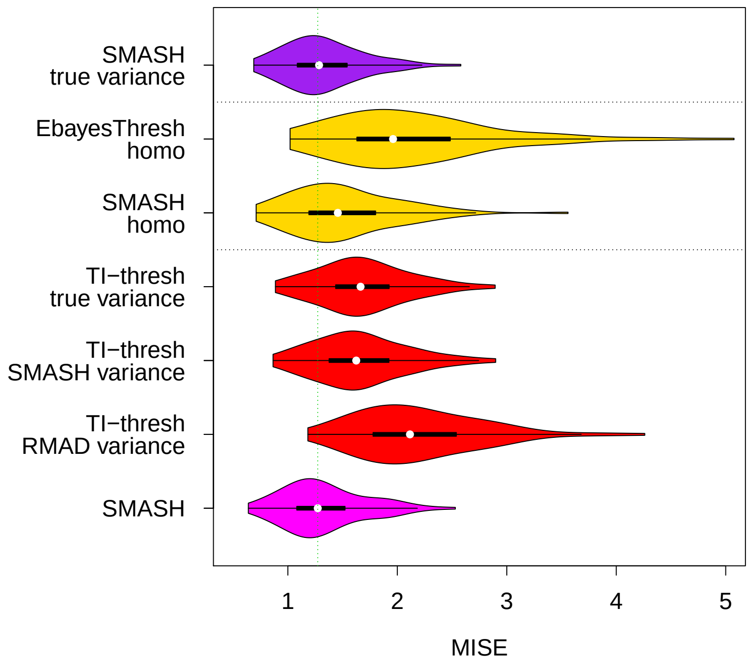
1. We generated $n = 500$ independent (X_i, Y_i) pairs, with $X_i \sim$ Uniform(0,1), and $Y_i|X_i = x_i \sim N(m(x_i), s(x_i)^2)$ where $m(\cdot)$ and $s(\cdot)$ denote the mean and standard deviation functions (Figure 4). We measured performance by the MSE evaluated at 201 equally spaced points on (X_{min}, X_{max}) for both the mean and the standard deviation.
2. We generated $n = 1024$ independent (X_i, Y_i) pairs, with the X_i's (deterministically) equally spaced on (0,1), and $Y_i|X_i$ as above. Performance is measured by MSE evaluated at the 1024 X_i's for both the mean and the standard deviation.



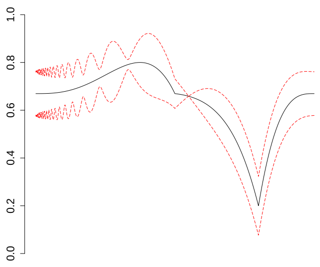
(a)



(b)



(c)



(d)

Figure 3: Comparison of wavelet-based methods for mean estimation with heteroskedastic Gaussian errors. Figures on the left show violin plots of MISEs for various methods on two sets of mean-variance functions: "Spikes" mean function with "Clipped Blocks" variance function, and "Corner" mean function with "Doppler" variance function. Smaller MISE implies better performance; dashed green line indicates the median MISE for SMASH. Figures on the right plot the mean functions ±2 standard deviations.

