2015

New Results in ell_1 Penalized Regression

Edward A. Roualdes
University of Kentucky, edward.roualdes@uky.edu

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained needed written permission statement(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine) which will be submitted to UKnowledge as Additional File.

I hereby grant to The University of Kentucky and its agents the irrevocable, non-exclusive, and royalty-free license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless an embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's thesis including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Edward A. Roualdes, Student

Dr. David M. Allen, Major Professor

Dr. Constane L. Wood, Director of Graduate Studies

NEW RESULTS IN ℓ_1 PENALIZED REGRESSION

DISSERTATION

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy in the
College of Arts and Sciences
at the University of Kentucky

By

Edward A. Roualdes

Lexington, Kentucky

Directors:
Dr. David M. Allen, Emeritus Professor of Statistics
Dr. Constance L. Wood, Associate Professor of Statistics

Lexington, Kentucky

ABSTRACT OF DISSERTATION

NEW RESULTS IN ℓ_1 PENALIZED REGRESSION

Here we consider penalized regression methods, and extend on the results surrounding the ℓ_1 norm penalty. We address a more recent development that generalizes previous methods by penalizing a linear transformation of the coefficients of interest instead of penalizing just the coefficients themselves. We introduce an approximate algorithm to fit this generalization and a fully Bayesian hierarchical model that is a direct analogue of the frequentist version. A number of benefits are derived from the Bayesian persepective; most notably choice of the tuning parameter and natural means to estimate the variation of estimates – a notoriously difficult task for the frequentist formulation. We then introduce Bayesian trend filtering which exemplifies the benefits of our Bayesian version. Bayesian trend filtering is shown to be an empirically strong technique for fitting univariate, nonparametric regression. Through a simulation study, we show that Bayesian trend filtering reduces prediction error and attains more accurate coverage probabilities over the frequentist method. We then apply Bayesian trend filtering to real data sets, where our method is quite competitive against a number of other popular nonparametric methods.

KEYWORDS: linear model, penalized regression, Bayesian analysis, Hierarchical Models

EDWARD A. ROUALDES

Student's Signature

JULY 31, 2015

Date

NEW RESULTS IN ℓ_1 PENALIZED REGRESSION

By

Edward A. Roualdes

DAVID M. ALLEN
Director of Dissertation

CONSTANCE L. WOOD
Director of Graduate Studies

JULY 31, 2015
Date

To my mother, Slice.

ACKNOWLEDGEMENTS

Big ups to my committee for their support and for caring about me and my education. Much appreciation goes out to Dr. David M. Allen for his guidance and encouragement to read and be interested in a wide variety of topics. Special thanks goes to Dr. Constance L. Wood for her passion for teaching, support, and honesty.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Linear regression remains amongst the most commonly used techniques of statisticians today. Its popularity is largely due to its modularity. Linear regression is the underlying framework for models that expect the response vector $\mathbf{y}$ to be linear in a set of predictors $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_p]$, but this framework extends much further than linear models. For sufficiently smooth models, linear regression can be used to iteratively minimize an objective function of interest and thus estimate parameters of models not linear in the response $\mathbf{y}$. Modern penalized regression exemplifies these dependencies on linear regression perfectly. As such, we first introduce some key aspects of linear regression. The remainder of the Introduction then quickly discusses a popular generalization of linear regression, cross validation, and the bootstrap. Cross validation and the bootstrap, are commonly used tools when modifications to the simple linear model make further analysis intractable. These techniques will come up again in later chapters.

The remaining chapters are organized chronologically, relative to the development of ideas within the penalized regression literature. Chapter 2 discusses the first of the penalized regression methods where only the coefficient vector β is constrained. Of note is the more recent change from the ℓ_2 norm penalty to the ℓ_1 penalty. We discuss the more important algorithms that solve the minimization problems associated with ℓ_1 penalized regression. The Bayesian interpretation of these methods concludes Chapter 2. Chapter 3 introduces a generalization of the methods in Chapter 2, where a linear transformation of the coefficient vector is now penalized. This generalization requires new minimization techniques. We introduce a faster method that computes an approximate solution to this generalization. To conclude Chapter 3, we develop a fully Bayesian hierarchical model that is a Bayesian analogue to the frequentist generalization of ℓ_1 penalized regression. In certain cases, we find the Bayesian hierarchical model to outperform the frequentist formulation. For example, we discuss a specific case of this method called trend filtering. Trend filtering is a nonparametric, univariate smoothing technique, that when fit under this new Bayesian hierarchical model is found to increase prediction accuracy. Further, the Bayesian model produces empirical coverage probabilities that are much closer to their nominal values. Chapter 5 provides directions for future research.

1.1 Linear Regression

Suppose $\mathbf{y}$ is an n-vector of responses thought to be generated from a stochastic process that consists of two parts, a linear combination $\mathbf{X}\beta$ and an additive error term ϵ such that $\mathbb{E}\epsilon = 0$ and $\mathbb{E}\epsilon\epsilon^t = \sigma^2 I_n$, where I_n denotes the $n \times n$ identity matrix. This linear model is written

$$\mathbf{y} \sim \mathcal{N}(\mathbf{X}\beta, \sigma^2 I_n) \tag{1.1}$$

when the error term ϵ is hypothesized to be Gaussian with mean $\mathbf{X}\beta$ and covariance matrix $\sigma^2 I_n$. The matrix $\mathbf{X} \in \mathbb{R}^{n \times p}$ consists of p covariates $\mathbf{x}_j = (x_{1j}, \ldots, x_{nj})^t$ thought to generate via a linear transformation of coefficients $\beta \in \mathbb{R}^p$ the expected value of the response vector $\mathbf{y}$, namely $\mathbb{E}\mathbf{y} = \mathbf{X}\beta$. We use $x_i = (x_{i1}, \ldots, x_{ip})^t$ to denote the ith measurement of the p covariates. The minimization problem, which produces the least squares estimate of β,

$$\hat{\beta}^{LS} = \arg\min_{\beta} \frac{1}{2} \sum_{i=1}^{n} (y_i - x_i^t \beta)^2 \tag{1.2}$$

is mathematically equivalent to the maximum likelihood estimate $\hat{\beta}$. We will often prefer to write the minimization problem (1.3) using the ℓ_p norm. Define the ℓ_p norm as the map $||\cdot||_p : V \mapsto \mathbb{R}$ from an arbitrary vector space V to the real numbers. Equation (1.3) can now be written using the squared ℓ_2 norm,

$$\hat{\beta}^{LS} = \arg\min_{\beta} \frac{1}{2} ||\mathbf{y} - \mathbf{X}\beta||_2^2. \tag{1.3}$$

Scheffe [1960] provides some well known facts about linear regression. The expected value and variance are

$$\mathbb{E}\hat{\beta}^{LS} = \beta, \quad \mathbb{V}\hat{\beta}^{LS} = \sigma^2(\mathbf{X}^t\mathbf{X})^{-1}.$$

Further, the columns of $\mathbf{X}$ can be any function θ_j of the covariates $\mathbf{x}_j$, such that $x_i^t\beta = \sum_{j=1}^p \beta_j\theta_j(x_{ij})$. Thus, linear refers to the linearity of the coefficient vector β. The Gauss-Markov theorem follows from the above facts. When $\sigma^2 < \infty$, the linear, unbiased estimator $\hat{\beta}^{LS} = (\mathbf{X}^t\mathbf{X})^{-1}\mathbf{X}^t\mathbf{y}$ has minimum variance amongst all estimators of its class. Often $\hat{\beta}^{LS}$ is referred to as the best linear unbiased estimator of β.

Later work produced generalizations of this simple model, replacing the identity function between the expected value of $\mathbf{y}$ and the linear combination $\mathbf{X}\beta$ with new functions [Nelder and Baker, 1972]. Known as generalized linear models (GLM), the linear regression model (1.1) is widely used in much of the sciences. New disciplines are increasingly using the simple minimization problem in Equation (1.3) by finding new models with clever interpretations to their discipline and because researchers continue to push this model up against recent work in convex analysis.

1.2 Generalized Linear Models

Generalized linear models modify the linear regression framework in Equation (1.3) to estimate the expected value of the response vector $\mathbf{y}$ from a monotonic transformation, often called the link function, of the linear combination $\mathbf{X}\beta$. Write $\mu = \mathbb{E}\mathbf{y}$, so that the link function g models

$$g(\mu) = \mathbf{X}\beta, \tag{1.4}$$

where $\mathbf{y}$ follows an exponential family distribution, not necessarily the Gaussian distribution. The generalized linear model is fit via a procedure called iteratively reweighted least squares [Nelder and Baker, 1972; Wood, 2006]. This minimization strategy, as it sounds, weights the standard least squares norm and repeatedly solves for β until some convergence criteria is established. Indexing updates of the estimate of β by k, solve

$$\beta^{(k+1)} = \arg\min_{\beta} \sum_{i=1}^{n} w_i(\beta^{(k)})(y_i - f_i(\beta))^2, \tag{1.5}$$

where w is a diagonal matrix dependent on the kth estimate of β and f is a function of the parameters of interest, β. For generalized linear models the function f is the inverse of the link function, but in a broader sense it does not have to be.

1.3 Cross Validation

Cross validation is a widely used method to estimate expected prediction error, $\mathbb{E}_{\mathbf{y},\mathbf{X}}||\mathbf{y} - \mathbf{X}\beta||_2^2$, with the expectation over the joint distribution of $(\mathbf{y}, \mathbf{X})$. Since this is not a simple task given a fixed sample size, the idea is to cleverly reuse the data by creating K subsets of the data and estimating the prediction error for each subset. The average of the K prediction errors provides an estimate of the expected prediction error.

A particularly elegant explanation is given by Hastie *et al.* [2005]. Let $\kappa : \{1, \ldots, n\} \mapsto \{1, \ldots, K\}$ map each observation in the sample to one of the K subsets. Let $\hat{\beta}^{-\kappa(i)}$ be the estimate of β when the subset corresponding to $\kappa(i)$ is not present in the data set. Then we can express the cross validation estimate of the prediction error as

$$\mathrm{CV}(\hat{\beta}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i^t\hat{\beta}^{-\kappa(i)})^2.$$

This estimate of prediction error varies by choice of K; there exists a trade off between bias and variance. On the one hand, leave one out cross validation, where $K = n$ and $\kappa(i) = i$, is

approximately unbiased, but incurs high variance [Allen, 1971; Hastie *et al.*, 2005]. Conversely, $K = 5$ or $K = 10$, two popular choices, produce smaller variance, but higher bias [Breiman and Spector, 1992; Kohavi and others, 1995; Hastie *et al.*, 2005]. Later, we will index the cross validation estimate of prediction error by another parameter, and use this method to form an educated guess about the indexing parameter.

1.4 Bootstrap

The bootstrap is a popular method to estimate properties of the distribution of a statistic. Specifically, by sampling the data $z_i = (y_i, x_i)$ with replacement, the bootstrap estimates the mean and variance of a statistic T. For example, B resamples of the data z creates "new" data sets z_b for $b = 1, \ldots, B$, each of which yields a statistic $T(z_b)$. The B test statistics then produce an estimate of the expected value and variance of the estimator

$$\widehat{\mathbb{V}}T(z) = \frac{1}{B-1}\sum_{b=1}^{B}(T(z_b) - \bar{T}(z))^2, \text{ with } \bar{T}(z) = \sum_{b=1}^{B} T(z_b).$$

The bootstrap is particularly helpful when the asymptotic distribution of a statistic of interest is analytically intractable. While this is not the case for standard linear regression, the methods presented later are mathematically difficult to work with. In the cases below, the bootstrap replaces intractability with heavy computation. Despite the computational burden, the actual code to produce bootstrap estimates involves little extra work. This method has become popular because often minimal extra effort is required. We will revisit the bootstrap in Chapters 2 and 3.

1.5 Gibbs Sampler

The Gibbs samplers is a popular Markov chain algorithm for drawing samples from arbitrary posterior distributions of interest. It was popularized by Geman and Geman [1984], who attributed their work to the physicist Josiah Willard Gibbs. The idea of the Gibbs sampler is to iteratively sample from alternating conditional distributions given the other parameters contained in the model. Using the notation of Gelman *et al.* [2014], consider a parameter vector $\theta = (\theta_1, \ldots, \theta_d)$, where each θ_j describes individual parameters or subvectors of parameters. At each iteration of the algorithm, the tth sample θ_j^t is drawn from the conditional distribution

$$[\theta_j^t | \theta_{-j}^{t-1}, y],$$

where θ_{-j}^{t-1} denotes all elements of θ except for θ_j. The tth iteration of elements of θ are updated dependent on the previous iterations' values of the parameter vector.

We will use this algorithm for each of the Bayesian hierarchical models discussed below. There we will see that the conditional distributions of interest take on recognizable, analytic forms. Thus, the conditional distributions will be recognizable probability density functions.

Chapter 2

Penalizing Coefficients

Linear regression has been and continues to be a corner stone of applied statistics. Amongst the earliest of developments beyond the standard linear model is the use of, though it was not originally phrased this way, a penalty function. The penalty function proposed by Hoerl and Kennard [1970] was a simple additive ℓ_2 norm onto the minimization problem in Equation (1.3). This additive ℓ_2 norm was first theorized to control for correlations amongst the predictors of interest and, when applied correctly, large reductions in mean squared error are traded for small amounts of bias.

Chapter 2 introduces the first penalized regression model, called ridge regression. Then we discuss the slight change from penalties utilizing the ℓ_2 norm to the ℓ_1 norm in Section 2.2. Though this idea seems simple now, it took nearly two decades for this change in penalty function to find its place in the statistics literature. The ℓ_1 norm penalty function created a whirlwind of literature and debate. Sections 2.4 to 2.8 cover the ways in which statisticians have attempted to match the theory of linear regression to ℓ_1 penalized regression: everything from generalized linear models with ℓ_1 norm penalties to efficient computation and how cross validation and the bootstrap aid analysis beyond estimation of parameters. A Bayesian analogue to penalizing coefficients of a linear model is discussed in Section 2.9.

2.1 Ridge Regression

Often applied statisticians implicitly assume that the covariates are not correlated and hence that $\mathbf{X}^t\mathbf{X}$ is a unit matrix, after standardizing the covariates such that $||\mathbf{x}_j|| = 1$. This, though, is not always the case in the real world. Often in practice, $\mathbf{X}^t\mathbf{X}$ can be arbitrarily far away from a unit matrix, causing $(\mathbf{X}^t\mathbf{X})^{-1}$ to be ill-conditioned. Highly correlated columns of $\mathbf{X}$ create small eigenvalues of $\mathbf{X}^t\mathbf{X}$. As the eigenvalues of $\mathbf{X}^t\mathbf{X}$ shrink, the variance $\mathbb{V}\hat{\beta}^{LS}$ increases. To remedy this Hoerl and Kennard [1970] proposed the ridge regression estimator, where the least squares estimator $\hat{\beta}^{LS}$ is replaced with $\hat{\beta}^R = (\mathbf{X}^t\mathbf{X} + \lambda I_n)^{-1}\mathbf{X}^t\mathbf{y} = \mathcal{R}_\lambda\mathbf{y}$ for $\lambda \geq 0$. The matrix $\mathcal{R}_\lambda$ is the projection matrix associated with the ridge regression estimator. Obviously, when $\lambda = 0$ the least squares estimator is recovered, and large values of λ will dominate the influence of $\mathbf{X}^t\mathbf{X}$. Written as a minimization problem, ridge regression solves

$$\hat{\beta}^R = \arg\min_\beta \frac{1}{2}||y - X\beta||_2^2 + \lambda||\beta||_2^2. \tag{2.1}$$

Since 1970, justification for the ridge regression estimator has expanded. Not only does $\hat{\beta}^R$ improve the conditioning of $\hat{\beta}^{LS}$, it also has interpretation as a simple addition to equation (1.3) to get equation (2.1). Geometrically, this addition shrinks the elements of β towards the unit hypersphere, see Figure (2.2). Because the ridge regression estimator is a linear transformation of the observations, we can easily find the expected value and variance,

$$\mathbb{E}\hat{\beta}^R = (\mathbf{X}^t\mathbf{X} + \lambda I_p)^{-1}\mathbf{X}^t\mathbf{X}\beta, \quad \mathbb{V}\hat{\beta}^R = \sigma^2\mathbf{Z}(\mathbf{X}^t\mathbf{X})^{-1}\mathbf{Z}^t,$$

where $\mathbf{Z} = \mathbf{Z}(\lambda) = [I_p + \lambda(\mathbf{X}^t\mathbf{X})^{-1}]^{-1}$ is dependent on some choice of the penalty parameter λ.

The benefits of the ridge estimator are best seen when comparing its properties to those of the least squares estimator. First, we can define the ridge regression estimator in terms of the least squares estimator, $\hat{\beta}^R = \mathbf{Z}\hat{\beta}^{LS}$, remember that $\mathbf{Z}$ is indexed by λ. Using this, break up the distance between the ridge estimator $\hat{\beta}^R$ and the true coefficient vector β into two terms,

$$\begin{aligned}\mathbb{E}(\hat{\beta}^R - \beta)^t(\hat{\beta}^R - \beta) &= \mathbb{E}[(\hat{\beta}^{LS} - \beta)^t\mathbf{Z}^t\mathbf{Z}(\hat{\beta}^{LS} - \beta)] + (\mathbf{Z}\beta - \beta)^t(\mathbf{Z}\beta - \beta) \\ &= \gamma_1(\lambda) + \gamma_2(\lambda).\end{aligned}$$

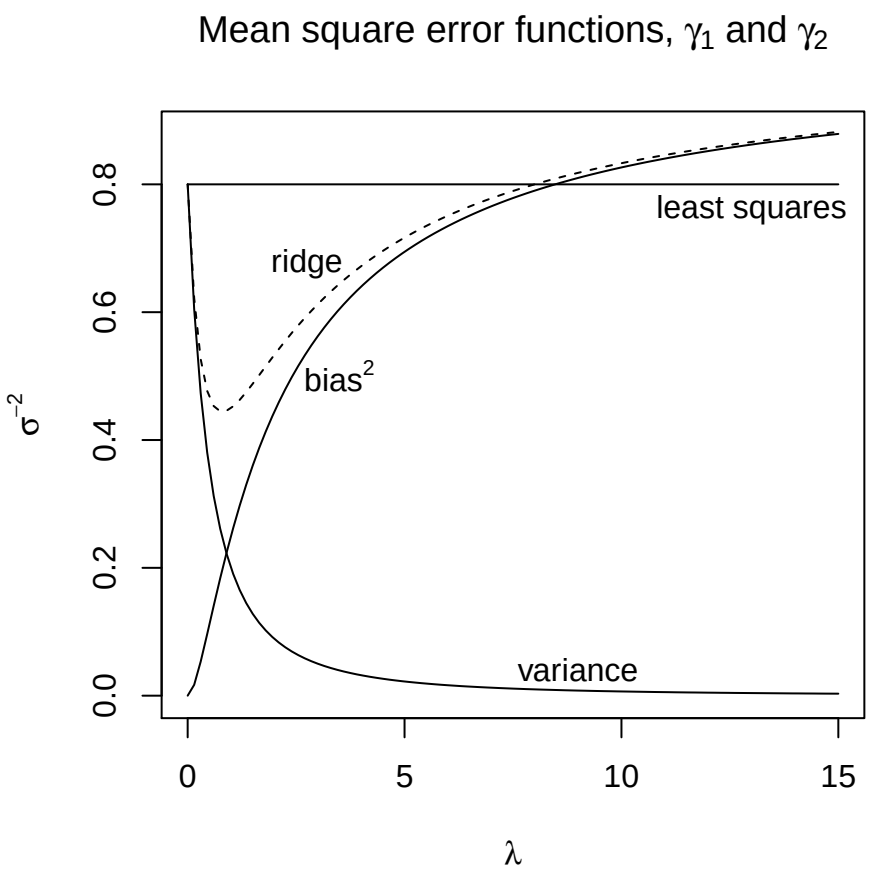

Figure 2.1: Ridge regression mean squared error as a function of λ.

The two terms γ_1 and γ_2, as functions of the penalty parameter λ, can be interpreted as the sum of the variances of the estimators and the squared distance from $\mathbf{Z}\beta$ to β, respectively. Thus, the terms $\gamma_1(\lambda)$ and $\gamma_2(\lambda)$ are akin to the standard variance and square bias decomposition induced by using $\hat{\beta}^R$ instead of $\hat{\beta}^{LS}$. Hoerl and Kennard [1970] go on to show that indeed, as hoped, the shape of γ_1 and γ_2 as functions of λ trade small amounts of squared bias for drastically reduced variance. Figure (2.1) plots the different mean squared error functions discussed in Hoerl and Kennard [1970]. Ridge regression, represented by the dashed line, for some values of λ dips below the least squares's mean squared error, but then as the bias increases with larger values of λ the ridge regression mean squared error overtakes it.