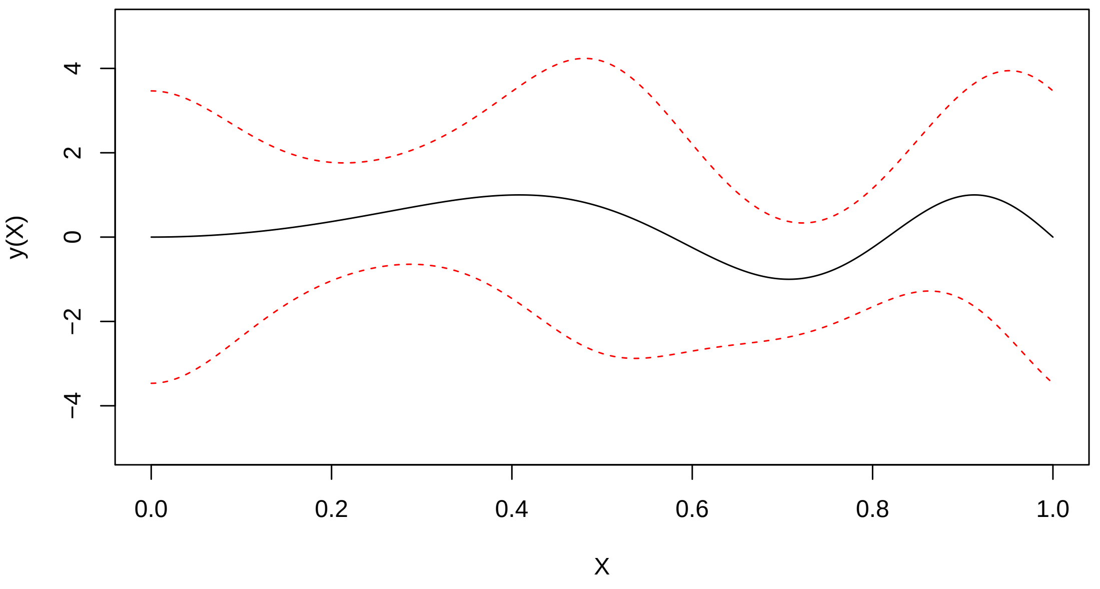


Figure 4: Shows the mean function $m(x)$ ± 2 standard deviations $s(x)$ used in the simulation study comparing SMASH against MFVB. These functions correspond to mean and standard deviation functions (A) from [?].

The first scenario presents some issues for SMASH because the number of data points is not a power of two, nor are the points equally-spaced. To deal with the first issue, following standard ideas in the wavelet literature, we first mirrored the data about the right edge and extract the first $2^{\lfloor \log_2(2n) \rfloor}$ sample points, so that the number of data points in the new "dataset" is a power of two, and the mean curve is continuous at the right edge. To further ensure that the input to SMASH is periodic, we reflected the new dataset about the right edge and used this as the final input. To deal with the second issue we follow the common practice of treating the observations as if they are evenly spaced (see [?] for discussion). To estimate the original mean and variance functions, we extract the first n points from the SMASH estimated mean and variance. To evaluate MSE at the 201 equally spaced points (Scenario 1) we use simple linear interpolation between the estimated points.

Table 1 shows mean MSEs over 100 independent runs for each scenario. Despite the fact that these simulation scenarios – particularly Scenario 1 – seem better suited to MFVB than SMASH, SMASH performs comparably or better than MFVB for both mean and variance estimation in both Scenarios.

4.2.3 Poisson Data

To assess SMASH on Poisson data we simulated data from several different test functions from [?], [?] and [?]. We varied the minimum and maximum intensity of each

	Scenario 1		Scenario 2	
	MSE (for mean)	MSE (for sd)	MSE (for mean)	MSE (for sd)
MFVB	0.0330	0.0199	0.0172	0.0085
SMASH	0.0334	0.0187	0.0158	0.0065

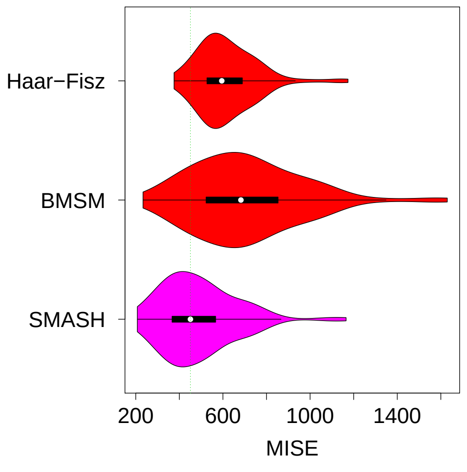
Table 1: Comparison of accuracy (MSE) of SMASH and MFVB for two simulation scenarios. True mean and sd functions are shown in **??**. In Scenario 1 the data are not equally spaced and not a power of 2; here SMASH is comparable to MFVB in mean estimation and more accurate for sd estimation. In Scenario 2 the data are equally spaced and a power of 2; here SMASH outperforms MFVB in both mean and sd estimation.

test function, using (min,max) intensities of (0.01,3), (1/8,8) and (1/128,128). For each test function and intensity level we simulated 100 datasets, each with $n = 1024$ data points. We focus on results for the first two intensity settings, which have smaller average intensity. These settings produce smaller average counts, making them more challenging, and also more representative of the kinds of genomic application that we consider below. Complete results are included in Supplementary Materials.

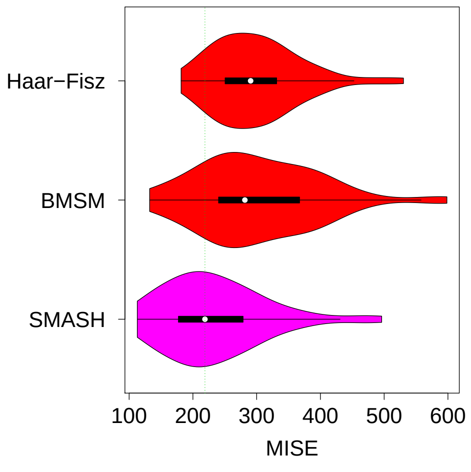
We compared SMASH with eight other methods, but we focus here on the comparisons with the best-performing other methods, which are Haar-Fisz (HF) ([**?**]) and BMSM ([**?**]). The latter, like SMASH, is an Empirical Bayes method, but with a less flexible prior distribution on the multi-scale coefficients. The HF method involves first performing a transformation on the Poisson counts, and applying Gaussian wavelet methods to the transformed data. There are many choices for Gaussian wavelet methods, and the performance depends on these choices (and with different choices being optimal for different data sets). The settings we used here are documented in Supplementary Information, and were chosen by us to optimize (average) performance through moderately extensive experimentation on a range of simulations.