2.2 Lasso

Tibshirani [1996] introduced a slight modification to the penalty term. This model simply replaces the l_2 norm penalty function in ridge regression with the l_1 norm. Dubbed the lasso, the least absolute shrinkage and selection operator has become incredibly popular. The lasso solution $\hat{\beta}^{\ell_1}$ is the minimum vector to the objective function

$$\hat{\beta}^{\ell_1} = \arg\min_{\beta} \frac{1}{2}||\mathbf{y} - \mathbf{X}\beta||_2^2 + \lambda||\beta||_1. \tag{2.2}$$

The l_1 norm carries desirable geometric properties with it. Instead of simply shrinking the elements

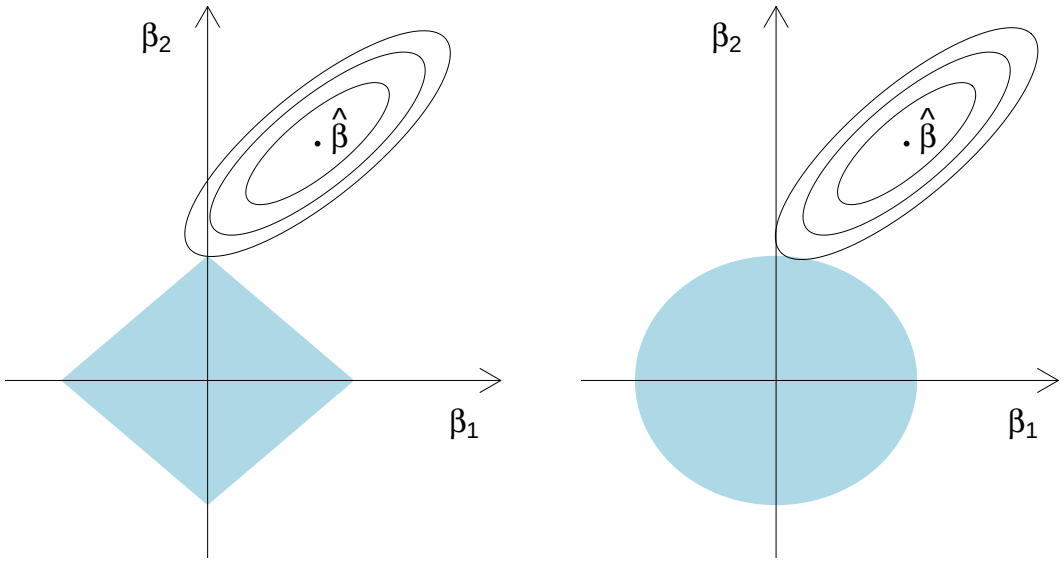

Figure 2.2: The sharp corners of the ℓ_1 penalty, compared to the ℓ_2, allows for some of the coefficients of β to be set identically to zero.

of the covariate vector β towards zero, the sharp contours of the l_1 norm cause some of the elements to be set identically to zero [Tibshirani, 1996]. This is easily seen in Figure (2.2) where the space of $\beta \in \mathbb{R}^2$ subject to each norm is plot. A potential solution, $\hat{\beta}$, and contours of its ellipsoids are plot for each norm. While it is theoretically possible for ridge regression to produce a sparse solution, in the sense that some elements of β are identically zero, it is quite unlikely, and is much more likely in the case of the lasso.

The reduction of β depends on the size of the penalty parameter λ and on the number of observations n relative to the number of parameters p. When $p < n$, lasso estimates of β ranges from $\hat{\beta}^{\ell_1}(\lambda) = \hat{\beta}^{LS}$ to $\hat{\beta}^{\ell_1}(\lambda) = 0$, as λ approaches positive infinity starting from zero. In Section 2.6 we discuss solution paths of lasso estimates indexed by λ in greater detail. On the other hand, the ℓ_1 penalty function provides estimates of β even when $n < p$. In this case, there is no direct comparison to $\hat{\beta}^{LS}$.

Tibshirani [1996] attempted to address the properties of $\hat{\beta}^{\ell_1}$ as an estimator. Accurate estimates of the standard errors are hard to find, because the lasso estimator is a nonlinear and nondifferentiable function of the response vector $\mathbf{y}$. Similar to the ridge regression estimator, it is well known that the lasso is a biased estimator of β [Tibshirani, 1996; Fan and Li, 2001; Breheny and Huang, 2011]. Lasso's bias is most notable for heavily shrinking truly large coefficients towards zero. Calculating the variance of $\hat{\beta}^{\ell_1}$ proves to be even more difficult. Tibshirani [1996] suggested approximating the variance of $\hat{\beta}^{\ell_1}$ with something similar to the variance of ridge regression,

$$\mathbb{V}\hat{\beta}^{\ell_1} = (X^t X + \lambda\Omega^{-1})^{-1} X^t X (X^t X + \lambda\Omega^{-1})^{-1} \hat{\sigma}^2,$$

where $\Omega = \text{diag}(|\hat{\beta}_j^{\ell_1}|)$. Many authors have since attempted to calculate reasonable standard errors for the lasso. Standard errors will be discussed in greater detail in Section 2.8.

The original lasso paper proposed an algorithm to minimize the objective function in Equation (2.2) using quadratic programming techniques. That solution is guaranteed to find the minimizing value of β in $2^p + 1$ iterations. As suggested by Tibshirani [1996], rarely are this many iterations required in application. Still, quite a bit of work has gone into finding more efficient solutions to this problem. We turn to such algorithms next.

2.3 Algorithms to compute the lasso solution

Lars Arguably the most important of the solutions to Equation (2.2) is called least angle regression (lars) [Efron *et al.*, 2004]. Lars, method in its own right, is a generalization of the Forward/Backward stage-wise regression methods, where step sizes are determined algebraically instead of by a predetermined amount so that covariates are added to the model sequentially. Intuitively, the algorithm moves through $\hat{\mu} = \mathbf{X}\hat{\beta}$ space, sequentially adding variables most correlated with the response vector to the "active set" $\mathcal{A}$. Each step moves in an equiangular direction relative to the predictors in $\mathcal{A}$. Each step size in this equiangular direction is stopped when another predictor is added to the set, i.e. when another predictor has greater than or equal correlation, relative to the members of the active set, with the response. Upon each stop a new equiangular direction is calculated, as the direction depends on the elements of the active set.

We summarize the lars algorithm mathematically, by describing movements of the vector $\hat{\mu} = X\hat{\beta} \in \mathbb{R}^n$. Lars, described in Algorithm (2.1), requires two values: i) the equiangular direction $u_{\mathcal{A}}$, and ii) the distance $\hat{\gamma}$ moved in direction $u_{\mathcal{A}}$. We refer the reader to the original paper Efron *et al.* [2004] or a good summary by Khan *et al.* [2007] for calculations of $u_{\mathcal{A}}, \hat{\gamma}$, and $\omega_{\mathcal{A}}$ and suffice ourselves with the intuitive conditions necessary to calculate $u_{\mathcal{A}}$:

1. $u_{\mathcal{A}}$ is a linear combination of the active predictors: $u_{\mathcal{A}} = X_{\mathcal{A}}\omega_{\mathcal{A}}$, where $\omega_{\mathcal{A}}$ is a vector of weights.
2. $u_{\mathcal{A}}$ has unit variance, $n^{-1}u_{\mathcal{A}}^t u_{\mathcal{A}} = 1$.
3. $u_{\mathcal{A}}$ has equal correlation a with each of the active predictors, $n^{-1}X_{\mathcal{A}}^t u_{\mathcal{A}} = a1_{\mathcal{A}}$ such that $1_{\mathcal{A}}$ has length $|\mathcal{A}|$.

The lars algorithm adapts to the lasso problem, by enforcing $\hat{\beta}_j$ to agree in sign with $\hat{c}_j = x_j^t(y - \hat{\mu})$. When $p < n$, lars translates what is otherwise a large (2^p) dimensional quadratic programming problem into an algorithm that has the same computational cost of a least squares fit, $\mathcal{O}(p^3 + np^2)$ [Efron *et al.*, 2004]. When $p > n$, as is often the case in lasso problems, lars has a computational cost of $\mathcal{O}(n^3)$ and terminates with $n - 1$ variables included in the model.

Algorithm 2.1: Least Angle Regression

Input: $(\mathbf{X}, \mathbf{y})$
$\mathcal{A} \leftarrow \emptyset,\ \hat{\mu}_{\mathcal{A}} = \mathbf{X}\hat{\beta} \leftarrow 0,\ \hat{\mu}_{\mathcal{S}} \leftarrow \emptyset$
for $j = 1$ **to** p **do**
  $\mathcal{A} \leftarrow \mathcal{A} \cup \arg\max_i |(x_i^t\mathbf{y} - \hat{\mu})|$
  $\hat{\mu}_{\mathcal{A}} \leftarrow \hat{\mu}_{\mathcal{A}} + \hat{\gamma}u_{\mathcal{A}}$
  $\hat{\mu}_{\mathcal{S}} \cup \hat{\mu}_{\mathcal{A}}$
end for
Output: $\hat{\mu}_{\mathcal{S}}$

Coordinate descent Another strategy to fit the lasso objective function in Equation (2.2) is to use the coordinate descent methods of Luo and Tseng [1992]. Consider a closed, convex function $f : \mathbb{R}^p \mapsto \mathbb{R}$, with variables $x_1, \ldots, x_p$. Denoting by $f^{(r)}$ successive minimizations of f over its domain, the global minimizers is found by repeatedly solving

$$x_i^{(r+1)} = \arg\min_{x_i} f(x_1^{(r)}, \ldots, x_{i-1}^{(r)}, x_i, x_{i+1}^{(r)}, \ldots, x_p^{(r)}). \tag{2.3}$$

Applying this method to the lasso problem is established by noting that

$$\begin{aligned} f(\beta) &= \frac{1}{2}||\mathbf{y} - \mathbf{X}\beta||_2^2 + \lambda||\beta||_1 \\ &= g(\beta) + \sum_{j=1}^{p} h_j(\beta_j) \end{aligned}$$

where g is both differentiable and convex, and the h_j are convex [Friedman *et al.*, 2007; Wu and Lange, 2008; Friedman *et al.*, 2010]. So long as the h_j are separable the coordinate descent algorithm is guaranteed to converge to the optimal solution [Luo and Tseng, 1992]. In fact, this formulation can be applied to many other penalty functions for which the h_j are appropriately separable [Friedman *et al.*, 2007, 2010; Breheny and Huang, 2011].

Because Equation (2.3) is often both analytically tractable and costs very little, coordinate descent methods can be very fast. This is true even if there is no analytic solution to the original minimization problem. One full update step, over all parameters costs $\mathcal{O}(np)$ operations. Thus, if the solution is found in less than p iterations, which it often is, the solution to a penalized regression problem costs less than that of the solution to standard linear regression, $\mathcal{O}(np^2)$ [Breheny and Huang, 2011]. Since many penalized regression problems can be formulated as such, especially when combined with the approximation methods of Hunter and Li [2005], it seems likely that the coordinate descent methods will gain popularity with time.

We briefly list some other penalty functions, for which coordinate descent directly solves the associated penalized regression minimization problem. Some penalty functions retain the convexity of the loss function. Since the objective function of interest is then convex, minimization strategies that use coordinate descent are guaranteed to find the global minimum.

2.4 Extensions

In an effort to improve upon the lasso, authors experimented with a number of other penalty functions. Some of these methods are simply a weighted mixture of the lasso and ridge regression, while other methods seek stronger asymptotic properties. Below we provide a few of the more notable penalty functions and describe how they contributed to the literature. In many cases, solutions derived from differing penalty functions involve the soft-threshold operator of Donoho and Johnstone [1995],

$$S(\hat{\beta}, \gamma) = \text{sign}(\hat{\beta})(|\hat{\beta}| - \gamma)_+, \tag{2.4}$$

which is often written

$$S(\hat{\beta}, \gamma) = \begin{cases} \hat{\beta} - \gamma, & \hat{\beta} > 0 \text{ and } \gamma < |\hat{\beta}| \\ \hat{\beta} + \gamma, & \hat{\beta} < 0 \text{ and } \gamma < |\hat{\beta}| \\ 0, & |\hat{\beta}| \leq \gamma \end{cases}.$$

All of the penalty functions $h : \mathbb{R} \mapsto \mathbb{R}$ listed below can be fit using the coordinate descent algorithm described above, with variations on the soft-threshold operator.

1. *Lasso:* $h(x) = |x|$. Friedman *et al.* [2007] find the solution to Equation (2.2) by using coordinate descent with S in the updating function

$$\beta_j^{(r+1)}(\lambda) \leftarrow S\left(\sum_{i=1}^{n} x_{ij}(y_i - \tilde{y}_i^{(j)}), \lambda\right)$$

   where $\tilde{y}_i^{(j)} = \sum_{l \neq j} x_{il}\beta_l^{(r)}$, to be the partial residual for fitting β_j with the rth estimate $\beta_j^{(r)}$. Because the objective function of the lasso is globally convex, this updating function is guaranteed to converge to the global minimum for some value of λ.

2. *Elastic net:* $h(x) = \lambda_1|x| + \lambda_2 x^2/2$. The elastic net penalty function combines the ℓ_1 and ℓ_2 norms of the lasso and ridge regression [Zou and Hastie, 2005; Kooij, 2007].

$$\beta_j^{(r+1)}(\lambda_1, \lambda_2) \leftarrow \frac{S(\sum_{i=1}^{n} x_{ij}(y_i - \tilde{y}_i^{(j)}), \lambda_1)_+}{1 + \lambda_2}$$

3. *Grouped lasso:* $h_j(\mathbf{x}_j) = \lambda_j ||\mathbf{x}_j||_2$, where $\mathbf{x}_j$ is a grouping of p_j variables, say for dummy variables, and $\lambda_j = \lambda\sqrt{p_j}$ [Yuan and Lin, 2006]. In this case, one uses block updates since only the blocks of p_j parameters $\mathbf{x}_j$ are separable

$$\beta_j^{(r+1)}(\lambda_j) \leftarrow (||S_j||_2 - \lambda_j)_+ \frac{S_j}{||S_j||_2},$$

where $S_j = \mathbf{x}_j^t(\mathbf{y} - \tilde{\mathbf{y}}^{(j)})$ and $\tilde{\mathbf{y}}^{(j)} = \sum_{k \neq j} \mathbf{x}_k \tilde{\beta}_k$.

4. *Smoothly clipped absolute deviation:* $h(x) = (\lambda x)1(x \leq \lambda) + \frac{\gamma\lambda x - 0.5(x^2+\lambda^2)}{\gamma - 1} 1(\lambda < x \leq \gamma\lambda) + \frac{\lambda^2(\gamma^2-1)}{2(\gamma-1)} 1(x > \gamma\lambda)$. The smoothly clipped absolute deviation (SCAD) penalty was designed to eliminate bias encouraged by the standard ℓ_1 norm, which heavily shrinks all, even truly large, coefficients towards zero. The SCAD achieves the oracle property, where asymptotically using this penalty function is as good as knowing which coefficients are truly zero and which are not [Fan and Li, 2001; Breheny and Huang, 2011]. Indexed by two tuning parameters γ and λ, one updates updates the coefficients with

$$\beta_j^{(r+1)}(\lambda, \gamma) \leftarrow \begin{cases} S(\beta_j^{(r)}, \lambda), & |\beta_j^{(r)}| \leq 2\lambda \\ \frac{S(\beta_j^{(r)}, \gamma\lambda/(\gamma-1))}{1 - 1/(\gamma-1)}, & 2\lambda < |\beta_j^{(r)}| \leq \gamma\lambda \\ \beta_j^{(r)}, & |\beta_j^{(r)}| > \gamma\lambda \end{cases}.$$

5. *Minimax concave penalty:* $h(x) = (\lambda x - \frac{x^2}{2\gamma}) 1(x \leq \gamma\lambda) + (\frac{1}{2}\gamma\lambda^2) 1(x > \gamma\lambda)$. The minimax concave penalty (MCP) is the next generation of penalty functions that obtains similar properties to the SCAD, but produces more favorable numerical results [Zhang, 2010; Breheny and Huang, 2011]. MCP is also indexed by two parameters γ and λ. Given values for these tuning parameters, update β with

$$\beta_j^{(r+1)}(\lambda, \gamma) \leftarrow \frac{S(\beta_j^{(r)}|, \lambda) 1(|\beta_j^{(r)}| \leq \gamma\lambda)}{1 - 1/\gamma} + \beta_j^{(r)} 1(|\beta_j^{(r)} > \gamma\lambda).$$

In specific cases some mathematical considerations can drastically decrease the computational costs of these algorithms. Consider the partial residual $y_i - \tilde{y}_i^{(j)}$. Note that, as in Friedman *et al.* [2010],

$$y_i - \tilde{y}_i^{(j)} = y_i - \hat{y}_i + x_{ij}\beta_j = r_i + x_{ij}\beta_j$$

where r_i is the current residual for observation i. Hence,

$$n^{-1} \sum_{i=1}^n x_{ij}(y_i - \tilde{y}_i^{(j)}) = n^{-1} \sum_{i=1}^n x_{ij} r_i + \beta_j$$

because we have standardized the $\mathbf{x}_j$, so that $\mathbf{x}_j^t \mathbf{x}_j = 1$. Next, write

$$\sum_{i=1}^n x_{ij} r_i = \langle \mathbf{x}_j, \mathbf{y} \rangle - \sum_{k:|\tilde{\beta}_k|>0} \langle \mathbf{x}_j, \mathbf{x}_k \rangle \tilde{\beta}_k$$

so that a good number of the calculations can be performed before the algorithm even begins.

2.5 Lassoed GLMs

Friedman *et al.* [2010], and Breheny and Huang [2011] showed how weighted updates, for iterated reweighted least squares algorithms, can be combined with penalized regression methods. For instance, to use weights w_i with the lasso penalty, updates are as follows

$$\beta_j^{(r+1)} \leftarrow \frac{S\left(\sum_{i=1}^n w_i x_{ij}(y_i - \tilde{y}_i^{(j)}), \lambda\right)}{\sum_{i=1}^n w_i x_{ij}^2 + \lambda}.$$

Though the weighting scheme adds some complication to the convexity analysis, the development of penalized generalized linear models greatly expands the diversity of the lasso.

2.6 Solution paths

Consider a generalization of penalized regression written as

$$\hat{\beta}(\lambda) = \arg\min_{\beta} L(y, X\beta) + \lambda P(\beta). \tag{2.5}$$

Under certain conditions on the loss function L and penalty function P, penalized regression produces piecewise linear solution paths. That is, the coefficients of the estimator $\hat{\beta}$ indexed by the penalty parameter λ, are piecewise linear.
Rosset and Zhu [2007] prove two sufficient conditions for piecewise linear solution paths to exist:

1. L is quadratic, or piecewise quadratic in β,
2. P is piecewise linear in β.

Interestingly, this implies that the lasso will produce piecewise linear solution paths while ridge regression will not. Figure 2.3 displays the solution paths of a lasso fit to the diabetes data set of Efron *et al.* [2004]. Since we are not interested in the data per se, but just the solution paths, a description of the data is conspicuously missing. As the ℓ_1 norm penalty increases, caused by decreases in the value of λ, the coefficients grow from 0 to their greatest value – recall that their greatest value depends on the relation between the number of observations n and the number of parameters p.