In summary, SMASH outperformed both HF and BMSM in the majority of simulations, with the gain in accuracy being strongest for the more challenging lower-intensity scenarios. A typical result is shown in Figure 5, with complete results in Supplementary Information (?). Among the other two methods, BMSM tends to be the more consistent performer. HF, with the settings we used here, performs quite variably, being worse than the other two in most scenarios, but occassionally performing the best (specifically for the Angles, Bursts and Spikes test functions, with (min,max) intensity (1/128,128)). As noted above, the HF transform can be used with many settings, so the results here should be viewed as a guide to the performance that can be achieved in practice.

One disadvantage of the HF transform is that, to achieve translation invariance (TI), the transform has to be done explicitly for each shift of the data: the tricks usually used to do this efficiently [**?**] do not work here. Thus, making HF fully translation invariant increases computation by a factor of T, rather than the factor of $\log(T)$ for the other methods. Here we follow advice in [] to reduce the computational burden by averaging over 50 shifts of the data rather than T. Even so, HF was substantially slower than the other methods. A direct comparison of computational efficiency between SMASH and BMSM is difficult, as they are coded in different programming



(a)



(b)

Figure 5: Comparison of methods for denoising Poisson data (for the "Bursts" test function). The violin plots show distributions of MISE for each method over 100 datsets, with smaller values indicating better performance. The dashed green line indicates the median MISE for SMASH. Panel a) corresponds to a (min,max) intensity of (0.01,3), and b) corresponds to a (min,max) intensity of (1/8,8).

environments. Nevertheless, similarities between the two methods suggest that they should have similar computational cost. Both SMASH and BMSM took, typically, less than a second per dataset in our simulations.

5 Illustrative applications

5.1 Motorcycle Acceleration Data

As a further illustration of the heteroskedastic Gaussian version of SMASH, we apply it to the motorcycle acceleration dataset from [?]. The data consist of 133 observations measuring head acceleration in a simulated motorcycle accident that is used to test crash helmets. The dependent variable is *acceleration* (in *g*), and the independent variable is *time* (in *ms*). To deal with repeated measurements, we take the median of the measurements for acceleration for any given *time* value. As in Section 4.2.2 we treat the data as if they are equally spaced although they are not. The fitted mean and variance curves (Figure 6) provide a visually appealing fit to the data, and were achieved without hand tuning of any parameters. This contrasts with results in [?] – which also uses a wavelet-based approach for heteroskedastic variance, but accounts for the unequal spacing of the data – which required the *ad hoc* removal of high-resolution wavelet coefficients to produce a visual appealing fit.

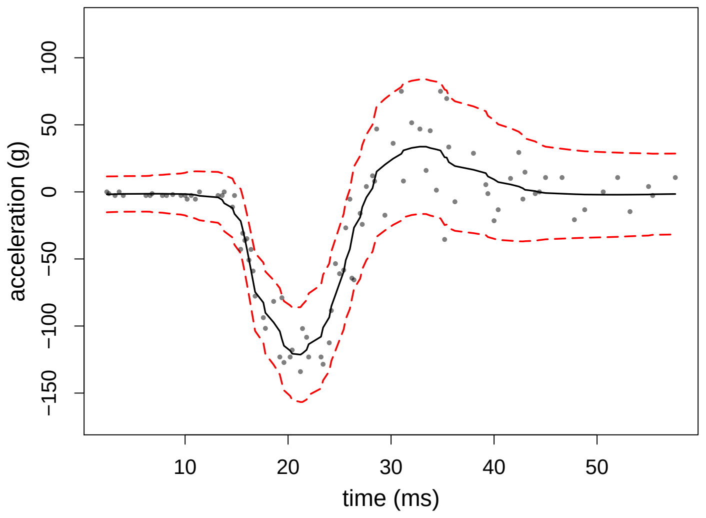


Figure 6: Results from fitting SMASH to the motorcycle acceleration data discussed in [**?**]. The figure shows the estimate mean curve (solid black line) with ± 2 the estimated standard deviation curve (dashed red line).