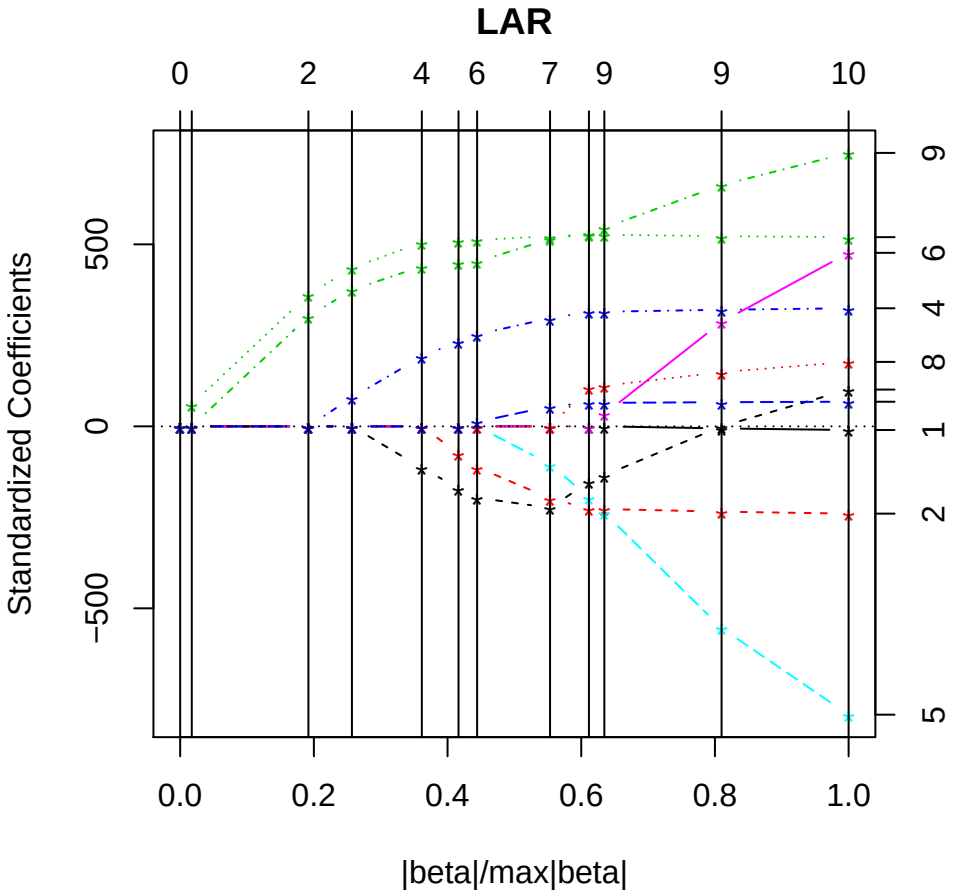

Figure 2.3: Solution paths of a lasso fit to the diabetes data.

The lars algorithm finds an exact piecewise linear solution path for the lasso. However, with other more complex penalty functions, even if they meet the conditions stated by Rosset and Zhu [2007] no such algorithm yet exists to find an exact solution path. In these cases a different strategy is recommended by Friedman *et al.* [2010]. To find a pathwise solution across a range of values of λs, start at $n\lambda_{\max} = \max_l |\langle x_l, y\rangle|$. Then with $\lambda_{\min} = \epsilon\lambda_{\max}$, move from $\lambda_{\max} \to \lambda_{\min}$ on the log scale, in K steps. Default values are commonly chosen to be $\epsilon = 0.001$ and $K = 100$. Increasingly, to make the penalized methods more comparable, ridge regression is being fit via a similar solution path idea when a solution path is indeed desired. In some cases though, no such path is necessary and instead a simple choice of the penalty parameter is preferred. We next turn to estimating a "best" value of the penalty parameter.

2.7 Choosing the Penalty Parameter

As discussed above, the penalized regression estimate of β varies with changing values of the penalty parameter λ, and as such λ indexes the solution path of β. Naturally, interest lies in a "best" choice of the penalty parameter. The apparent consensus across the penalized regression literature is to use some form of cross validation to estimate the penalty parameter [Wahba, 1990; Tibshirani, 1996; Wood, 2006; Friedman *et al.*, 2007; Breheny and Huang, 2011; Friedman *et al.*, 2010]. Though an information criterion could be used, the literature surrounding penalized regression tends to avoid these methods.

Since the ridge regression estimator $\hat{\beta}^R$ is linear in the response vector $\mathbf{y}$, generalized cross

validation (GCV) is often used to estimate λ. Generalized cross validation is an approximation to the specific case of cross validation where the number of groups K is set equal to the sample size n. By setting $K = n$, one creates n subsets of the data, leaving one observation out at a time. This idea, named the prediction sums of squares (PRESS) criterion, was originally developed by Allen [1971, 1974]. In some situations, as is the case with $\hat{\beta}^R$, it is possible to avoid the computational cost otherwise associated with repeatedly subsampling the original data [Allen, 1971; Hastie *et al.*, 2005]. Making use of ridge regression's projection matrix $\mathcal{R}_\lambda$, the GCV criterion is

$$\text{GCV}(\hat{\beta}, \lambda) = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i - x_i^t \hat{\beta}}{1 - \text{tr}(\mathcal{R}_\lambda)/n} \right)^2 .$$

Unlike ridge regression, any estimator using the ℓ_1 norm is nonlinear in the response vector $\mathbf{y}$. This includes all of the variations on the lasso mentioned above. Because of this nonlinearity, there is no obvious approximation of the PRESS criterion and thus no known form for generalized cross validation. Thus, for estimators nonlinear in $\mathbf{y}$ we use the standard cross validation estimate

$$\text{CV}(\hat{\beta}, \lambda) = \frac{1}{n} \sum_{i=1}^{n} (y_i - x_i^t \hat{\beta}^{-\kappa(i)}(\lambda)).$$

There are two main drawbacks to this. The first is extra computational effort. Under non-general cross validation one must take K random subsets of the data, calculate the method of interest on each subset, and then use the K-fold cross validation criterion to choose λ. The second issue is that, specifically in smoothing problems, as is demonstrated in Section 3.6, cross validation is well known to under smooth or to produce "wiggly" estimates [Davison, 1997; Hastie *et al.*, 2005].

2.8 Bootstrapped Standard Errors

As in the previous section, penalized regression estimators that are linear in the response vector $\mathbf{y}$ can be dealt with simply and elegantly. An estimator's variance, say, can be computed with matrix calculus. Recall, in Section 2.1 we were able to provide simple calculations of the variance of the estimator $\hat{\beta}^R$. However, this is not the case for estimators nonlinear in the response vector. Lasso-type estimators are not linear in the response $\mathbf{y}$ and thus no simple calculation of the standard errors of their coefficients is available. Estimating standard errors of the lasso is a well known problem in the literature and this issues has seen a significant amount of attention [Tibshirani, 1996; Knight and Fu, 2000; Osborne *et al.*, 2000; Fan and Li, 2001; Pötscher and Leeb, 2009; Kyung *et al.*, 2010]. At the heart of the issue is the fact that some coefficients will have probability concentrated around zero. Concentration amassed around zero produces sample distributions of the lasso estimator that are a mixture of singular and truncated normal distributions [Pötscher and Leeb, 2009]. Kyung *et al.* [2010] provide a formal proof that bootstrapped estimates of the lasso are not consistent for any coefficients that are truly 0.

It seems that deriving estimates of variance for the lasso has no easy solution. At the time of this writing no one technique stands out as the best way to produce standard errors of the lasso estimator. However, we discuss below a potential remedy to this problem. This remedy comes from a change of perspective. Bayesian penalized regression methods provide estimates of variance for parameters of interest, largely due to the prior assumed for the coefficients. The priors assumed on the coefficients provides an analogue to the penalty term in the frequentist case. The estimates of error though come at the cost of never setting any coefficient identically to zero, as the frequentist lasso does. We explore variations on penalized regression under the Bayesian perspective next.

2.9 Bayesian Variations

The Bayesian equivalent of model (2.1) is derived by putting independent Gaussian priors on the β_j,

$$f(\beta_j) = (2\pi c^2)^{-1/2} \exp\left(\beta_j^2/c^2\right).$$

Assuming that all the coefficients β have zero mean provides an analogue to the shrinkage effect caused by the norm in the penalty term. The mode of the posterior distribution on β_j is equivalent to the penalized least squares solutions found from equation (2.1). The fully Bayesian hierarchical model is then written as

$$\begin{aligned} \mathbf{y}|\mathbf{X}, \beta, \sigma^2 &\sim \mathcal{N}_p(\mathbf{X}\beta, \sigma^2 I_n) \\ \beta_j|c &\sim \prod_{j=1}^{p} \mathcal{N}(\beta_j, c) \\ \sigma^2 &\sim p(\sigma^2), \end{aligned} \tag{2.6}$$

where the the scale of the prior on β_j, namely c, plays a similar, though not directly analogous, role to the tuning parameter λ. The connection between ridge regression and the hierarchical model above is easily seen through the log of the joint distribution of $\mathbf{y}$ and β, treated as a function of β

$$\begin{aligned} \log[\mathbf{y}, \beta|\mathbf{X}, \sigma^2, c] &= \log \mathcal{N}_p(\mathbf{X}\beta, \sigma^2 I_n)\mathcal{N}_p(\beta, c) \\ &\propto ||\mathbf{y} - \mathbf{X}\beta||_2^2 + \frac{1}{c}||\beta||_2^2. \end{aligned}$$

By a similar argument, the Bayesian analogue to the lasso simply replaces the normal prior on β_j with a double exponential distribution prior [Tibshirani, 1996]. The double exponential distribution has the form

$$[x|\lambda] = \frac{\lambda}{2} \exp\left(-\lambda|x|\right).$$

This distribution can be recovered as the gamma distribution scale mixture of normals, as first shown by Andrews and Mallows [1974],

$$\begin{aligned} [x|\lambda] &= \int_0^\infty \frac{1}{\sqrt{2\pi s}} \exp\left(-x^2/(2s)\right) \frac{\lambda^2}{2} \exp\left(-\lambda^2 s/2\right) ds \\ &= \int \mathcal{N}(x|0, \psi)\Gamma(1, d\psi) \end{aligned}$$

where $\mathcal{N}(\cdot|0, \psi)$ is the Gaussian density function and $\Gamma(1, \cdot)$ is a gamma density function with shape equal to 1. In fact, any gamma distribution is acceptable. The double exponential simply produces full conditionals that are easier to work with; see Griffin and Brown [2011] for more complex priors on β. We therefore focus solely on an exponential mixing of normals.

The Bayesian lasso was first published by Park and Casella [2008]. There, they make a convincing argument in favor of a double exponential prior on β conditional on σ^2. The conditional double exponential prior ensures a unimodal posterior distribution for $[\beta, \sigma^2]$. This conditioning eases interpretation of point estimates and would otherwise slow convergence of the Monte Carlo Markov Chain used to fit the Bayesian lasso. Building off of the model in Equation (2.6), the Bayesian lasso comes from the following hierarchical model.

$$\begin{aligned} \mathbf{y}|\mathbf{X}, \beta, \sigma^2 &\sim \mathcal{N}_n(\mathbf{X}\beta, \sigma^2 I_n) \\ \beta|\sigma^2, \omega_1, \ldots, \omega_m &\sim \mathcal{N}_p(0, \sigma^2 \Sigma_\beta^{-1}) \\ \Sigma_\beta^{-1} &= \Sigma_\beta^{-1}(\omega) = \text{diag}(\omega_1^{-1}, \ldots, \omega_m^{-1}) \\ \omega_1, \ldots, \omega_m|\lambda &\sim \prod_{j=1}^{m} \frac{\lambda^2}{2} \exp(-\lambda^2\omega_j/2) d\omega_j, \quad \omega_j > 0, (j = 1, \ldots, m) \\ \lambda^2|\alpha, \rho &\sim \Gamma(\lambda^2|\alpha, \rho), \quad \lambda^2 > 0 \\ \sigma^2 &\sim \sigma^{-2}, \quad \sigma^2 > 0 \end{aligned} \tag{2.7}$$

As is common in the literature, we use the limiting improper prior from an inverse gamma distribution on σ^2, but any inverse gamma distribution would maintain conjugacy. The hyperparameters α and ρ in the prior on λ^2 must be chosen by the user. Keeping in mind that the Bayesian lasso is inherently a shrinkage method, small values of α and ρ should be chosen. Often values greater than zero and less than two are chosen. Any improper prior on λ^2 will produce improper posteriors.

The Bayesian lasso, like the Bayesian ridge regression, shrinks coefficients by assuming they are a priori centered about zero. However, because the double exponential distribution spikes around zero, the Bayesian lasso weights the information from the coefficients differently than does Bayesian ridge regression. In either case though, none of the coefficients are set identically to zero as they are in the frequentist lasso. This is due to the continuity of the prior put on the coefficients β_j. On the other hand, because of the prior, the posterior distribution of each coefficient is readily available and thus many features of the posterior are easily obtained. For instance, credible intervals are simple and informative. Credible intervals allow the practitioner to draw simple conclusions otherwise provided by the frequentist lasso. If a credible interval for a coefficient includes zero, then one could make the claim that it is not very important for predicting the response.

Figure 2.4 compares three different prior distributions for β. The Gaussian distribution (solid), which produces an analogue to ridge regression, and the double exponential (dash) have small, exponential tails. The heavy tails shrink coefficients towards zero even when the data otherwise would encourage large coefficients. As mentioned above, the heavy shrinkage of the lasso is well known to produce biased estimates [Fan and Li, 2001; Lee *et al.*, 2010; Breheny and Huang, 2011]. Frequentist solutions to reduce such bias involved developing piecewise penalty functions, e.g. SCAD and MCP. Similar piecewise priors on β in the Bayesian setting were developed [see Carvalho *et al.*, 2009, 2010], but a simpler idea is to drop the exponential tails of the prior on β. Armagan *et al.* [2013] recommend a prior that they call the generalized double Pareto (`gdp`) distribution, which takes the form

$$[x|\alpha,\rho] = \frac{1}{2\sigma\rho/\alpha}\left(1+\frac{1}{\alpha}\frac{|x|}{\sigma\rho/\alpha}\right)^{-(1+\alpha)}. \tag{2.8}$$

By putting less mass in the tails, the data can encourage larger values of the coefficients.

Tractable Gibbs samplers are readily found for the above priors and for some of the variations discussed in Section 2.4 [Gelfand and Smith, 1990]. Since these Gibbs samplers are specific cases of the samplers discussed in Chapter 3, we defer the details to later sections.

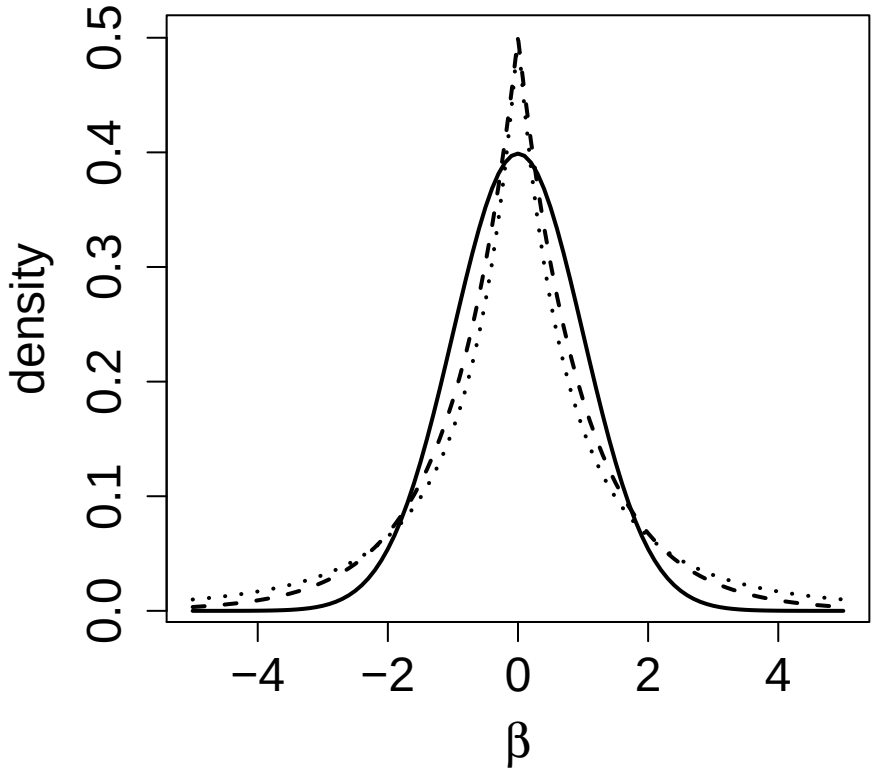

Figure 2.4: Comparison of three priors: Gaussian (solid), double exponential (dash), generalized double Pareto (dot).

Chapter 3

Penalizing Structure

Beyond penalizing the coefficients of a linear model, much literature focuses on penalizing linear transformations of the coefficient vector. This idea has been around for quite some time, generally going back to the literature on inverse problems. The ℓ_2 normed, linearly transformed penalty functions are referred to as Tikhonov regularization, named for the early work of Tikhonov [1943]. Some decades later, the statistics community developed smoothing splines using a framework very similar to the Tikhonov regularization methods. More recently, new techniques have arisen alongside the ℓ_1 norm literature. Just as before, the ℓ_1 norm acts as a data dependent choice of the important predictors. But now, the sparse solutions coupled with linear transformations of the coefficients encourage penalized geometric constraints. Such work has also generated new convex analysis problems.

Below we sketch some key points about one of the most popular smoothing methods ever developed. Smoothing splines continue to be an accurate estimator that requires minimal computational complexity. We then discuss the generalized lasso and only briefly mention the first, of what is sure to be any number of, general minimization strategies for linearly transformed penalty functions. This discussion is kept short for two reasons: 1) this first method to solve the generalized lasso's objective function is, though a recent development, already being replaced by faster methods, and 2) we provide a Bayesian solution to the generalized lasso that in particular cases seems preferable, for instance Bayesian trend filtering discussed in Section 3.6. Thus, we suffice our discussion of minimization methods for the generalized lasso to an approximate method, detailed in Section 3.3, and our Bayesian solution provided in Section 3.5. We also discuss some details that come along with the Bayesian solution to the generalized lasso and a maximum a posteriori solution to the generalized lasso in Section 3.5. We conclude this chapter with the introduction of Bayesian trend filtering in Section 3.6.

3.1 Smoothing Splines

de Boor [2001] and Wahba [1990] are credited for their large role in the development of smoothing splines, a method that penalizes a linear transformation of the coefficient vector instead of the coefficient vector itself. The popularity of smoothing splines stems from two main points: 1) the difficult mathematical theory makes, somewhat surprisingly, for a simple solution, and 2) smoothing splines are both empirically and theoretically strong, albeit not optimal, estimators. The minimization problem is incredibly simple, despite the mathematical rigor it takes to justify it. The discrete minimization problem associated with smoothing splines is

$$\hat{\theta} = \arg\min_{\theta} ||\mathbf{y} - N\theta||_2^2 + \lambda\theta^t\Omega\theta. \tag{3.1}$$

The matrix N is composed of elements $N_{ij} = \eta_j(x_i)$, where $\eta_j(x)$ are the evaluations of B-spline basis functions. The penalty term in Equation (3.1) involves the coefficients θ left and right multiplying the integral of the inner product of the second derivatives of η [Wahba, 1990; Green and Silverman, 1993; de Boor, 2001; Hastie *et al.*, 2005; Tibshirani, 2014]. That is, $\Omega_{jk} = \int \eta_j''(t)\eta_k''(t)dt$. The solution is then found with matrix calculus to be

$$\hat{\theta} = S_\lambda\mathbf{y} = (N^tN + \lambda\Omega)^{-1}N^t\mathbf{y} \tag{3.2}$$

This simple solution, bearing resemblance to the solution for ridge regression, enables easy computation of the estimated function of interest, $\hat{f} = N\hat{\theta}$. Further, the projection matrix S_λ has a number of favorable properties; [see Hastie *et al.*, 2005]. One in particular is that the degrees of freedom of a smoothing spline fit are easily found as $\mathrm{df}_\lambda = \mathrm{tr}(S_\lambda)$. Hence, smoothing spline estimators are often coupled with the generalized cross validation methods, allowing for quick estimation of the penalty parameter λ.

3.2 Generalized Lasso

We now turn our attention to a generalization of the ℓ_1 regularization methods mentioned in Section 2.4. We assume that a known matrix $D \in \mathbb{R}^{m \times p}$ first transforms the coefficients β before penalizing the resulting vector. The lasso is represented by the case where $D = I_p$ and the ℓ_1 norm is used. Though the penalty function could be any arbitrary function h_j, most work has revolved around the ℓ_1 norm penalty function. The generalized lasso takes the the following form

$$\hat{\beta} = \arg\min_{\beta} ||\mathbf{y} - \mathbf{X}\beta||_2^2 + \lambda||D\beta||_1. \tag{3.3}$$

The parameters β are no longer separable within the penalty function. The separability of the coefficients was a key assumption made in the minimization techniques of Chapter 2. Coordinate descent is no longer a valid algorithm to fit the generalized lasso. Thus, we have seen strict attention paid to the ℓ_1 norm, because Tibshirani [2011] rigorously detailed a solution path algorithm to the generalized lasso using the Lagrangian dual of (3.3). Slight modifications to his algorithm are required for the two scenarios, $\text{rank}(D) = m$ and $\text{rank}(D) < m$. The math underlying the dual path is not in and of itself extremely difficult, however dealing with the Karush-Kuhn-Tucker optimality conditions does get quite tricky. We will skip the frequentist solutions to this problem.

Below we highlight some of the applications the generalized lasso adapts to, dependent on choice of the penalty transformation D. Specifically, we list a number of penalty matrices D that fit into the generalized lasso framework of Equation (3.3). Many of the general forms listed below have been discussed in great detail elsewhere; see for example Tibshirani *et al.* [2005]; Liu *et al.* [2010]; Tibshirani [2011]; Wytock *et al.* [2014].

- *Fused lasso:* The fused lasso puts $X = I_n$ and uses the penalty matrix in Equation 3.4 to enforce adjacent observations to be small.

$$D = D_{1d} = \begin{bmatrix} -1 & 1 & 0 & \dots & 0 & 0 \\ 0 & -1 & 1 & \dots & 0 & 0 \\ & & & \dots & & \\ 0 & 0 & 0 & \dots & -1 & 1 \end{bmatrix} \tag{3.4}$$

  Under the ℓ_1 norm D_{1d} encourages adjacent observations of the response vector to have a difference of zero, and in some cases sets their difference to be identically equal to zero. Figure 3.1 displays an example of a piecewise constant function with normal error. The fused lasso, as a special case of the generalized lasso, allows for change point detection of a piecewise constant function. This proves useful in comparative genomic hybridization studies, where genes along the genome are thought to have a similar number of copies; [see Tibshirani, 2011, for an example].

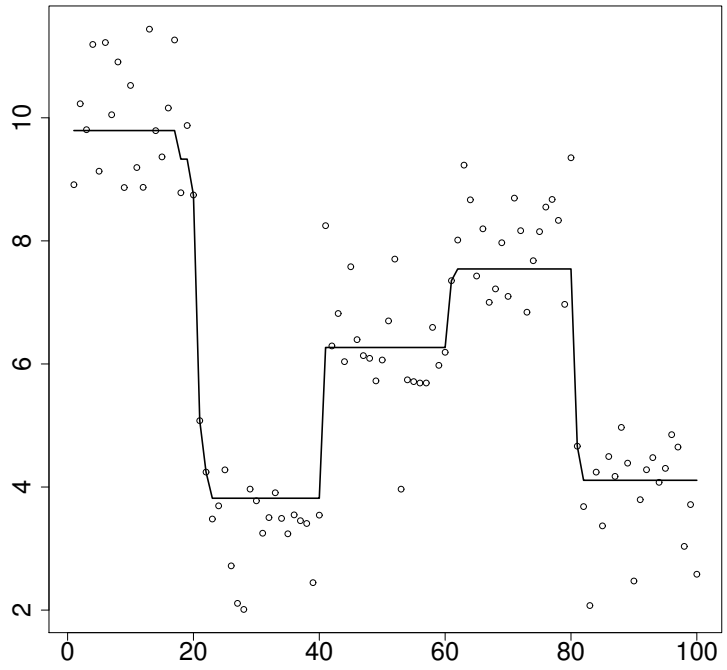

Figure 3.1: Simulated data with cross-validated fused lasso estimates.

- *2d Fused lasso:* We can apply the fused lasso to arbitrary 2-dimensional graphs. The penalty matrix D must be cleverly designed to pick out elements of the response vector $\mathbf{y}$ that share an edge. In this context, the penalty term insists that adjacent vertices are of similar magnitude. A colorful illustration of this is the contiguous 48 states of the United States.

  Consider the Center for Disease Control's 2010 data set describing lung cancer rates by state. Each state has its own rate, and it is reasonable to assume that contiguous states have similar if not identical rates. The 2-d fussed lasso makes this assumption exactly; adjacent vertices (contiguous states) have similar responses. Figure 3.2 plots 2010 age-adjusted lung cancer rates by state.

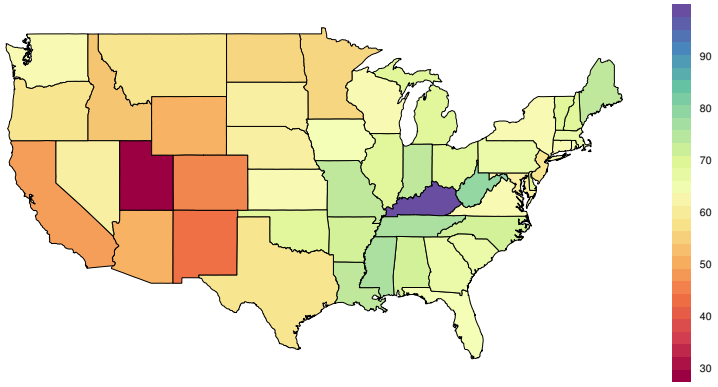

Figure 3.2: 2010 age-adjusted lung cancer rates by U.S. state.

While the generalized lasso fits this problem with ease, choosing the tuning parameter λ is still quite a difficult task. In fact, neither the frequentist nor Bayesian methods have a sufficient answer for this problem. Nonetheless, choosing by hand a value of λ demonstrates the ability of the generalized lasso to fit data that represents an arbitrary 2-dimensional graph. Figure 3.3 plots the estimates obtained from these data using the parameter value $\lambda = 1$.

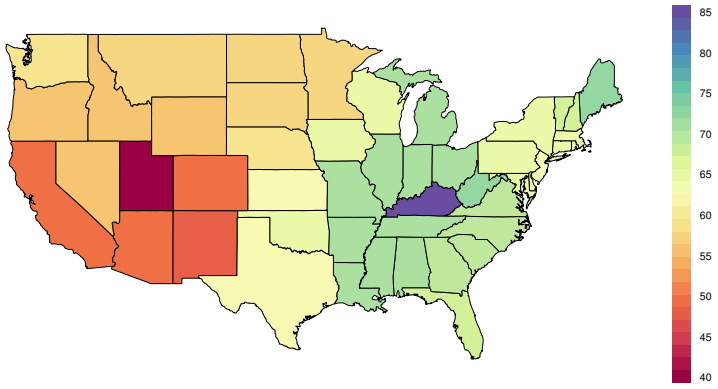

Figure 3.3: Predicted age-adjusted lung cancer rates based on a generalized lasso fit with tuning parameter $\lambda = 1$.

As hypothesized, contiguous states are pulled to the extremes. For instance, due to Utah's relatively low cancer rates we see that Colorado, New Mexico, and Arizona are each predicted

to have less than or equal rates to their true observed values. However, Nevada, which begins with a higher observed rate than its neighbors maintains a relatively high rate despite the trend of the majority of its South-West neighbors.

Different than fitting this model with an ℓ_2-norm is that some neighbors are, with the ℓ_1-norm penalty, encouraged to have identical rates across contiguous states. Whether or not this is realistic in the scenario described above is rather immaterial, as there are certainly cases in which one could imagine this feature is necessary.

- *Trend filtering:* Trend filtering is a generalization of the $1d$ fused lasso and continues to use $\mathbf{X} = I_n$. The fused lasso estimates piecewise constant functions by penalizing first order discrete differences. By penalizing estimates of discrete derivatives of higher orders one can estimate functions of higher order. Consider derivatives of order $k = 2, 3, \ldots$, and the resulting penalty matrix defined by the following recursive definition

$$D = D^{(x,k+1)} = D_{1d} \cdot D^{(x,k)}.$$

We'll discuss trend filtering in greater detail in Section 3.4 as it is one of the most interesting cases of the generalized lasso.

- *Varying coefficients:* Varying coefficient models were originally introduced by Hastie and Tibshirani [1993]. Another take on them can be handled with the generalized lasso. Consider letting the elements in the coefficient vector each depend on another continuous variable t, for instance

$$\mathbb{E}(y_i|t_i, x_i) = \beta_0(t_i) + \beta_1(t_i)x_i.$$

We could design a model matrix X to be

$$X_{ij} = \begin{cases} 1 & \text{if } t_i \text{ lies in the } j\text{th bin} \\ x_i & \text{if } t_i \text{ lies in the } (j+b)\text{th bin} \\ 0 & \text{otherwise} \end{cases}.$$

With the choice

$$D = \begin{bmatrix} D^{(t,3)} & 0 \\ 0 & D^{(t,3)} \end{bmatrix}$$

$\beta_0(t_i)$ and $\beta_1(t_i)$ are both constrained to be piecewise quadratic polynomials.

- *Outlier detection:* Relative to the model, $\mathbb{E}(\mathbf{y}|\mathbf{X}) = \mathbf{X}\beta$, suppose some of the observations y_i are not generated from the hypothesized linear model. We could fit

$$\min_{z \in \mathbb{R}^n, \beta \in \mathbb{R}^p} \frac{1}{2}||\mathbf{z} - \mathbf{X}\beta||_2^2 \text{ subject to } ||\mathbf{z} - \mathbf{y}||_1 \leq k$$

for some predetermined k. Here, k counts the number of observations labeled outliers. Consider letting $\alpha = \mathbf{y} - \mathbf{z}$ and writing

$$\min_{\alpha \in \mathbb{R}^n, \beta \in \mathbb{R}^p} \frac{1}{2}||\mathbf{y} - \alpha - \mathbf{X}\beta||_2^2 + \lambda||\alpha||_1.$$

This then fits into the generalized lasso framework with the transformed variables, $D = \tilde{D} = (I, 0)^t$ design matrix $\tilde{X} = (I, \mathbf{X})^t$, and coefficients $\tilde{\beta} = (\alpha, \beta)^t$.

3.3 Approximate Generalized Lasso

Sometimes an exact solution to Equation (3.3) is not necessary, and instead an approximate solution, computed faster is desired. A quick approximation to the objective function (3.3) can be found using the majorization-minimization techniques developed by Hunter and Li [2005]. Convergence, up to numerical precision, is nearly guaranteed since the objective function is convex in the parameters β. We begin with the definition of a majorizing function. Let $\theta^{[m]}$ be the m^{th} iteration in a search for the minimum value of some function.

Definition 3.3.1 (Majorization)**.** *A function $g(\theta|\theta^{[m]})$ is said to majorize a real-valued function $f(\theta)$ at the point $\theta^{[m]}$ if*

$$\begin{aligned} g(\theta|\theta^{[m]}) &\geq f(\theta), \quad \forall\theta, \text{ and} \\ g(\theta^{[m]}|\theta^{[m]}) &= f(\theta^{[m]}). \end{aligned}$$

Minimization of the function of interest comes after repeated minimization of the majorization and some stopping criterion is satisfied, or until some maximum number of iterations is reached. Combined with coordinate descent, minimization-maximization becomes a very powerful tool [Hunter and Li, 2005].

Hunter and Li [2005] provide us with the tools necessary to create a majorization of some popular penalty functions. Using these tools, we can find a majorizationthe of the objective function (3.3). It is reasonable to find a majorization for the ℓ_1 norm specifically, since we assume quadratic loss in the case of the generalized Bayesian lasso. For some $\epsilon > 0$, the following is a majorization of the generalized lasso model

$$g(f|f^{[0]}) = ||y-f||_2^2 + \lambda \left\{ \sum_{i=1}^{n-k-1} (||D^{(x,k+1)}f^{[0]}||_1)_i - \epsilon \log\left(1 + \frac{(||D^{(x,k+1)}f^{[0]}||_1)_i}{\epsilon}\right) + \frac{\{(||D^{(x,k+1)}f||_1)_i - (||D^{(x,k+1)}f^{[0]}||_1)_i\}^2}{2(||D^{(x,k+1)}f^{[0]}||_1)_i + \epsilon} \right\}.$$

Unfortunately, the ϵ is not easily avoided as division by zero is otherwise encouraged. Numerical precision becomes more and more of an issue as smaller values of ϵ, τ are chosen. Despite said issues with this approximation strategy, in our experience when used to fit trend filtering estimates within Section 4.1, the mean absolute difference between this approximate solution and the exact solution was generally around 10^{-3}.

3.4 Trend Filtering

Consider the nonparametric model of the function $f_0 : [0,1] \mapsto \mathbb{R}$, with zero mean, independent, sub-Gaussian errors ϵ_i

$$y_i = f_0(x_i) + \epsilon_i, \quad i = 1, \ldots, n. \tag{3.5}$$

Assume the observations y_i are generated via f_0 from the unique inputs $x_1 < \cdots < x_n$. Mammen and van de Geer [1997] propose an estimator of f_0, convergent at the minimax rate, by penalizing the total variation of the k^{th} derivative of the function f_0, defined as

$$\mathrm{TV}(f_0^{(k)}) = \sum_{i=1}^{p} |f_0^{(k)}(t_{i+1}) - f_0^{(k)}(t_i)|,$$

with the set of knots $\{t_1, \ldots, t_{p+1}\}$ taken to be the inputs. Since the penalty $\mathrm{TV}(f^{(k)})$ is not easily computed, an alternative idea is to penalize an estimate of the total variation penalty. Consider the estimator $\hat{f} = (\hat{f}(x_1), \ldots, \hat{f}(x_n))^t$ of $f_0 = (f_0(x_1), \ldots, f_0(x_n))^t$ that solves

$$\arg\min_f ||y-f||_2^2 + \lambda \widehat{\mathrm{TV}}(f^{(k)}) \tag{3.6}$$

where y is taken to be the vector of responses. R.J. Tibshirani [2014] proposed such an idea that maintains the optimal (minimax) rate of convergence. His estimate of $\text{TV}(f^{(k)})$ relies on the divided difference of order k of f. We provide some facts about the divided difference in Section 3.4, alongside some mathematical intuition behind this choice of penalty term [for a more complete survey, see de Boor, 2005].

The estimator of the function in (3.5), entitled trend filtering, by Tibshirani [2014] uses the generalized lasso framework discussed in Section 3.2. Trend filtering puts $\mathbf{X} = I_n$, $\beta = f$, and uses a scaled estimate of $\text{TV}(f^{(k)})$. The solution $\hat{f}$ is a piecewise polynomial of order k with knots taken to be the set of inputs $\{x_i\}_i^n$. Because of the close relation to locally adaptive regression splines, trend filtering adapts locally to the fluctuations of the underlying curve, and thus achieves the same optimal convergence rate [Tibshirani, 2014]. The ability to adapt locally comes from the estimate of the total variation penalty. We begin by briefly defining trend filtering's estimate of $\text{TV}(f^{(k)})$, and then discuss standard errors of a trend filtering fit.

Estimating Total Variation Let the linear space of polynomials in the field $\mathbb{R}$ be denoted by Π. Π_n is defined to be the subspace of all polynomials in Π of degree $< n$. For a set $\{x_i\}_{i=0}^p$ of unique points in $\mathbb{R}$, put $x_{\nu:\nu+k} = (x_\nu, x_{\nu+1}, \ldots, x_{\nu+k})$. We define, using the notation of de Boor [2005], the divided difference of order k as the continuous, linear functional $\triangle\colon \Pi \mapsto \mathbb{R}$.

Definition 3.4.1 (Divided Difference)**.** *The divided difference of order* 0 *is defined to be* $\triangle(x_\nu)f = f(x_\nu)$*, for* $\nu \in \{0, \ldots, p\}$*. All higher order differences follow the recurrence relation*

$$\triangle(x_{\nu:\nu+k})f = \frac{\triangle(x_{\nu+1:\nu+k})f - \triangle(x_{\nu:\nu+k-1})f}{x_{\nu+k} - x_\nu}, \quad \nu \in \{0, \ldots, p-k\}, k \in \{1, \ldots, j\}.$$

DeVore and Lorentz [1993] offer a mean value theorem (MVT) for divided differences.

Proposition 3.4.2 (MVT for Divided Differences)**.** *If* $f \in C^n[a, b]$ *and* $a \leq x_i \leq b, \forall i$*, then there exists a* $\xi \in [\min x_{\nu:\nu+k}, \max x_{\nu:\nu+k}]$ *such that*

$$\triangle(x_{\nu:\nu+k})f = \frac{f^{(k)}(\xi)}{k!}.$$

We now define the estimate of $\text{TV}(f^{(k)})$ with the linear operator $\Delta^{(k+1)} \in \mathbb{R}^{n-k-1 \times n}$. Each row of $\Delta^{(k+1)}$ is defined to be

$$\Delta_i^{(k+1)} = (x_i - x_{i+k+1})\triangle(x_{i:i+k+1}).$$

This operator is no more than a proportional rewriting of Tibshirani's operator $D^{(x,k+1)}$, which itself generalizes his transformation $D^{(k+1)}$ to unevenly spaced inputs. To see the intuition behind the use of $\Delta^{(k+1)}$, we focus on the ith row of $\Delta^{(k+1)}f$,

$$\begin{aligned}
(\Delta^{(k+1)}f)_i &= (x_i - x_{i+k+1})\triangle(x_{i:i+k+1})f \\
&= (x_i - x_{i+k+1})\frac{\triangle(x_{i+1:i+k+1})f - \triangle(x_{i:i+k})f}{x_i - x_{i+k+1}} \\
&\approx (f^{(k)}(x_{i+1}) - f^{(k)}(x_i))/k!.
\end{aligned}$$

Hence, trend filtering uses the penalty $k!||\Delta^{(k+1)}f||_1 \approx \sum_{i=1}^{n-k-1} |f^{(k)}(x_{i+1}) - f^{(k)}(x_i)|$ in a generalized lasso framework. While the intuition is nice, $\Delta^{(k+1)}$ is quite a bit less computationally friendly as it contains numbers on the order of n^k. Therefore, there is a direct advantage to using $D^{(x,k+1)} = k!\Delta^{(k+1)}/n^k$ in computations, absorbing the extra constant $n^k/k!$ into the prior on λ.

Minimax Convergence Rate Trend filtering converges at the minimax rate to the true underlying function of interest, as is shown by Tibshirani [2014]. Tibshirani, uses purely algebraic methods to show that trend filtering is sufficiently close to the locally adaptive regression spline estimator for the minimax convergence rate to carry over to trend filtering. Another strategy to prove the minimax convergence rate for trend filtering would make use of the metric entropy of the underlying function space for which trend filtering finds its solution, analogous to the strategy taken by Mammen and van de Geer [1997]. The metric entropy route requires the estimator of interest to live in a function space such that the total variation of the estimator is less than or equal to up to a constant the total variation of the true underlying function. Below we provide just enough detail to outline how such a proof could go.

Consider the strategy taken by Mammen and van de Geer [1997]. Let $\mathcal{G} = \{g : [0,1] \mapsto \mathbb{R}\}$ be a linear space endowed with the empirical inner-product and induced norm therefrom

$$\langle g_1, g_2 \rangle_n = n^{-1} \sum_{i=1}^{n} g_1(x_i) \cdot g_2(x_i)$$
$$||g||_n^2 = \langle g, g \rangle_n \quad g, g_1, g_2 \in \mathcal{G}.$$

For $\mathcal{G}_n$ a linear subspace of $\mathcal{G}$, suppose the penalty function $\mathcal{F} : \mathcal{G}_n \mapsto [0, \infty)$ has the following properties

$$\mathcal{F}(g_1 + g_2) \leq \mathcal{F}(g_1) + \mathcal{F}(g_2), \quad g_1, g_2 \in \mathcal{G}_n$$
$$\mathcal{F}(ag) \leq |a|\mathcal{F}(g), \quad g \in \mathcal{G}_n, a \in \mathbb{R}.$$

For a sequence $\{A_n\}$ that is bounded in probability by another sequence $\{B_n\}$ we write $A_n = O_P(B_n)$. Further, if $A_n^{-1} = O_P(B_n^{-1})$ is also true we write $A_n = \Theta(B_n)$. Put $\mathcal{G}_n(1) = \{g \in \mathcal{G} : \mathcal{F}(g) \leq 1\}$. For a subset $\mathcal{A}$ of $\mathcal{G}$ denote the δ-entropy of $\mathcal{A}$ by $\log N_2(\delta, ||\cdot||_n, \mathcal{A})$.

Theorem 3.4.3. *Let c_n be a positive sequence such that for a function $g_1 \in \mathcal{G}_n$ we have $||g_0 - g_1||_n = O(n^{-1/(2+w)} c_n^{w/(2+w)})$ and $\mathcal{F}(g_1) \leq c_n$. Suppose $\lambda_n = \Theta(n^{w/(2+w)} c_n^{-(2-w)/(2+w)})$. Assume $\exists C > 0$ and $0 < w < 2$ such that*

$$\log N_2(\delta, ||\cdot||_n, \mathcal{G}_n(1)) \leq C\delta^{-w}, \quad \delta > 0. \tag{3.7}$$

Some of the requirements for Theorem 3.4.3 are straight forward enough and some require much effort. For instance, Equation (3.7) is true when the function space of interest, namely $\mathcal{G}$, consists of splines [Van de Geer, 1990; Mammen, 1991; Mammen and van de Geer, 1997]. However, trend filtering uses piecewise polynomials with potentially discontinuous lower order derivatives. Thus, a more elegant solution using metric entropy methods for trend filtering would require a Herculean effort.