5.2 ChIP-Seq Data

To further illustrate our Poisson de-noising procedure we applied it to data from genomic sequencing. Specifically we applied SMASH to chromatin immunoprecipitation sequencing (ChIP-seq) data for the transcription factor *YY1* in cell line GM12878 collected by the ENCODE (**Enc**yclopedia **O**f **D**NA **E**lements) project. The data (e.g. Figure 7a) consist of counts of sequencing reads mapping to each location in genome. These can be treated as arising from an inhomogeneous Poisson process, whose intensity at base b is related to the strength of the binding of the transcription factor near b. Because binding tends to be quite localized, the intensity is low on average (the vast majority of counts are 0), but has a small number of intense "peaks". Identifying these peaks can help discover regions where binding occurs, which is an important component of understanding gene regulation. Consequently there are many methods published for "peak detection" in ChIP-Seq data [**?**].

Our goal here is to outline how Poisson de-noising could provide an alternative approach to the analysis of ChIP-seq data. The idea is simply to estimate the underlying intensity function, and then identify "peaks" as regions where the estimated intensity exceeds some threshold.

To illustrate this idea we applied SMASH to ChIP-seq data from a region of length $2^{15}(\approx 33k)$ basepairs from chromosome 1 (hg18 (or 19?) chr1:880001-1011072). The estimated intensity is shown in Figure 7b, overlaid on peaks called by the popular peak calling software MACS []. We see that the locations with the strongest SMASH intensity estimates corresponds to peaks found by MACS. However, the intensity estimate also suggests the presence of several additional weaker peaks not identified by MACS.

The reliable calling of peaks in ChIP-seq data is a multi-faceted problem, and a full assessment lies outside the scope of this paper. Nonetheless, we believe that these illustrative results suggest that this approach could be worth pursuing further. One nice feature of the multi-scale poisson approach is that it deals well with a range of intensity functions, and could perform well even in settings where peaks are broad and/or not especially well defined. In contrast, the performance of different peak-finding algorithms is often reported to be quite sensitive to the "kinds" of peak that are present, meaning that an algorithm that performs well in one setting may perform poorly in another.

6 Discussion

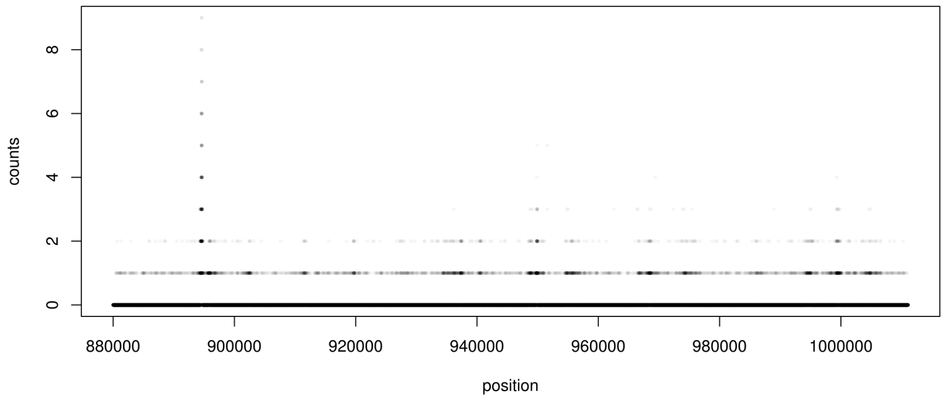
We have adaptive shrinkage method ASH; while it was originally developed in the setting of FDR control for multiple comparisons, we have illustrated its usage as part of two wavelet denoising techniques. Both applications discussed in this paper relax the standard assumption of i.i.d. Gaussian noise, and are thus challenging tasks. Through these applications we are able to demonstrate the flexibility and accuracy of the shrinkage method, revealing its potential in many other applications.

First, our method provides a measure of uncertainty (through both the credible band for the mean curve as well as the variance curve itself), which is often of interest in many applications.