Standard Errors It was noted in Section 2.8 that standard errors are a particularly troubling point for lasso estimators. However, with trend filtering the previously cited theory does not find easy evidence. A quick simulation provides seemingly reasonable bootstrapped standard errors. Figure 3.4 shows the true, piecewise linear function in red and bootstrapped 95% confidence intervals completely containing it. Though this is not the case in every simulation, we find that biased estimates reduce the coverage of bootstrapped confidence intervals more than anything else. On the one hand it seems that the difference between the lasso literature and trend filtering is that in trend filtering a linear transformation of f is being penalized, instead of just the coefficients themselves. On the other hand, it seems that a linear transformation would not drastically change, let alone improve, an otherwise barely tractable distribution – recall, from Section 2.8, the asymptotic distribution of a lasso estimate for which the true coefficient is zero. It is unclear exactly why bootstrapped standard errors appear to work for trend filtering. Figure 3.5 displays the 500 bootstrapped estimates of $f(x_{25})$ and the true value is shown by the red vertical line. By sight, the worst part of the bootstrapped estimates appears to be the bias seen in Figure 3.5.

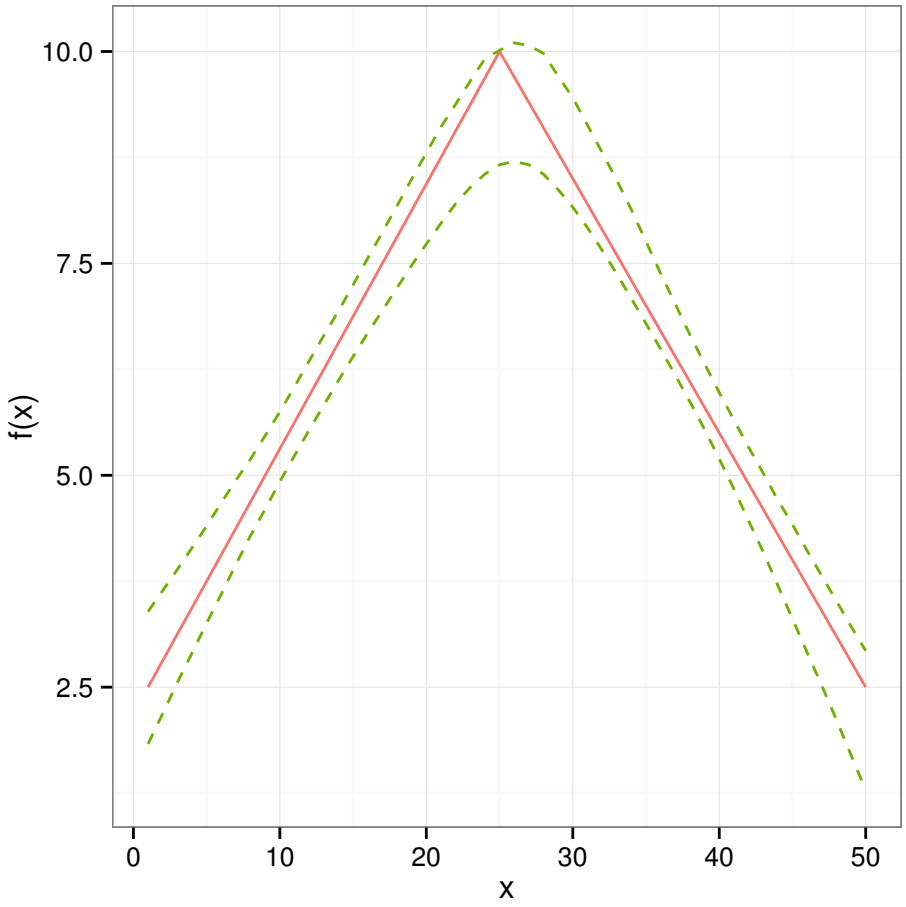

Figure 3.4: Bootstrapped 95% confidence intervals (dash green) surrounding the true function (solid red), efficiently calculated with the approximation algorithm from Section 3.3.

There are though two other problems with bootstrap estimates of the confidence intervals of a trend filtering fit. First is the amount of time it takes to estimate f for each bootstrap sample. Frequentist trend filtering methods take a good amount of time to calculate an estimate of f, as the complete solution path across all possible values of λ is computed. Using an approximation to the objective function in Equation (3.6), details of which were discussed in Section 3.3, was significantly faster. For a precalculated value of $\hat{\lambda}_{CV(5)}$, it took a standard MacBook Pro 3.1 GHz Intel Core i7 around half the time to calculate the bootstrap estimates, using 500 resamples, as it did to fit trend filtering and calculate the value of $\hat{\lambda}_{CV(5)}$.

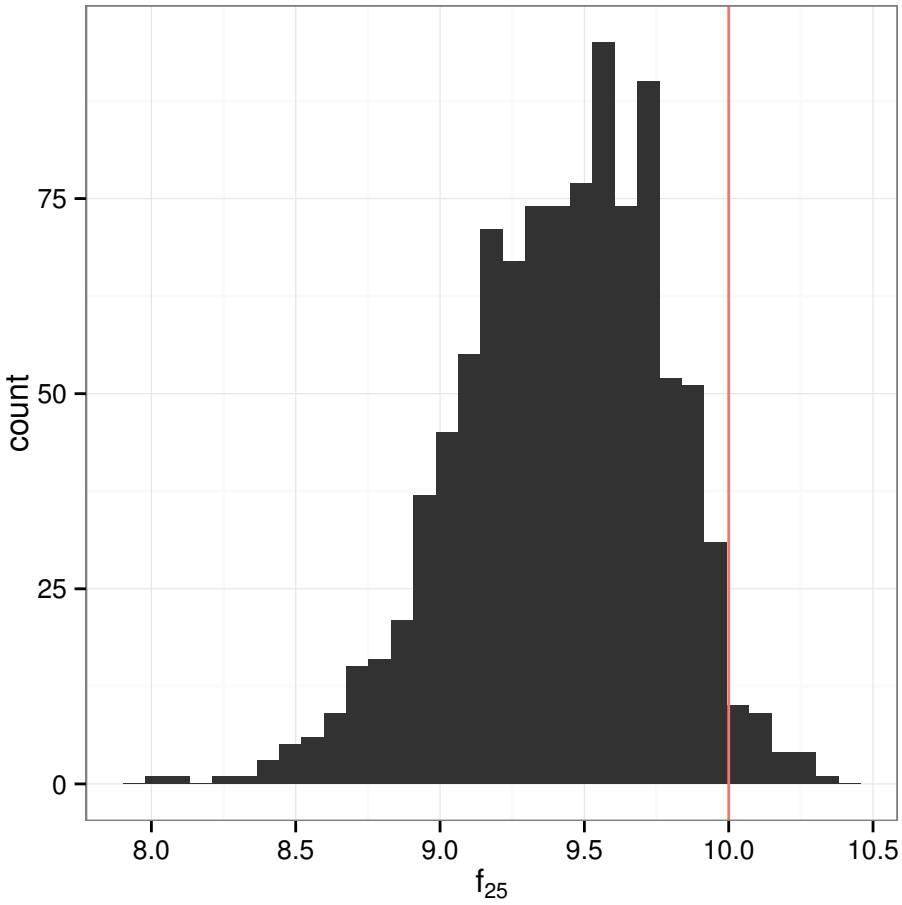

Figure 3.5: Histogram of bootstrapped values of $\hat{f}(x_{25})$ with the true value $f(x_{25})$ drawn in red.

A bigger problem stems from the estimate of $\hat{\lambda}$ found via k-fold cross validation. Cross validation in well known to encourage over-fitting [Davison, 1997; Hastie *et al.*, 2005]. The examples presented in Sections 4.1 and 4.2 highlight this point exactly. Bootstrap methods that rely on $\hat{\lambda}_{CV}$ can also appear too wiggly. From the experience gained in performing the simulations of Section 4.1 with trend filtering, when $\hat{\lambda}_{CV}$ provides a reasonable fit to the data, the bootstrap also provides reasonable estimates of $\hat{f}$. Unfortunately, it is often the case that the original trend filtering method, with $\hat{\lambda}$ chosen via cross validation, provides a poor fit to the data. Alternatively, one could choose $\hat{\lambda}$ via the one standard error rule, but this seems more of an ad hoc fix than a solution to the problems of cross validation.

The problem with choosing λ disappears under the Bayesian approach. In Bayesian trend filtering, λ is estimated by incorporating it into the hierarchical model (3.10) so that the estimates of f are marginalized over all values of λ. This in a sense encourages robust, stable estimates of f, as can be seen in the examples of Section 4.1.

3.5 Bayesian Generalized Lasso

The Bayesian analogue to the generalized lasso is found by expanding model (2.7) discussed in Section 2.9. There the conditional prior on β took the following form

$$[\beta|\sigma^2] = \prod_{j=1}^{m} \frac{\lambda}{2\sigma} \exp\left(-\lambda|\beta_j|/\sigma\right).$$

Recall that the conditional prior is preferred as this ensures a unimodal joint posterior distribution on β, σ^2. In the case of the generalized lasso, the coefficients β are now replaced with the linear combination $D\beta$. Thus, the new conditional prior to be recovered from a gamma distribution scale mixture of normals is the following conditional prior

$$[\beta|\sigma^2] \propto \exp\left(-\lambda||D\beta||_1/\sigma\right).$$

The generalized lasso form encompasses the work of Park and Casella [2008] and Kyung *et al.* [2010] and some of the variations on the lasso penalty discussed in Section 2.4. The full hierarchical model for the Bayesian generalized lasso is

$$\begin{aligned}
\mathbf{y}|\mathbf{X}, \beta, \sigma^2 &\sim \mathcal{N}_p(\mathbf{X}\beta, \sigma^2 I_n) \\
\beta|\sigma^2, \omega_1, \ldots, \omega_m &\sim \mathcal{N}_p(0, \sigma^2 \Sigma_\beta^{-1}) \\
\Sigma_\beta^{-1} = \Sigma_\beta^{-1}(\omega_1^{-1}, \ldots, \omega_m^{-1}) &= D^t \text{diag}(\omega_1^{-1}, \ldots, \omega_m^{-1}) D \\
\omega_1, \ldots, \omega_m|\lambda &\sim \prod_{j=1}^{m} \frac{\lambda^2}{2} \exp(-\lambda^2 \omega_j/2) d\omega_j, \quad \omega_j > 0, (j = 1, \ldots, m) \\
\lambda|\alpha, \rho \sim \text{Ga}(\lambda|\alpha, \rho), \quad &\text{or} \quad \lambda^2|\alpha, \rho \sim \text{Ga}(\lambda^2|\alpha, \rho), \quad \lambda > 0 \\
\sigma^2 &\sim \sigma^{-2}, \quad \sigma^2 > 0
\end{aligned} \tag{3.8}$$

where $\omega_1, \ldots, \omega_m$ are mutually independent. A tractable Gibbs sampler follows from the hierarchical model in equation (3.8). Equation (3.9) contains the full conditionals used to fit the Bayesian generalized lasso. The details behind the full conditionals are provided in Appendix A.1. We will use the inverse Gaussian distribution, and denote it IG. The density function for the inverse Gaussian distribution is

$$f(x|\mu, \xi) = \left(\frac{\xi}{2\pi x^3}\right)^{1/2} \exp\left\{\frac{-\xi(x-\mu)^2}{2\mu^2 x}\right\} 1(x > 0).$$

Further, let Γ and Γ^{-1} denote the gamma and inverse gamma distributions, respectively.

$$\begin{aligned}
\beta|\cdot &\sim \mathcal{N}((\mathbf{X}^t\mathbf{X} + \Sigma_\beta^{-1})^{-1}\mathbf{X}^t\mathbf{y}, \sigma^2(\mathbf{X}^t\mathbf{X} + \Sigma_\beta^{-1})) \\
1/\omega_j|\cdot &\sim IG\left(\sqrt{\frac{\lambda^2\sigma^2}{|(D\beta)_j|^2}}, \lambda^2\right) \\
\sigma^2|\cdot &\sim \Gamma^{-1}(\frac{n-1+p}{2}, \frac{1}{2}(y - X\beta)^t(y - X\beta) + \frac{1}{2}\beta^t\Sigma_\beta^t\beta) \\
\lambda^2|\cdot &\sim \Gamma(m + \alpha, \rho + \sum_{j=1}^{m} \omega_j/2)
\end{aligned} \tag{3.9}$$

Sampling from the posterior for β is the most computationally intensive part of the Gibbs sampler for the Bayesian generalized lasso. To sample from the multivariate Gaussian distribution, a full matrix inversion must take place. Unfortunately, no quick solution to a linear system is feasible within this Gibbs sampler. The least computationally expensive strategy to avoid this direct matrix inversion is to take advantage of the positive definiteness of the matrix $(\mathbf{X}^t\mathbf{X} + \Sigma_\beta^{-1})$. By using a Cholesky decomposition defined by L, a lower triangular matrix such that $LL^t = (\mathbf{X}^t\mathbf{X} + \Sigma_\beta^{-1})$, only one matrix inversion is necessary [Golub *et al.*, 1979]. This is the exact strategy used in the Gaussian process regression literature; [see Rasmussen, 2006]. Algorithm 3.1 provides the details behind the Cholesky decomposition employed within the full conditional for β found in equation 3.9.

Algorithm 3.1: β FULL CONDITIONAL

Input: $(\mathbf{X}, \Sigma_\beta^{-1}, \mathbf{y}, \sigma)$
$L \leftarrow \text{Cholesky}(\mathbf{X}^t\mathbf{X} + \Sigma_\beta^{-1})$; $\quad L^{-1} \leftarrow L^t \backslash I$
$\mu \leftarrow L^{-t}(L^{-1}\mathbf{y})$; $\quad \Lambda^{-1} \leftarrow L^{-t}L^{-1}/\sigma$
Output: $\beta \leftarrow N_m(\mu, \Lambda^{-1})$

The last line of Algorithm 3.1 writes the multivariate normal with the precision matrix instead of the covariance matrix. Sampling from the multivariate normal distribution using Algorithm 3.1 requires inverting only L, a lower triangular matrix, instead of inverting the entire covariance matrix. This strategy offers reduced computationally complexity, albeit only by reducing the coefficient of the $\mathcal{O}(p^3)$ cost of a matrix inversion.

As in Section 2.9 the generalized double Pareto conditional prior is a simple modification to hierarchical model (3.8). The generalized double Pareto conditional prior is found by putting a gamma prior on λ instead of λ^2. For other hierarchical models, this slight variation is shown to produce significant advantages over the double exponential conditional prior [Armagan *et al.*, 2013]. The full conditional for the generalized double Pareto conditional prior is $[\lambda|\cdot] \sim \Gamma(n - k - 1 + \alpha, ||D^{(x,k+1)}f||_1/\sigma + \rho)$. Similar to the double exponential conditional prior, small values of the parameters α and ρ should be used to encourage shrinkage [Lee *et al.*, 2012; Armagan *et al.*, 2013]. Some benefits of this flatter conditional prior in the case of Bayesian trend filtering are explored in Section 4.1.

Alternatively, it is possible to put hyperpriors on α and ρ, which don't involve any hyperparameters. Such hyperpriors remove all user choice of parameters when fitting the Bayesian generalized lasso. Following the work of Armagan *et al.* [2013], consider the generalized double Pareto conditional distribution and a prior with median of one on α. This is a reasonable choice since many Bayesian lasso author's seem to arbitrarily choose α to be small and relatively close to zero [Park and Casella, 2008; Kyung *et al.*, 2010; Armagan *et al.*, 2013]. Since the same framework can be used for ρ, we provide details only for α. Assume the prior on α to be $[\alpha] = 1/(1+\alpha)^2$. The full conditional on α can be sampled via the griddy Gibbs sampler [Ritter and Tanner, 1992; Armagan *et al.*, 2013]. First, transform α into the domain $[0, 1]$ via $a = 1/(1+\alpha)$. Draw M values $a^{(m)}$ from $\mathcal{U}(0, 1)$ and randomly select one of the $\{a^{(m)}\}_{m=1}^M$ with probabilities $\{w^{(m)}\}_{m=1}^M$ where $w^{(m)} = [a^{(m)}|\cdot]$, and then transform back to the desired scale for α;

$$[a|\cdot] \propto \left(\frac{1-a}{a}\right)\left(1 + \frac{||D\beta||_1}{\sigma\rho}\right)^{-(m+1/a-1)}.$$

Model (3.8) can be viewed as an extension of the work in Kyung *et al.* [2010]. Therefore, we appeal to their Propositions 4.1 and 4.2 which show that the underlying Gibbs sampler is geometrically ergodic. The two Gibbs samplers, for the double exponential and the generalized double Pareto conditional priors, converge both in theory and in practice very quickly.

Proposition 3.5.1. *The Gibbs sampler for the hierarchical model (3.8) is geometrically ergodic.*

Since the proof is exactly the same, we refer the reader to the proof in Kyung *et al.* [2010]. We will make use of this proposition in Section 3.6 in an effort to speed up the fit of Bayesian trend filtering.

Maximum a Posteriori Solution As maximum a posteriori estimators are getting much attention of late, we briefly note how the generalized lasso, and hence trend filtering, can be adapted within the generalized double Pareto conditional prior framework [Lee *et al.*, 2010; Griffin and Brown, 2011; Lee *et al.*, 2012; Armagan *et al.*, 2013]. We essentially build an expectation-maximization algorithm out of the Bayesian hierarchical model in Equation (3.9). The $E-$step consists of taking the expected value of the log-posterior with respect to the distribution of the latent variables, $1/\omega_j$ and λ. Hence,

$$\mathbb{E}_{1/\omega_j,\lambda} l \propto -\left(\frac{n+p}{2} + 1\right)\log\sigma^2 - \frac{(\mathbf{y} - \mathbf{X}\beta)^t(\mathbf{y} - \mathbf{X}\beta) - (D\beta)^t B^{(l)}(D\beta)}{2\sigma^2},$$

where superscript (l) denotes the l^{th} iteration, $B^{(l)} = \text{diag}(b_1^{(l)}, \ldots, b_m^{(l)})$, and

$$b_j^{(l)} = \frac{(m+\alpha)\sigma^{2(l)}}{(||D\beta^{(l)}||_1)_j^{(l)}\{(||D\beta^{(l)}||_1)_j^{(l)} + \sigma^{(l)}\rho\}}.$$

The $M-$step is easily found to be

$$\begin{aligned}
\beta^{(l+1)} &\leftarrow (\mathbf{X}^t\mathbf{X} + (D\beta^{(l)})^t B^{(l)}(D\beta^{(l)}))^{-1}\mathbf{X}^t\mathbf{y} \\
\sigma^{2(l+1)} &\leftarrow \frac{(\mathbf{y} - \mathbf{X}\beta^{(l+1)})^t(\mathbf{y} - \mathbf{X}\beta^{(l+1)}) + (D\beta^{(l+1)})^t B^{(l)}(D\beta^{(l+1)})}{2(n+p+2)}.
\end{aligned}$$

Although the formulation of maximum a posteriori generalized lasso is not difficult, we do not consider it any further.

3.6 Bayesian Trend Filtering

Bayesian trend filtering is the adaption of trend filtering, itself a special case of the generalized lasso, into a fully Bayesian hierarchical model. Two main benefits come from using the Bayesian specification of trend filtering. A fully tractable Gibbs sampler allows for easy estimation of the parameters of interest, including the tuning parameter λ. As shown in Section 3.4, choice of the penalty parameter in the frequentist case is not easy to come by, especially since no GCV criterion is available. With an accurate estimate of λ in hand, Bayesian trend filtering achieves strong predictive and frequentist properties. Coverage probabilities for Bayesian trend filtering, as compared to a number of other smoothers, are closer to nominal levels. Further, as with any Bayesian model, summary statistics of the posterior distributions are readily available.

Similar to the Bayesian lasso [Park and Casella, 2008; Kyung *et al.*, 2010] and the fully Bayesian hierarchical model in Equation (3.8), trend filtering can be adapted into a fully Bayesian hierarchical model using a scale mixture of normals to achieve the desired conditional prior on β [Andrews and Mallows, 1974; Park and Casella, 2008; Kyung *et al.*, 2010; Griffin and Brown, 2011; Armagan *et al.*, 2013]. We explore the double exponential conditional prior, `dexp`,

$$[f|\sigma] \propto \exp\left(-\frac{\lambda}{\sigma}||D^{(x,k+1)}f||_1\right)$$

and the generalized double Pareto, `gdp`,

$$[f|\sigma] = \frac{1}{2\sigma\rho/\alpha}\left(1 + \frac{1}{\alpha}\frac{||D^{(x,k+1)}f||_1}{\sigma\rho/\alpha}\right)^{-(n-k-1+\alpha)}.$$

Both of these conditional priors stem from the same hierarchical model (3.10), while the difference comes from the prior put on the penalty parameter λ. The Bayesian trend filtering hierarchical model takes the form

$$\begin{aligned}
y|f,\sigma^2 &\sim \mathcal{N}_n(f, \sigma^2 I_n) \\
f|\sigma^2, \omega_1, \ldots, \omega_{n-k-1} &\sim \mathcal{N}_n(0, \sigma^2\Sigma_f^{-1}), \\
\Sigma_f^{-1} = \Sigma_f^{-1}(\omega_1^{-1}, \ldots, \omega_{n-k-1}^{-1}) &= (D^{(x,k+1)})^t \text{diag}(\omega_1^{-1}, \ldots, \omega_{n-k-1}^{-1}) D^{(x,k+1)} \\
\omega_1, \ldots, \omega_{n-k-1}|\lambda &\sim \prod_{j=1}^{n-k-1} \frac{\lambda^2}{2}\exp(-\lambda^2\omega_j/2)d\omega_j, \quad \omega_j > 0, \forall j \\
\lambda|\alpha,\rho &\sim \psi(\lambda|\alpha,\rho)d\lambda, \quad \lambda > 0 \\
\sigma^2 &\sim \pi(\sigma^2)d\sigma^2, \quad \sigma^2 > 0
\end{aligned} \tag{3.10}$$

where $\omega_1, \ldots, \omega_{n-k-1}$ are mutually independent and $\pi : x \mapsto x^{-1}$. The two conditional priors `dexp` and `gdp` are found by putting priors on λ^2 or λ, respectively. Following Park and Casella [2008];

Kyung *et al.* [2010], a $\Gamma(\alpha, \rho)$ prior on λ^2 leads to the `dexp` conditional prior on $[f|\sigma]$. The `gdp` conditional prior is found by putting a $\Gamma(\alpha, \rho)$ prior on λ. A simple and tractable Gibbs sampler for either of the conditional priors relies on the following set of full conditionals

$$\begin{aligned} f|\cdot &\sim \mathcal{N}_n\left((I_n + \Sigma_f^{-1})^{-1}y, \sigma^2(I_n + \Sigma_f^{-1})^{-1}\right), \\ 1/\omega_j|\cdot &\sim \text{invGaussian}\left(\sqrt{\frac{\lambda^2\sigma^2}{|(D^{(x,k+1)}f)_j|^2}}, \lambda^2\right), \quad \forall j, \\ \sigma^2|\cdot &\sim \Gamma^{-1}\left(n, \frac{1}{2}(y-f)^t(y-f) + \frac{1}{2}f^t\Sigma_f^{-1}f\right). \end{aligned} \tag{3.11}$$

The full conditionals for the penalty parameter λ^2, for `dexp`, is $\Gamma(n-k-1+\alpha, \sum_{j=1}^{n-k-1}\omega_j/2+\rho)$, and relative to `gdP`, $[\lambda|\cdot] \sim \Gamma(n-k-1+\alpha, ||D^{(x,k+1)}f||_1/\sigma+\rho)$.

The term ρ can negatively effect the overall fit of both `gdp` and `dexp`. To ensure a true thresholding rule, and thus encourage shrinkage, a small value of ρ should be chosen [Fan and Li, 2001; Lee *et al.*, 2012; Armagan *et al.*, 2013]. One can further consider putting a hyperprior on α or ρ; for instance consider the hyperprior setup discussed in Section 3.5 for Bayesian trend filtering. The weight function used in the griddy Gibbs sampler, relative to Bayesian trend filtering is as follows

$$[a|\cdot] \propto \left(\frac{1-a}{a}\right)\left(1 + \frac{||D^{(x,k+1)}f||_1}{\sigma\rho}\right)^{-(n-k-1+1/a-1)}.$$

Despite the appeal of removing all user choice of hyperparameters in Bayesian trend filtering, this hyperprior had very little effect. This is possibly due to the fact that the Bayesian trend filtering hierarchical model is already over parameterized, and thus little information was available to inform the hyperpriors on α or ρ.

With or without hyperpriors on α, ρ, the Gibbs samplers for Bayesian trend filtering converge quickly. The theoretical justification for this was developed by Kyung *et al.* [2010] and mentioned in Section 3.5. Thus, by Proposition 3.5.1, the Gibbs samplers in Equation (3.11) are geometrical ergodic. Empirical evidence of the convergence rate is given in Section 4.2.

An added benefit of the hierarchical model for Bayesian trend filtering comes from the penalty matrix $D^{(x,k+1)}$. This matrix is a sparse, $k+2$ banded matrix with all other elements, outside of the bands, zero. Special algorithms designed for sparse matrices can be used everywhere within Algorithm 3.1. Section A.2 contains C++ code, which uses the matrix library Eigen [Guennebaud *et al.*, 2010; Bates and Eddelbuettel, 2013], used to fit Bayesian trend filtering.

Not Identically Zero A few points contrasting trend filtering with Bayesian trend filtering should be noted. Trend filtering, by restricting the parameter space of the objective function (3.6) sets some terms in the penalty to exactly zero. Such a data dependent selection of important predictors is philosophically appealing. Within trend filtering, this data dependent selection corresponds to setting estimates of the terms in the total variation of $f^{(k)}$ to zero. Bayesian trend filtering, however, never sets any terms identically to zero. Figure 3.6 compares Bayesian trend filtering to the frequentist trend filtering, where it can be seen that the Bayesian fit doesn't quite predict a piecewise linear function when in fact the true underlying function is piecewise linear, with three knots at $20, 45$, and 80. Though, it should be noted that the trend filtering fit lies completely within the highest posterior density (and credible) intervals of the Bayesian trend filtering fit. What Bayesian trend filtering sacrifices in knot detection, it makes up for when fitting smooth curves. The information gained by incorporating λ into the Gibbs sampler, and the propagation of that information back to the estimates of f, provides stable estimation as is shown in Section 4.1. Thus, the primary advantage of Bayesian trend filtering is as a smoother and not as a knot-detection method.

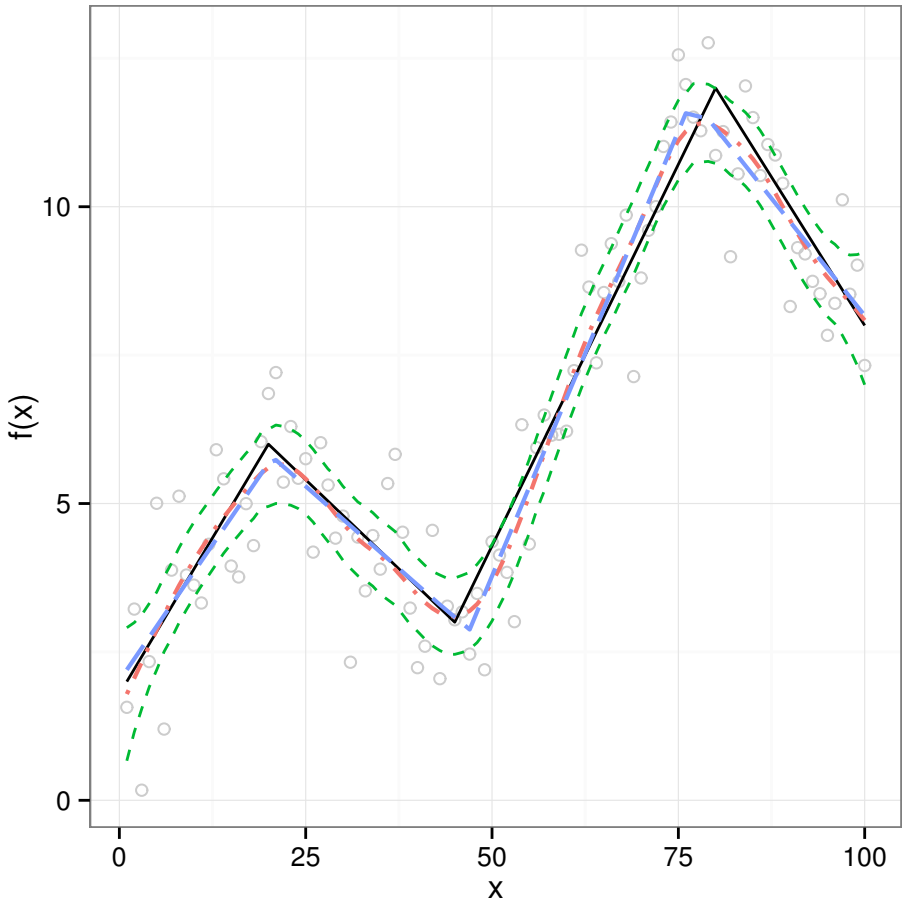

Figure 3.6: Bayesian trend filtering (dot-dash red) and trend filtering (dash blue) fits, with `BTF` highest posterior density intervals (dash green), plot against the true function (solid black).

Numerical Stability The very idea of the (generalized) lasso, to shrink some of the elements of the penalty term towards zero possibly setting some to exactly zero, can cause computational issues in the Bayesian setting. Consider the full conditional $[1/\omega_j|\cdot]$, for either `dexp` or `gdP`. The mean in the inverse Gaussian distribution inverts exactly that which we are seeking to shrink towards zero. Any sample from the posterior distribution such that an element of the penalty term is very close to zero, threatens numerical stability when drawing samples from the already rather numerically sensitive inverse Gaussian distribution; see Wheeler [2013] and the references there within. From the experience gained with the simulations of Bayesian trend filtering in Chapter 4, when an element of $|D^{(x,k+1)}f|$ is too small, simulating a draw from the full conditional for $[1/\omega_j|\cdot]$ can return a value less than or equal to zero; obviously a problem for a distribution with non-negative support. To ameliorate such events, we introduce the admittedly inelegant solution of resampling the entire vector f if any element of $|D^{(x,k+1)}f|$ is less than 10^{-10}. We find that resampling happens less than roughly five percent of the time. Further, when resampling does occur, it is extremely rare that more than one resample is ever needed. Despite such numerical issues, restricting the support of certain posterior distributions does not appear to hinder Bayesian trend filtering from performing quite well.

Speed Up BTF Bayesian trend filtering, and for that matter any of the hierarchical models that fit penalized regression methods discussed above, requires heavy computations. The vast majority of the computational time is spent inverting the matrix specified in the full conditional for the

function evaluations f (or β in any of the other models discussed). While Bayesian trend filtering has the advantage of using the sparse, $k+2$ banded penalty matrix $D^{(x,k+1)}$, the cubic cost of the matrix inversion still exists. Worse, this cost exists in every sample of the full conditional. Much research exists and yet the Cholesky decomposition strategy presented here is often the best option [Rasmussen, 2006; Vehtari and Vanhatalo, 2007].

Bayesian trend filtering has the same hierarchical framework as Gaussian processes priors; both methods use a Gaussian distribution to model the observations and then a Gaussian process prior on the underlying function. Framed as work on Gaussian processes, there are many new methods that try to reduce the computational burden associated with these related hierarchical models. Some methods reformulate the Monte Carlo algorithm into the total energy of the system as is done in the hybrid Monte Carlo approach (also known as Hamiltonian Monte Carlo) [Duane *et al.*, 1987; Gelman *et al.*, 2014], and other methods, namely variational Bayes, attempts to approximate the posterior distribution of interest [Neal and Hinton, 1998; Jordan *et al.*, 1999; Winn, 2004; Winn and Bishop, 2005]. These methods generally won't work for Bayesian trend filtering. Hybrid Monte Carlo methods are designed specifically to perform well when there are a small number of highly correlated parameters. Variational Bayes methods work best when there are many parameters that exhibit little correlation. Bayesian trend filtering, however, has many, highly correlated parameters.

Here, we attempt a simpler idea than those mentioned in the previous paragraph. As borne out by the robust simulations of Section 4.1, the function evaluations of Bayesian trend filtering seems to converge quite quickly. This is not to say that all of the parameters of interest mix well, in fact the penalty parameter λ specifically does not mix well. We suggest to draw from the full conditional for f every mth iteration, while sampling from the other parameters of interest every iteration. Since all other parameters of Bayesian trend filtering's hierarchical model can be sampled quite quickly, it is of little concern to sample all other parameters every iteration. Sampling all other parameters encourages large effective sample sizes amongst the parameters that mix more slowly. This technique is empirically tested in Section 4.2.

Chapter 4

Empirical Study of Bayesian Trend Filtering

In this chapter, we study Bayesian trend filtering on a number of examples. Section 4.1 compares trend filtering methods with other popular smoothers and another Bayesian regression method. We explore real data sets in Section 4.2. Section 4.3 concludes this chapter with a summary of the empirical performance of Bayesian trend filtering.

4.1 Simulation Study

We compare Bayesian trend filtering (`BTF`), with the priors `dexp` and `gdp`, against four different methods: trend filtering (`TF`) from Tibshirani [2014], Bayesian additive regression trees (`BTree`) from Chipman *et al.* [2010]; Kapelner and Bleich [2013], and two versions of cubic smoothing splines, `SM` and `CSM` [Wahba, 1990; Green and Silverman, 1993; de Boor, 2001; Wood, 2006; Core Team, 2014]. We include smoothing splines as they are arguably the most used method of smoothing, and also to highlight a different point than was made of the same comparison by Tibshirani [2014]. There, a strong argument was made for the efficiency of `TF`, implicitly defined as mean squared error (mse) per degree of freedom. In that world, `TF` clearly stands above as its asymptotic results are shown to hold in finite samples. Here, a more applied world is hypothesized; i.e. where estimation of the penalty parameter λ further affects each of the above methods' performance. `BTree` is included as it is a popular Bayesian regression method.

The functions, all from various `R` packages, used in the simulations are as follows. The two cubic smoothers were fit using `mgcv::gam` for `CSM` [Wood, 2006] and `stats::smooth.spline` for `SM` [Core Team, 2014]. These two methods were fit with all inputs used as knots (to make more fair the comparison between the cubic smoothing spline and the trend filtering methods) and their associated generalized cross validation function. `BTree` was fit with `bartMachine::bartMachine` using the default values of 50 trees and 9000 (after burn-in) iterations [Kapelner and Bleich, 2013]. `TF` was fit with `genlasso::trendfilter` also using a cubic piecewise polynomial, with both 5- and 10-fold cross validation [Tibshirani, 2014]. The `BTF` methods were fit using `btf::btf` with a piecewise polynomial of degree 3 and 9000 (after burn-in) posterior draws [Roualdes, 2014]. To explore the sensitivity of `BTF` with respect to the choice of the hyperparameters α and ρ, we tested all combinations of the following hyperparameters: $\alpha \in \{0.1, 0.5, 1.0, 1.5, 2.0\}$ and $\rho \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$.

For simulated data, we consider two univariate functions $f : [0, 1] \mapsto \mathbb{R}$ with regularly spaced inputs. The first, a piecewise cubic function is borrowed from Tibshirani [2014]. The second is a difficult to fit, spatially inhomogeneous function, colloquially known as dampened harmonic motion. The same framework is used throughout the simulations: $R = 1000$ replications of the above functions with three different levels of normal noise, evaluated upon the following criteria. For each replication r we calculate mean and standard deviations, across the 1000 replications, of the mean squared errors (mse)

$$\text{mse}_r = \frac{1}{n} \sum_{i=1}^{n} (f_i - \hat{f}_i)^2$$

and 95% confidence intervals for both the underlying function evaluated at all inputs and the variance, using the bootstrap with 1000 resamples for `SM` and `TF`. Since, `CSM` has its own built-in method to calculate standard errors, it was used instead of the bootstrap. Posterior samples are used to create credible intervals for all the Bayesian methods. Thus, the 1000 replications are used to estimate an average and standard deviations of the mses, and overall mean and standard errors of the two different coverage probabilities.

For real world data, we consider two datasets common to the smoothing literature. The first is a dataset of global mean surface temperature deviations for the years 1881 to 2005 from Hodges [2013]. The second is the SILSO dataset which consists of monthly average sunspot counts for the years 1980 to 2014, inclusive [Center, 2014].

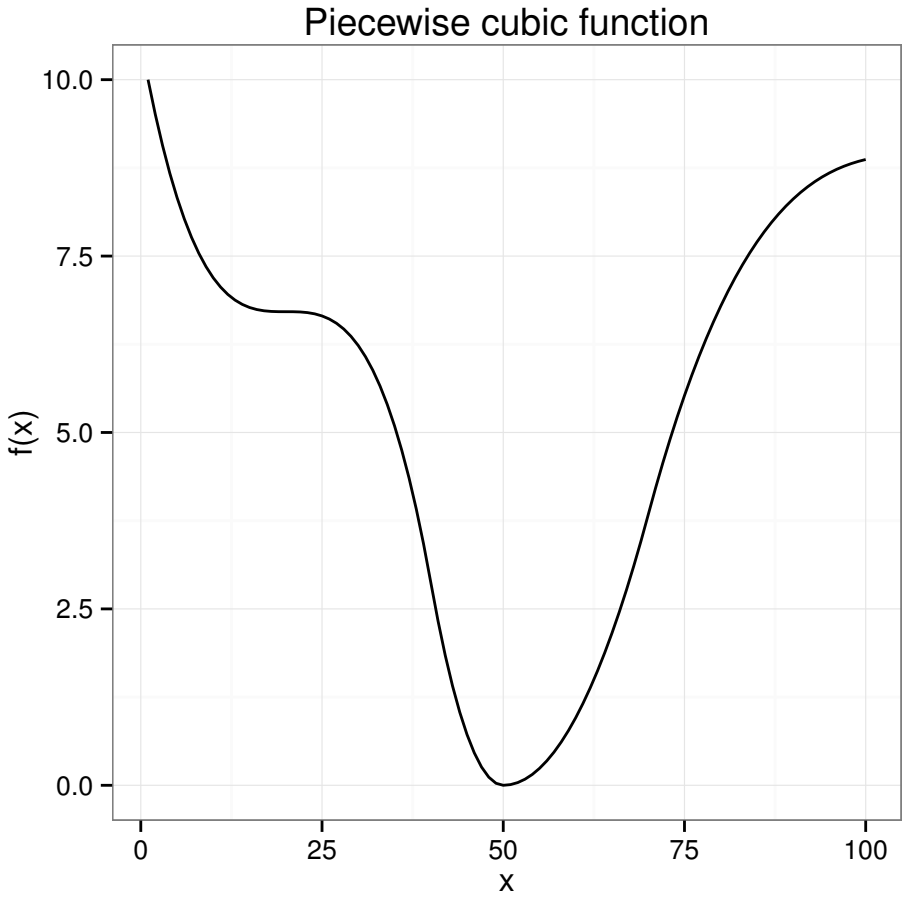

Figure 4.1: Piecewise cubic function.

Piecewise Cubic Boxplots of the mses of the 1000 replications of 100 observations from the true piecewise cubic function shown in Figure 4.1 look quite similar across the three levels of error, $\sigma \in \{0.75, 1.0, 1.25\}$. The `BTF` fits change most drastically as the hyperparameter ρ varies, as is seen by comparing Figure 4.2 with Figure 4.4, both of which show only the case $\sigma = 1$ and $\alpha = 1$. Figure 4.2 best represents all scenarios (inclusive of σ, α varied across their chosen values) where $\rho < 1$, while Figure 4.4 presents the case $\rho = 1$. This is exactly what Armagan *et al.* [2013] mean when they say suggest small values of the hyperparameters to encourage shrinkage. The mses for `BTF` don't vary greatly with respect to changes of the hyperparameter α within the range of values chosen. The fits for `TF` were quite similar for both 5- and 10-fold cross validation.