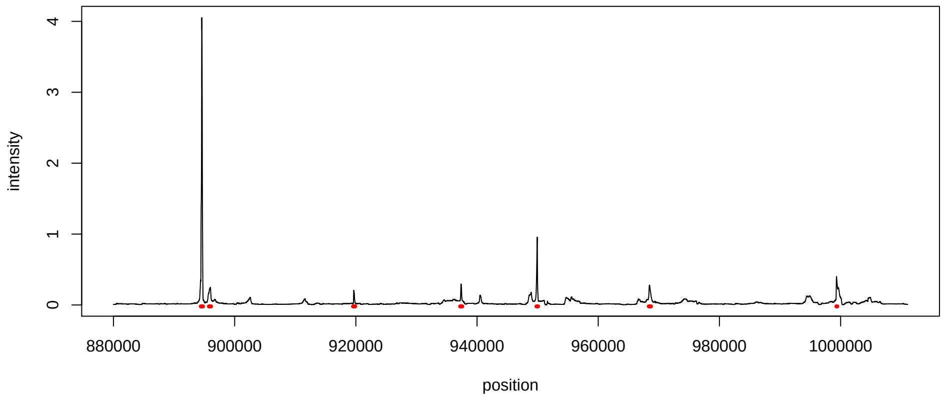
In both the aforementioned applications, our software allows users to easily obtain point estimates for the mean function as well as their approximate posterior variances as a measure of uncertainty. In addition, the variance function can also be estimated, which would provide frequentist confidence intervals for other forms of mean estimation. To the best of our knowledge, there is no readily available software in the wavelet literature that implements joint mean and variance estimation. Simulations have also confirmed that our method is relatively robust to simple forms of autocorrelation between the errors (details needed). In the case of Poisson regression, we improved upon the conjugate Beta priors in [**?**] by using ASH as a shrinkage procedure, which allows for more flexibility and precision. In both the applications, one further advantage of both our methods is that there is no tuning parameter other than the type of wavelet basis used. On the other hand, the primary resolution level in almost all of the other wavelet-based methods actually affects their performance in varying degrees, depending on the underlying mean and/or variance function. Hence, our fully adaptive procedure allows users to easily apply it to any given dataset, depending on the type of noise.

potential to use likelihood to assess wavelet basis?

We have also demonstrated through numerical studies that our methods mostly outperform their respective counterparts from the standard wavelet literature, in terms of pointwise accuracy (MSE in this case). Furthermore, the simplicity of the approximated Gaussian likelihoods as well as the conjugacy of the mixture Gaussian priors imply that our methods are computationally fast, since the posteriors can be computed analytically. In the Gaussian case, simulation results demonstrated



(a)



(b)

Figure 7: Illustration of how SMASH could be used to identify peaks in ChIP-seq data. Panel (a) shows counts (summed across two replicate experiments). Due to over-plotting, darker regions of the plot correspond to higher concentrations of data points. Panel (b) shows the estimated intensity function from SMASH (black solid line) and location of peaks called by MACS (red markers beneath the estimated intensity).

that our method is competitive with standard wavelet methods in the case of i.i.d. errors (without explicitly assuming thus), whilst maintaining superior accuracy when heteroskedastic errors are present. Unfortunately, the lack of readily available software (except for MFVB, as described in [?]) for variance estimation made it difficult to assess the performance of our method in that context. On the other hand, we were able to compare our method to some of the more popular denoising techniques in the Poisson case. Specifically, we have improved upon the conjugate Beta priors used in conjunction with the binomial likelihoods [?]) by using ASH as the shrinkage procedure, which allows for more flexibility and accuracy. This is particularly evident when the mean intensity is low, as is common in many high-throughput genomic sequencing datasets. Our method is also much faster and comparable in accuracy to the popular Haar-Fisz algorithm. Unfortunately, we were not able to directly compare the computational efficiency of our method to many other methods due to differences in the programming software involved.

Although we focused on one-dimensional univariate denoising, our methods could be extended to more complex scenarios. In the one-dimensional domain, our methods could be used in conjunction with multiple samples, otherwise known as regression analysis of functional data (see [?]). Instead of dealing with a vector of observations, we perform regression analysis on a matrix of observations, each row of which encapsulates a sample with temporally or spatially structured data points. While [?] proposed a way to solve a generic regression model, they implicitly assumed the same variance structure for each sample in the same group or category. Our work in the Gaussian case potentially allows for differing variance structures amongst all the samples, thereby relaxing their assumptions. In the simplest case, we could obtain spatially structured differences between groups by including a single covariate that categorizes each sample. In particular, the Poisson model is extremely useful for discovering regions in sequencing reads where structured differences are present between say, various cell lines, as per our sequencing example in the previous section. The Gaussian model could potentially be used in...(??).

With some work, our methods could also be extended to higher dimensions, where a wider range of applications is possible. For example, we could attempt a straight extension to the two dimensional case for both the Gaussian and the Poisson cases as described in [?]. However, recent research in image denoising problems has shown that smooth curves present in many images such as photographs might render wavelet transformations undesirable []. We could thus incorporate ASH into other types of transformations such as curvelets [], which would be a potential direction for future work.