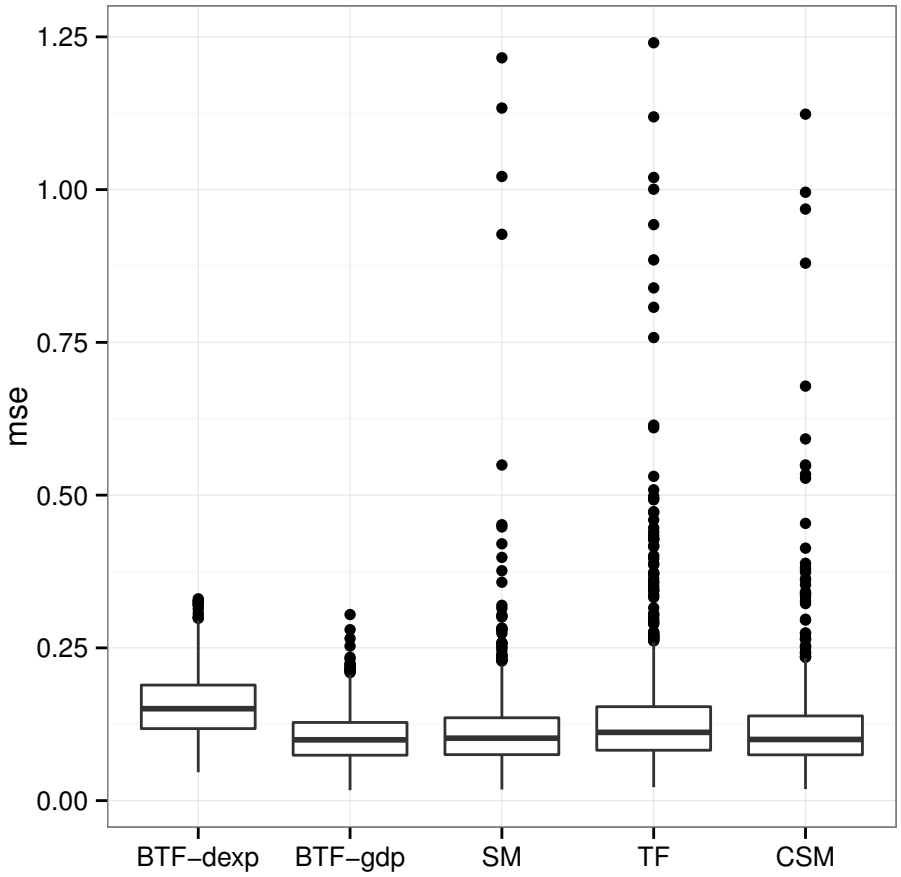

Figure 4.2: Box plots of mses by method for the piecewise cubic function with $\sigma = 1$. `BTF` with the hyperparameters $\alpha = 1$ and $\rho = 10^{-2}$, and `TF` used 10-fold cross validation are displayed.

Of interest are the worst (largest mse) fits of Bayesian trend filtering and the other methods. Plots of the worst fits from Figure 4.2 are contained in Figure 4.3. The cubic smoothing spline `CSM` is displayed, alongside Bayesian trend filtering, since `CSM` is the next best method as judged by largest mses contained in Figure 4.2. It is clear from Figure 4.3 that `CSM` performs poorly across the entire domain of the piecewise cubic function and not just in a specific subset of the domain (e.g. the tails). Because the two fits displayed in Figure 4.3 are for different simulations, it doesn't make sense to display the data points the generate these two fits. We investigated the data points for this poor fit of `CSM`, and there are no clear indications as to why this fit is relatively so poor.

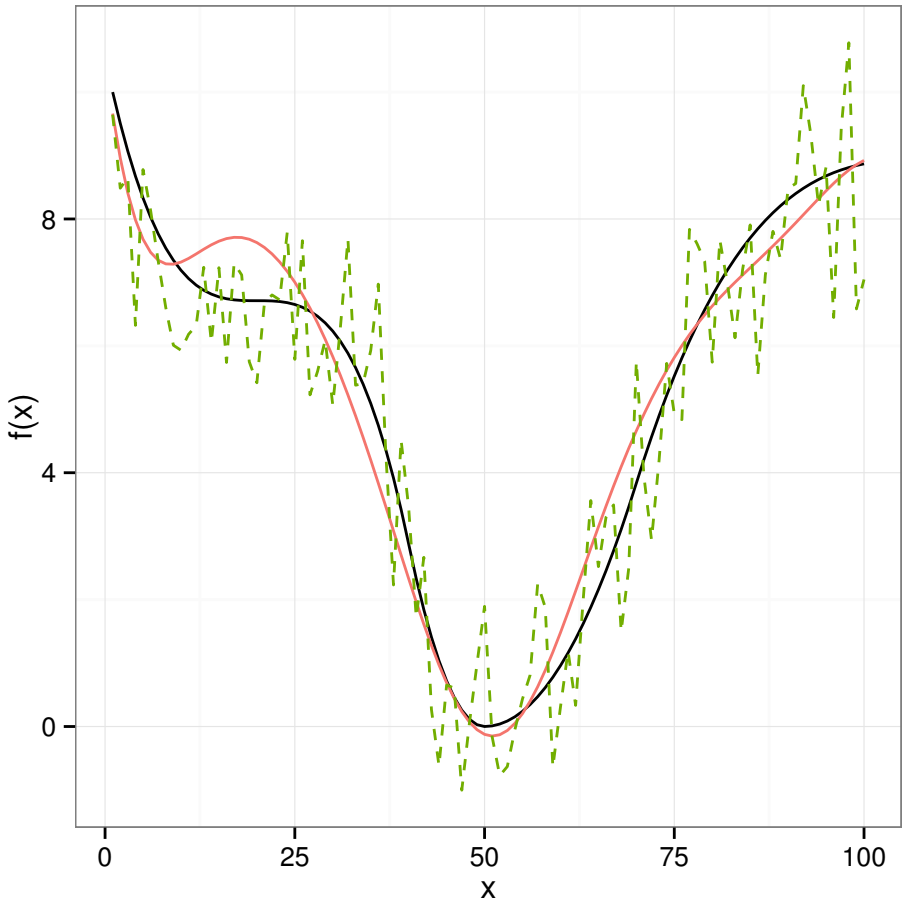

Figure 4.3: The two worst fits of `BTF` (solid red) and `CSM` (dash green) as judged by largest mse from Figure 4.2.

It is clear from Figures 4.2 and 4.4 that `BTF` with the `gdp` prior, compared to the `dexp` prior, is the preferred method for the piecewise cubic function. Simple, reasonable choices of the hyperparameter ρ, namely any $\rho < 1$, allow `BTF` to accurately fit this piecewise cubic function. For the cases where $\rho < 1$, `BTF-gdp` on average has smaller average mse and smaller standard deviation of the mses than the cubic smoothing splines and frequentist trend filtering. Further, `BTF-gdp` excels in minimizing the number of extreme fits to the data. For instance, in Figure 4.2 there are 13, 45, and 25 fits from `SM`, `TF`, and `CSM`, respectively, where their mse is greater than the largest mse for `BTF-gdp`. And conversely, there are no fits that provide an mse smaller than the smallest mse from `BTF-gdp`.

As shown in Table 4.1 Bayesian trend filtering with the `gdp` prior, where $\alpha = 1$ and $\rho = 10^{-2}$, improves both the overall mean and standard deviation of the 1000 mses as compared to the other methods. For the piecewise cubic function, Bayesian trend filtering with the `gdp` prior attains the minimum mean and standard deviation of the mses for all noise levels, though, this status is shared when $\sigma = 1$, where `SM` shares a mean mse of 11. In this case, however, `SM` has greater standard deviation.

	$\sigma = 0.75$		$\sigma = 1$		$\sigma = 1.25$	
method	mean	sd	mean	sd	mean	sd
BTF-dexp	8.9	3.0	16	5.2	25	7.9
BTF-gdp	**6.6**	**2.4**	**11**	**4.2**	**15**	**6.4**
SM	6.9	4.0	**11**	8.2	17	15
TF	7.6	5.9	14	11	20	17
CSM	7.2	5.3	12	8.6	18	14

Table 4.1: Mean and standard deviations, rounded and multiplied by 100 for readability, of the 1000 mean square errors for each of the three noise levels tested with the piecewise cubic function. The smallest value(s) within each column is(are) bold. `BTF` with the hyperparameters $\alpha = 1$ and $\rho = 10^{-2}$ is displayed.

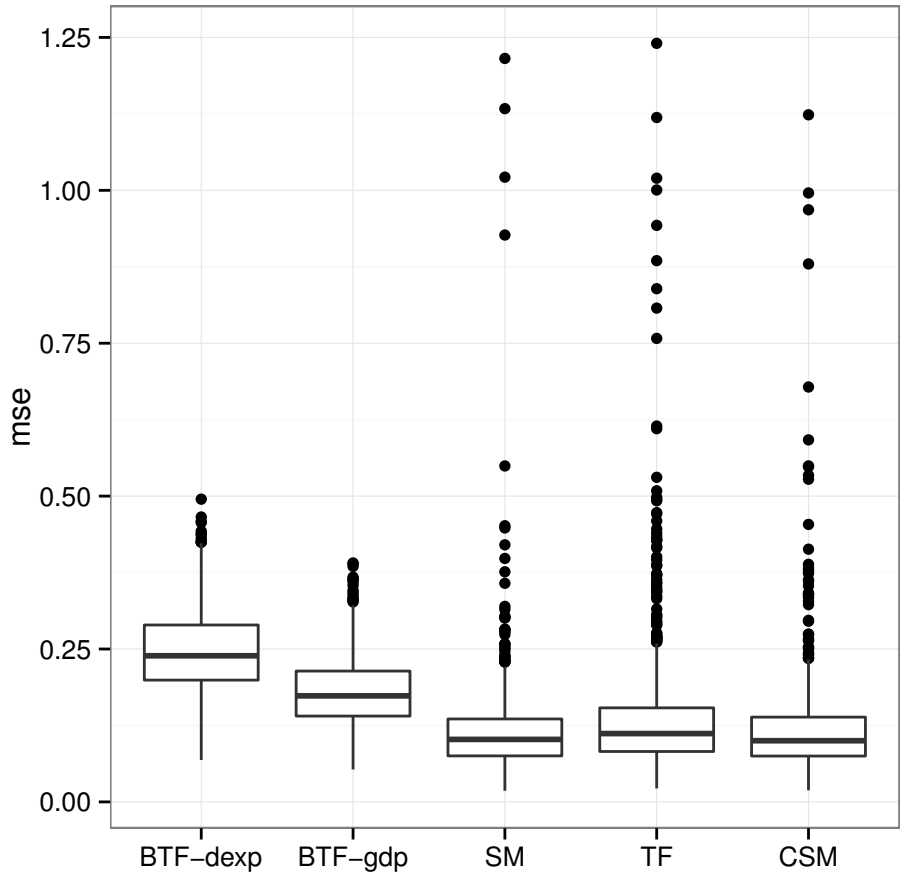

Figure 4.4: Box plots of mses by method for the piecewise cubic function with $\sigma = 1$. `BTF` with the hyperparameters $\alpha = 1$ and $\rho = 10^{-2}$, and `TF` used 10-fold cross validation are displayed.

`BTree` was left out of the plots because the method simply didn't perform as well in these simulations and detracted from the comparisons of interest. While `BTree` provides reasonably small mses, its real trouble stems from the fact that it is not inherently a smoothing technique. Still, `BTree` estimated the variance quite well (not shown). In other contexts, where a smooth fit is not necessary, `BTree` has much to offer beyond what any of these smoothing techniques are able to handle.

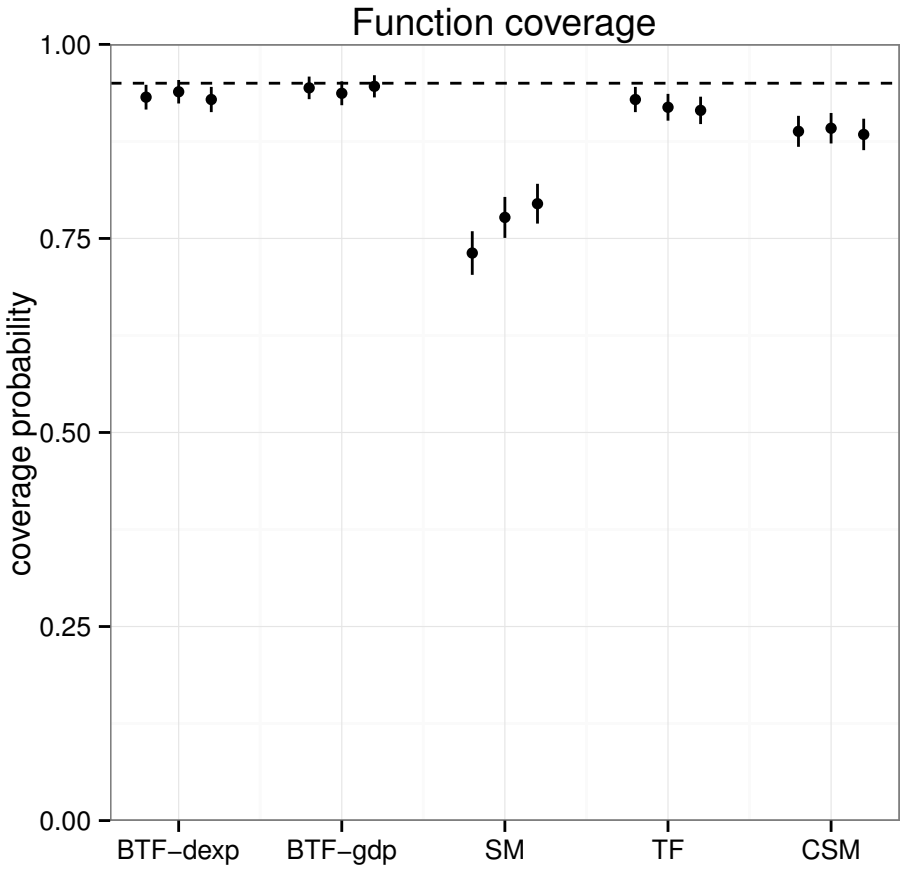

Figure 4.5: Overall function coverage, for the piecewise cubic function, of all the methods at each level of noise, where the `BTF` methods use the hyperparameters $\alpha = 1$ and $\rho = 10^{-2}$, and `TF` used 10-fold cross validation.

Consistent function estimation by all trend filtering methods is seen when we compare coverage probabilities. Figures 4.5 and 4.6 plot mean plus/minus two standard errors, at all noise levels, of function coverage probabilities. The function coverage for the Bayesian trend filtering with `gdp` is quite good, in fact just a bit better for all levels of noise than the frequentist version of trend filtering. We hypothesize that `SM`'s poor function coverage is part of the bootstrapping procedure. In each bootstrap, the penalty parameter is re-estimated. Thus, the coverage probabilities also take into account the variation due to estimating the penalty parameter.

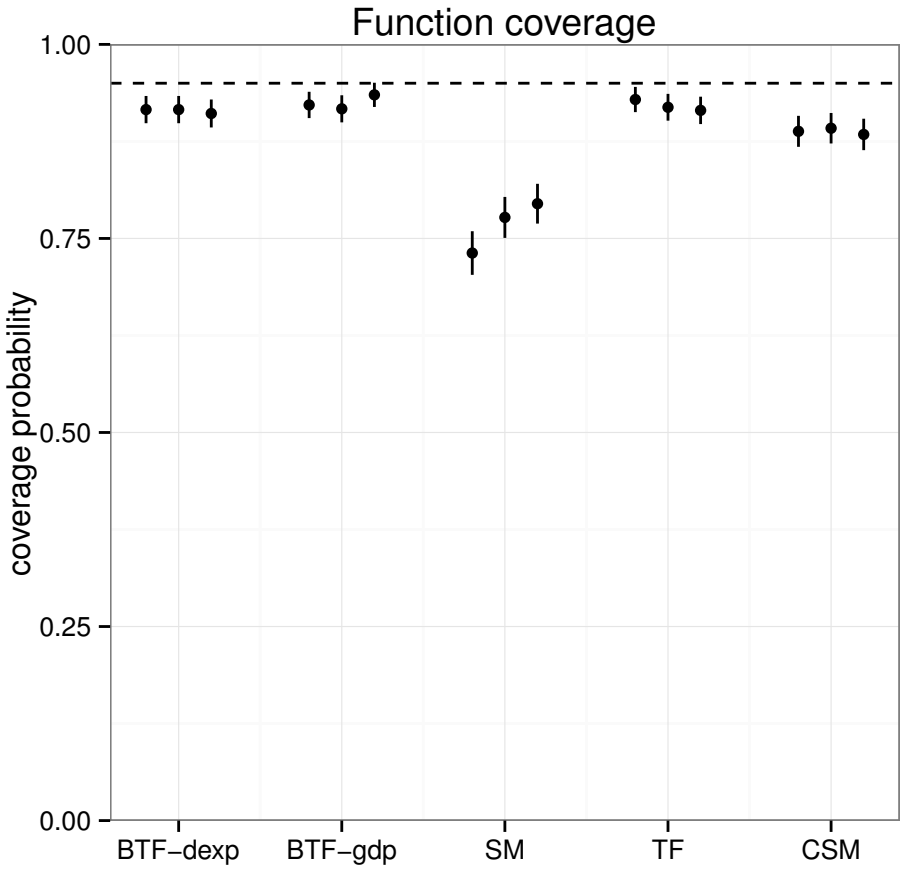

Figure 4.6: Overall function coverage, for the piecewise cubic function, of all the methods at each level of noise, where the `BTF` methods use the hyperparameters $\alpha = 1$ and $\rho = 1$, and `TF` used 10-fold cross validation.

Figure 4.5 is representative of the performance of `BTF` even when the hyperparameter α changes. The differences between Figure 4.5 and Figure 4.6 show that `BTF` with either prior `gdp` or `dexp` is fairly stable with respect to changes in the hyperparameter ρ for the piecewise cubic function. This is not the cases for variance coverage. The variance coverage probability is drastically reduced when the hyperparameter ρ becomes too large. With appropriate choices of ρ, `BTF-gdp` produces accurate coverage probabilities, but it is clear that `BTF-dexp` less accurately covers the true variance as ρ changes.

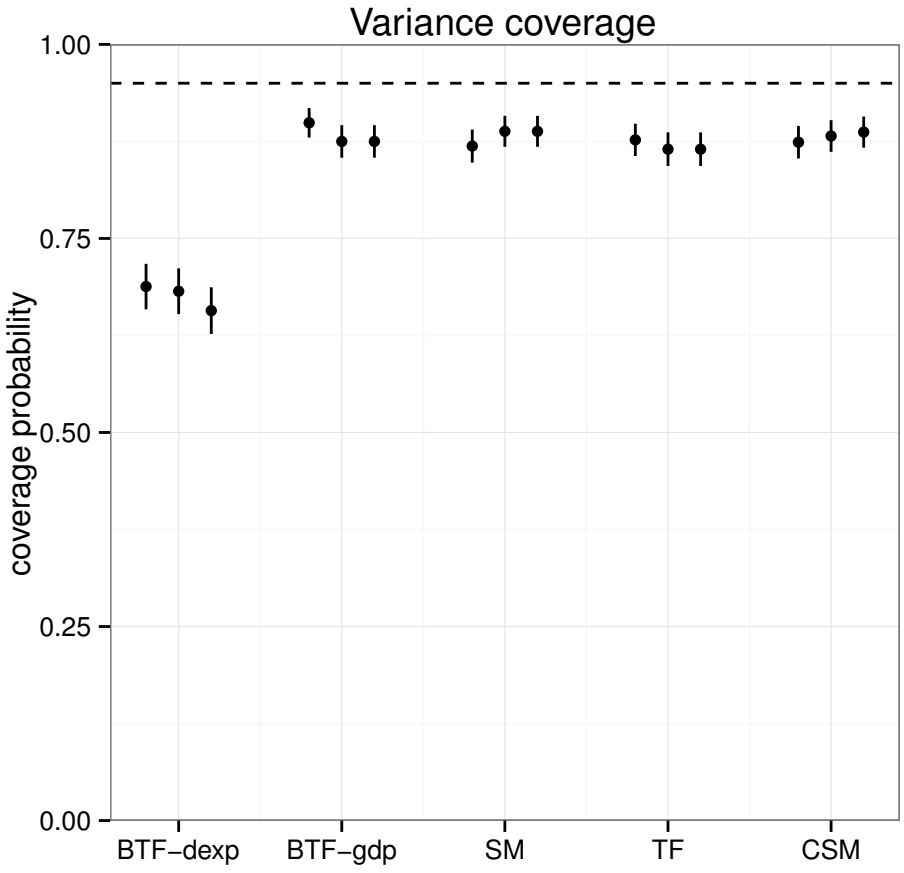

Figure 4.7: Overall variance coverage, for the piecewise cubic function, of all the methods at each level of noise, where the `BTF` methods set the hyperparameters to be $\alpha = 1$ and $\rho = 10^{-2}$ and `TF` used 10-fold cross validation.