7 Reference

Appendix A

Variance estimation for Gaussian denoising

With $\boldsymbol{Z}$ as defined in (10), we apply the wavelet transform W to $\boldsymbol{Z}^2$, and obtain the wavelet coefficients $\boldsymbol{\delta} = W\boldsymbol{Z}^2$. Note that $\mathbb{E}(\boldsymbol{\delta}) = (\boldsymbol{\gamma})$, where $\boldsymbol{\gamma} = W\boldsymbol{\sigma}^2$. As with (**??**), we treat the likelihood for $\boldsymbol{\gamma}$ as if it were independent, resulting in

$$L(\boldsymbol{\gamma}|\boldsymbol{\delta}) = \prod_{j=0}^{J}\prod_{k=0}^{T-1} P(\delta_{jk}|\gamma_{jk}) \tag{18}$$

However, the likelihoods $L(\gamma_{jk}|\delta_{jk})$ are not normal, and have no simple closed form expressions. As such, we approximate the likelihood by a normal likelihood through matching the moments of a normal distribution to the distribution $P(\delta_{jk}|\gamma_{jk})$ i.e.

$$P(\delta_{jk}|\gamma_{jk}) \approx N(\gamma_{jk}, \hat{\mathbb{V}}(\delta_{jk})) \tag{19}$$

so that

$$L(\gamma_{jk}|\delta_{jk}) \approx \phi(\delta_{jk}; \gamma_{jk}, \mathbb{V}(\delta_{jk})) \tag{20}$$

where ϕ is the normal density function, and $\mathbb{V}(\delta_{jk})$ is the variance of the detail coefficients. Since these variances are unknown, we estimate them from the data and then proceed to treat them as known. More specifically, since $Z_t \sim N(0, \sigma_t^2)$, we have that

$$\begin{aligned} & \mathbb{E}(Z_t^4) \approx 3\sigma_t^4 \\ \Rightarrow \quad & \mathbb{V}(Z_t^2) \approx 2\sigma_t^4 \end{aligned} \tag{21}$$

and so we simply use $\frac{2}{3}Z_t^4$ as an unbiased estimator for $\mathbb{V}(Z_t^2)$. It then follows that $\hat{\mathbb{V}}(\delta_{jk})$ is given by $\sum_{l=1}^{n} \frac{2}{3}Z_l^4 W_{jk,l}^2$, and is unbiased for $\mathbb{V}(\delta_{jk})$. These will be the inputs to ASH, which then produces shrunk estimates in the form of posterior means for the corresponding parameters. Although this works well in most cases, there are variance functions for which the above procedure tends to overshrink the detail coefficients at the finer levels. This is likely because the distribution of the wavelet coefficients are extremely skewed, especially when the true coefficients are large (at coarser levels the distributions are much less skewed since we are dealing a linear combination of a large number of data points). One way around this issue is to employ a procedure that jointly shrinks the coefficients $\boldsymbol{\gamma}$ and their variance estimates (see JASH). The final estimate of the variance function is obtained from the posterior means via the average basis inverse across all the shifts.

Appendix B

Poisson denoising

First summarize the data in a recursive manner:

$$Y_{Jk} \equiv Y_k \tag{22}$$

for $k = 1, ..., n$, and

$$Y_{jk} = Y_{j+1,2k} + Y_{j+1,2k+1} \tag{23}$$

for resolution $j = 0, ..., J-1$ and location $k = 0, ..., 2^j - 1$. Hence, we are summing more blocks of observations as we move to coarser levels.

This recursive scheme leads to:

$$Y_{jk} = \sum_{l=k2^{J-j}+1}^{(k+1)2^{J-j}} Y_l \tag{24}$$

for $j = 0, ..., J$ and $k = 0, ..., 2^j - 1$.

Further define the following:

$$\lambda_{Jk} \equiv \lambda_k \tag{25}$$

for $k = 1, ..., n$, and

$$\lambda_{jk} = \lambda_{j+1,2k} + \lambda_{j+1,2k+1} \tag{26}$$

for $j = 0, ..., J-1$ and $k = 0, ..., 2^j - 1$. Furthermore, define

$$\alpha_{jk} = \log(\lambda_{j+1,2k}) - \log(\lambda_{j+1,2k+1}) \tag{27}$$

$$\tag{28}$$

for $s = 0, ..., J-1$ and $l = 0, ..., 2^j - 1$. The α's defined this way is extremely similar to the (true) Haar wavelet coefficients, which forms the basis of our approach. Using this recursive representation, we can see that the likelihood for $\boldsymbol{\alpha}$ factorizes into a product of likelihoods, where $\boldsymbol{\alpha}$ is the vector of all the α_{sl}'s. To be specific, we have

$$\begin{aligned} L(\boldsymbol{\alpha}|\mathbf{Y}) &= P(\mathbf{Y}|\boldsymbol{\alpha}) & (29) \\ &= P(Y_{0,0}|\lambda_{0,0}) \prod_{j=0}^{J-1} \prod_{k=0}^{2^j-1} P(Y_{j+1,2k}|Y_{j,k}, \alpha_{j,k}) & (30) \\ &= L(\lambda_{0,0}|Y_{0,0}) \prod_{j=0}^{J-1} \prod_{k=0}^{2^j-1} L(\alpha_{j,k}|Y_{j+1,2k}, Y_{j,k}) & (31) \end{aligned}$$

where the factorization is due to the recursive definition above. Note that $Y_{00}|\lambda_{00} \sim \text{Pois}(\lambda_{00})$. For any given j, k, Y_{jk} is a sum of two independent Poisson random variables, and is itself a Poisson random variable. Hence

$$Y_{j+1,2k}|Y_{jk}, \alpha_{jk} \sim \text{Bin}(Y_{jk}, \frac{1}{1+e^{-\alpha_{jk}}} \equiv \frac{\lambda_{j+1,2k}}{\lambda_{jk}})$$

B.1 Estimates and standard errors for α_j

Each α_j is a ratio of the form $\log(\mu_{a:b}/\mu_{c:d})$ whose maximum likelihood estimate (mle) is $\log(Y_{a:b}/Y_{c:d})$. The main challenge here is that the mle is not well behaved when either the numerator or denominator of $Y_{a:b}/Y_{c:d}$ is 0. To deal with this, when either is 0 we use Tukey's modification [?]. Specifically, letting S denote $Y_{a:b}$ and F denote $Y_{c:d}$ (corresponding to thinking of these as successes and failures in a binomial experiment, given $Y_{a:b} + Y_{c:d}$), we use

$$\hat{\alpha} = \begin{cases} \log\{(S+0.5)/(F+0.5)\} - 0.5 & S = 0 \\ \log\{S/F\} & S = 1, 2, ..., N-1 \\ \log\{(S+0.5)/(F+0.5)\} + 0.5 & S = N \end{cases} \tag{32}$$

$$se(\hat{\alpha}) = \sqrt{V^*(\hat{\alpha}) - \frac{1}{2}\{V_3(\hat{\alpha})\}^2 \left\{V_3(\hat{\alpha}) - \frac{4}{N}\right\}} \tag{33}$$

where

$$V_3(\hat{\alpha}) = \frac{N+1}{N}\left(\frac{1}{S+1} + \frac{1}{F+1}\right) \qquad S = 0, ..., N \tag{34}$$

$$V^*(\hat{\alpha}) = V_3(\hat{\alpha})\left\{1 - \frac{2}{N} + \frac{V_3(\hat{\alpha})}{2}\right\} \tag{35}$$

The square of the standard error in (33) corresponds to V^{**} from p. 182 of [?], and is chosen because it is less biased for the true variance of $\hat{\alpha}$ (when N is small) as compared to the asymptotic variance of the MLE (see [?]). The other two variance estimators from [?], V_1^{++} and V^{++}, were also considered in simulations and gave similar results, but V^{**} was chosen for its simple form.

B.2 Signal reconstruction

Given the posterior means and variances of the α's from ASH, the first step to reconstructing the signal is to find the posterior means of $p_{jk} := \frac{\lambda_{j+1,2k}}{\lambda_{jk}}$ and $q_{jk} := \frac{\lambda_{j+1,2k+1}}{\lambda_{jk}}$ (for $j = 0, ..., J-1$ and $k = 0, ..., 2^j - 1$). Specifically, for each j and k, we wish to find

$$E(p_{jk}) \equiv E\left(\frac{e^{\alpha_{jk}}}{1+e^{\alpha_{jk}}}\right) \tag{36}$$

$$E(q_{jk}) \equiv E\left(\frac{e^{-\alpha_{jk}}}{1+e^{-\alpha_{jk}}}\right) \tag{37}$$

Given that we already have the posterior expectations and variances for α_{jk}, we can approximate (36)-(37) using the Delta method. First, define

$$ff(x) = \frac{e^x}{1+e^x} \tag{38}$$

and consider the Taylor expansion of $ff(x)$ about $ff(E(x))$:

$$ff(x) \approx ff(E(x)) + ff'(E(x))(x - E(x)) + \frac{ff''(E(x))}{2}(x - E(x))^2 \tag{39}$$

where

$$ff'(x) = \frac{e^x}{(1+e^x)^2} \tag{40}$$

$$ff''(x) = \frac{e^x(1-e^x)}{(1+e^x)^3} \tag{41}$$

It is easy to see that

$$E(p_{jk}) \approx ff(E(\alpha_{jk})) + \frac{ff''(E(\alpha_{jk}))}{2}Var(\alpha_{jk}) \tag{42}$$

$$E(q_{jk}) \approx ff(-E(\alpha_{jk})) + \frac{ff''(-E(\alpha_{jk}))}{2}Var(\alpha_{jk}) \tag{43}$$

noting that we have already computed $E(\alpha)$ and $Var(\alpha)$.

Finally, we can easily back-transform to construct an estimated signal, by noting that we can express λ_t as a product of the p's and q's for any $i = 1, 2, ..., n$. Specifically, let $\{c_1, ..., c_J\}$ be the binary representation of $i - 1$, and $d_m = \sum_{j=1}^{m} c_j 2^{m-j}$ for $j = 1, ..., J-1$. We then have

$$\lambda_k = \lambda_{00} p_{00}^{1-c_1} p_{1,d_1}^{1-c_2} ... p_{J-1,d_{J-1}}^{1-c_J} q_{00}^{c_1} q_{1,d_1}^{c_2} ... q_{J-1,d_{J-1}}^{c_J} \tag{44}$$

where we usually estimate λ_{00} by $\sum_l Y_l$ (see Kolaczyk (1999)). Using the independence of the p's and q's from different scales, we have:

$$\begin{aligned} E(\lambda_t) = \lambda_{00} E(p_{00})^{1-c_1} E(p_{1,d_1})^{1-c_2} ... E(p_{J-1,d_{J-1}})^{1-c_J} \\ E(q_{00})^{c_1} E(q_{1,d_1})^{c_2} ... E(q_{J-1,d_{J-1}})^{c_J} \end{aligned} \tag{45}$$

As an additional step, we can also construct a credible band around the signal using the posterior variances for inference purposes. From (44) we have the following:

$$\begin{aligned} E(\lambda_t^2) = \lambda_{00}^2 E(p_{00}^2)^{1-c_1} E(p_{1,d_1}^2)^{1-c_2} ... E(p_{J-1,d_{J-1}}^2)^{1-c_J} \\ E(q_{00}^2)^{c_1} E(q_{1,d_1}^2)^{c_2} ... E(q_{J-1,d_{J-1}}^2)^{c_J} \end{aligned} \tag{46}$$

To compute the terms in (46), we again make use of the Delta method (with $ff(x) = (\frac{e^x}{1+e^x})^2$) to obtain:

$$\begin{aligned} E(p_{jk}^2) \approx \left(ff(E(\alpha_{jk})) + \tfrac{ff''(E(\alpha_{jk}))}{2}Var(\alpha_{jk})\right)^2 + \\ \{ff'(E(\alpha_{jk}))\}^2 Var(\alpha_{jk}) \end{aligned} \tag{47}$$

$$\begin{aligned} E(q_{jk}^2) \approx \left(ff(-E(\alpha_{jk})) + \tfrac{ff''(-E(\alpha_{jk}))}{2}Var(\alpha_{jk})\right)^2 + \\ \{ff'(E(-\alpha_{jk}))\}^2 Var(\alpha_{jk}) \end{aligned} \tag{48}$$

Finally we combine (45) and (46) to find $Var(\lambda_k)$, which allows us to construct credible intervals.

Note here that for the reconstructed signal to possess the property of shift invariance (see Coifman & Donoho (1995)), the α's are extracted from a so-called translation invariant (TI) table (see [**?**], and [**?**]) rather than as described above. The idea remains the same however, and we can simply think of the extra α's as being defined similarly as the original α's, albeit from a shifted version of the original data points. To be more specific, the TI table contains the α_{jk} for all circulant shifts of the signal. Here we define the t-th shift of the signal $\boldsymbol{Y}$, denoted by $\boldsymbol{Y}^{(t)}$, to be created from $\boldsymbol{Y}$ itself by moving the first $n-t$ elements of $\boldsymbol{Y}$ t positions to the right and then putting the last t elements of $\boldsymbol{Y}$ in the first t locations. Using this table, we are essentially computing the posterior expectations in (45)-(46) by averaging over all posterior expectations for every shift of the original signal ie.

$$\frac{1}{n}\sum_{t=1}^{n} E(\hat{\lambda}_k^{(t)}) \tag{49}$$

which is an approximation to the true quantity we wish to compute, given by

$$E(\hat{\lambda}_k) = \sum_{t=1}^{n} E(\hat{\lambda}_k^{(t)}) P(t\text{-th shift}) \tag{50}$$