For the piecewise cubic function, `BTF-gdp` with reasonably and easily chosen hyperparameters provides arguably the best overall performance, with small average mean squared error and the smallest standard deviation of the 1000 mses. `BTF-gdp` gives roughly equal average mse, but does so with much smaller variance, than the rest of the methods, all while maintaining the closest to nominal function coverage probabilities.

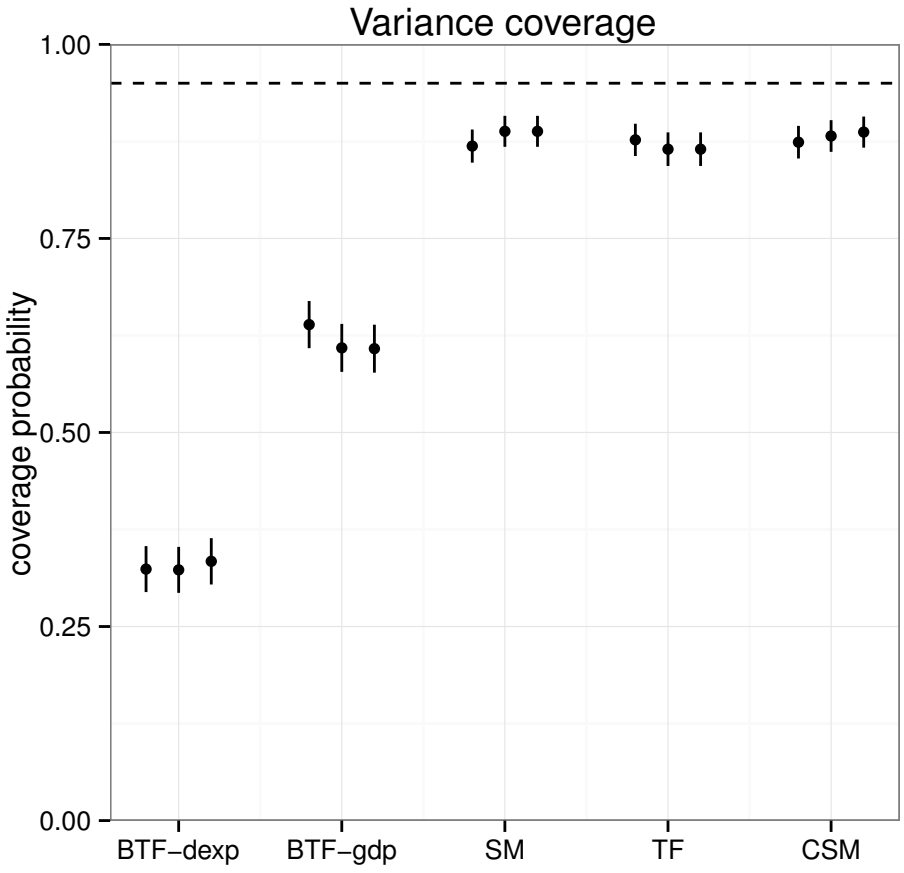

Figure 4.8: Overall variance coverage, for the piecewise cubic function, of all the methods at each level of noise, where the `BTF` methods set the hyperparameters to be $\alpha = 1$ and $\rho = 1$ and `TF` used 10-fold cross validation.

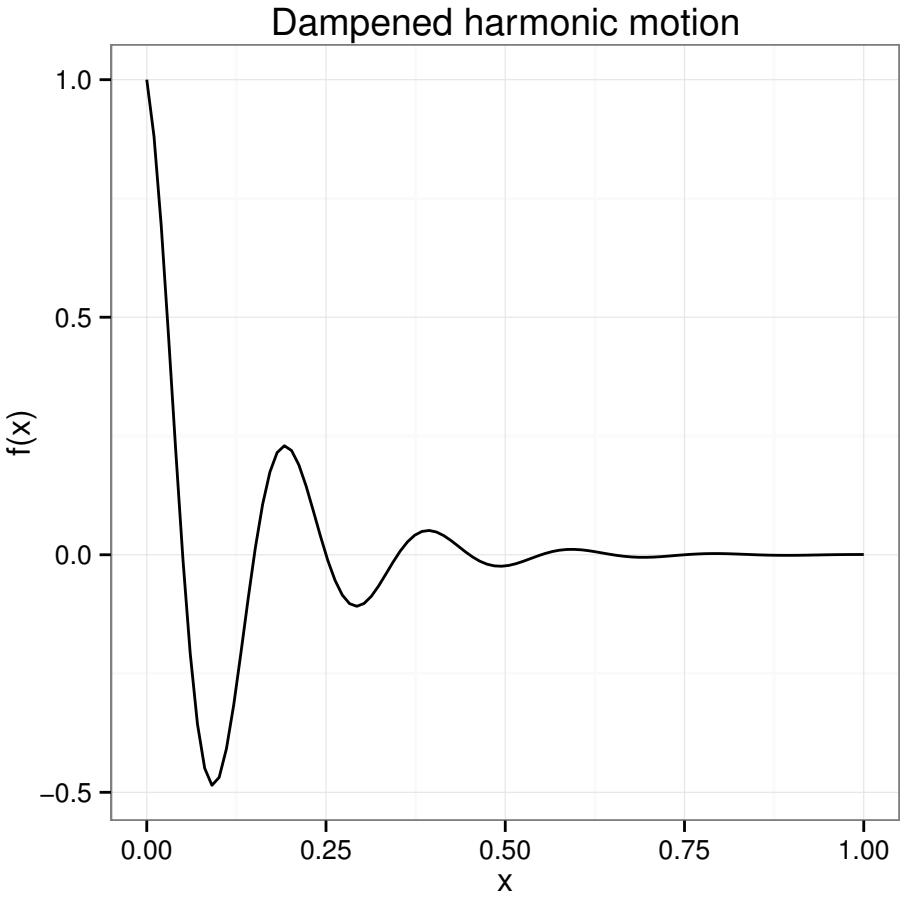

Figure 4.9: Dampened harmonic motion function.

Dampened harmonic motion The second example uses the spatially inhomogeneous function

$$f(x) = \exp\{-7.5x\}\cos(10\pi x),$$

seen in Figure 4.9. Various levels of $\mathcal{N}(0, \sigma^2)$ noise used $\sigma \in \{0.025, 0.05, 0.075\}$. All trend filtering methods' median mses are comparable. In Figure 4.10, where $\sigma = 0.05$, over-fitting is still a problem for `SM`, `TF`, and `CSM`, and becomes worse as the noise increases. `BTree` (not shown) performs notably the worst on the dampened harmonic motion function, but again this is largely due to the fact that it is inherently not a smoothing method and dampened harmonic motion requires heavy smoothing to ensure a small mean squared error.

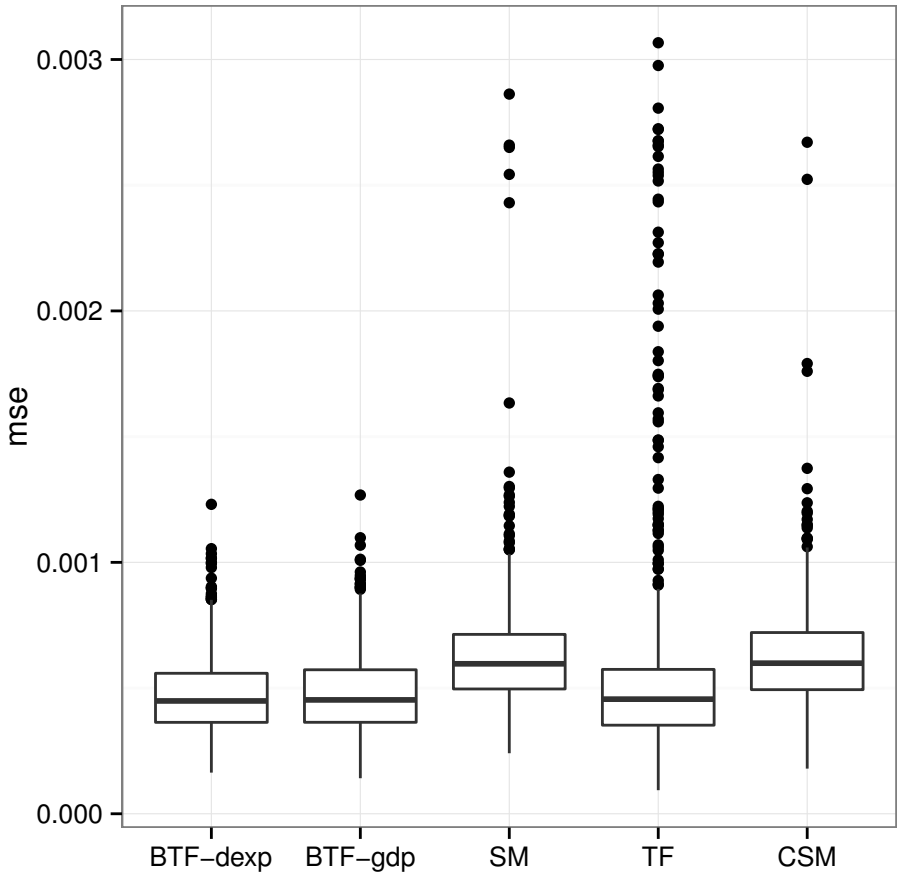

Figure 4.10: Box plots of mses by method for the dampened harmonic motion function with $\sigma = 0.05$. `BTF` with the hyperparameters $\alpha = 1$ and $\rho = 10^{-2}$, and `TF` used 10-fold cross validation are displayed.

To understand better why such (relatively) large mses arise for the methods other than `BTF`, we examine the largest mses of Figure 4.10. Consider `CSM`, since it is the next best method as judged by most largest mses as compared to `BTF`. Figure 4.11 displays the plots of `BTF` and `CSM` over the true dampened harmonic motion function. From Figure 4.11, we see that both methods perform poorly in the right tail, where the true function is relatively flat. Both methods have difficulty smoothing out the right tail after accounting for the curves of the true function closer to zero. Though both `BTF` and `CSM` have difficulty flattening out the right tail, it is clear that `BTF` adapts to the local fluctuations (or lack thereof) than does `CSM`.

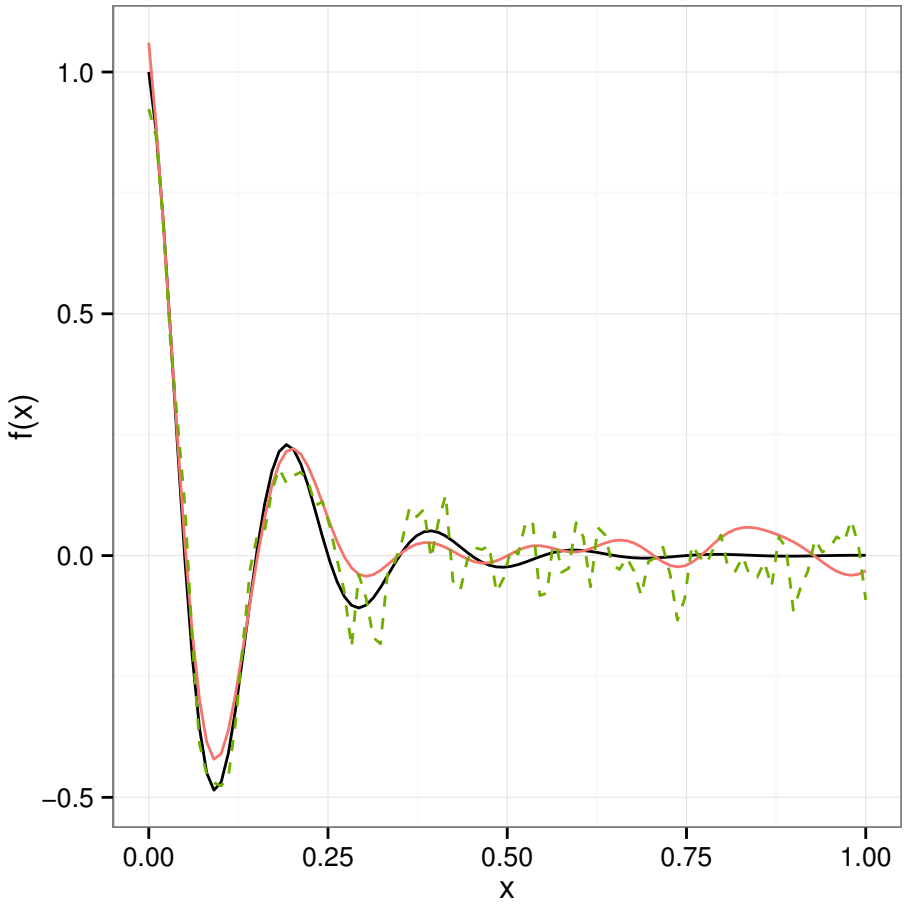

Figure 4.11: The two worst fits of `BTF` (solid red) and `CSM` (dash green) as judged by largest mse from Figure 4.10.

Generally with the dampened harmonic motion function, the `BTF` methods with either prior perform well, both providing nearly the same, and also the smallest, average mse. Further, the Bayesian trend filtering methods provide the smallest standard deviations of the 1000 mses of all the methods tested. These general results hold across all the levels of noise considered, as shown in Table 4.2.

	$\sigma = 0.025$		$\sigma = 0.05$		$\sigma = 0.075$	
method	mean	sd	mean	sd	mean	sd
BTF-dexp	**1.3**	**0.39**	**4.7**	**1.5**	**9.9**	**3.1**
BTF-gdp	**1.3**	**0.39**	4.8	1.6	11	3.9
SM	2.0	0.55	6.3	2.3	12	4.9
TF	1.5	0.92	5.4	3.9	11	7.2
CSM	1.9	0.53	6.2	2.0	12	4.2

Table 4.2: Mean and standard deviations, rounded and multiplied by 100 for readability, of the 1000 mean square errors for each of the three noise levels tested with the dampened harmonic motion function. The smallest value(s) within each column is(are) bold. `BTF` with the hyperparameters $\alpha = 1$ and $\rho = 10^{-2}$ is displayed.

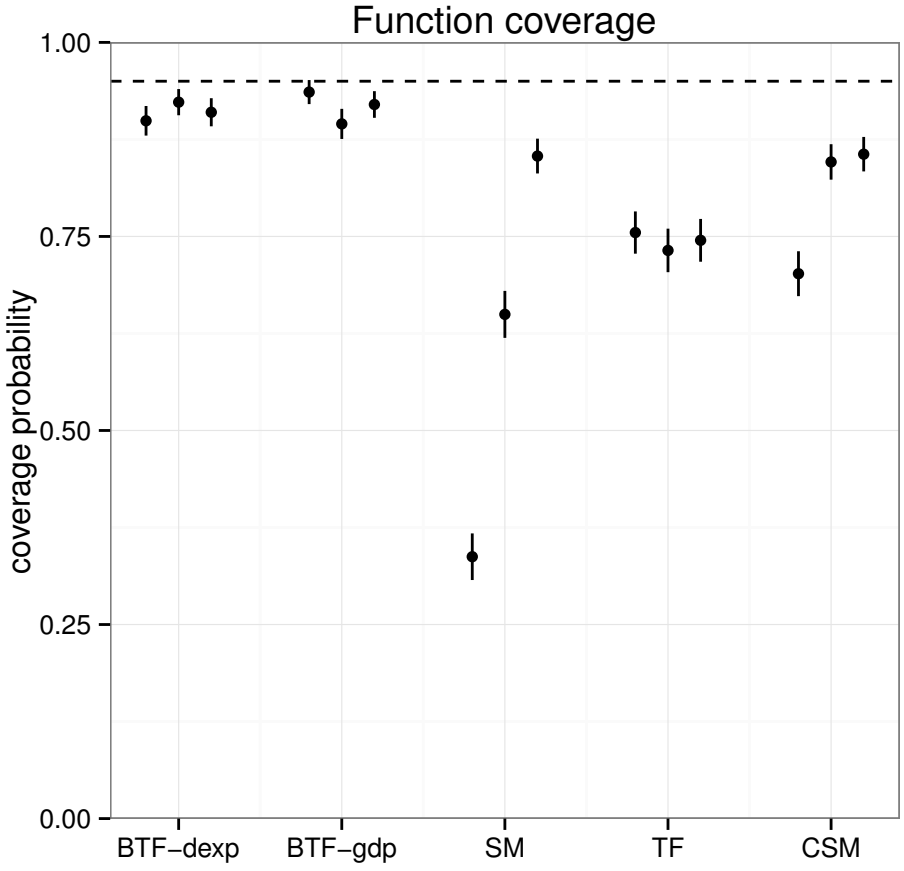

Figure 4.12: Overall function coverage, for the dampened harmonic motion function, of all the methods at each level of noise, where the `BTF` methods set the hyperparameters to be $\alpha = 1$ and $\rho = 10^{-2}$ and `TF` used 5-fold cross validation.

Figures 4.12 and 4.13 show mean function coverage probabilities plus/minus two standard errors for the dampened harmonic motion function. Here, function estimation proved to be more difficult for all the methods than it was for the piecewise cubic function above. This is likely do to the fact that we are measuring overall function coverage on a quite spatially inhomogeneous function. `BTF-gdp`'s function coverage appears to slightly decline as the variance of the noise increases and as the hyperparameter ρ decreases. In all cases, function coverage for the `BTF` methods with either prior is comparable to all the other methods. There is a sharp decline in the function coverage for the smoothing spline methods when the noise levels are low.

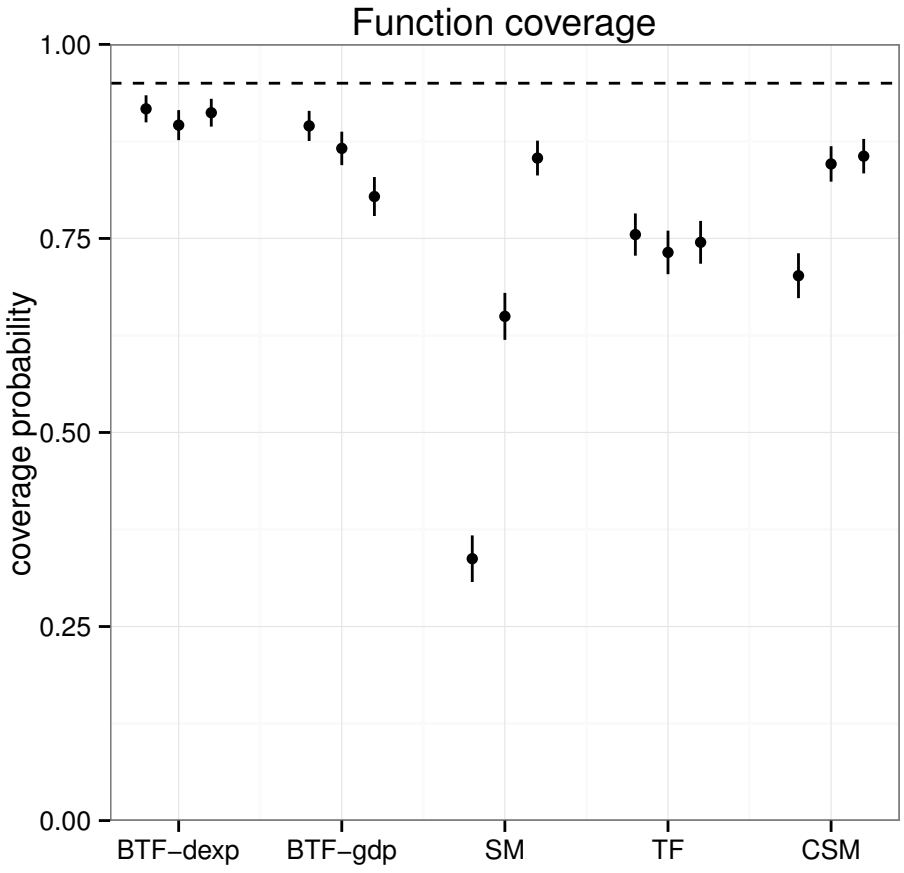

Figure 4.13: Overall function coverage, for the dampened harmonic motion function, of all the methods at each level of noise, where the `BTF` methods set the hyperparameters to be $\alpha = 1$ and $\rho = 1$ and `TF` used 5-fold cross validation.

For the dampened harmonic motion function the Bayesian trend filtering methods were clearly better than the frequentist trend filtering method. This is a starker picture from the piecewise cubic function where the Bayesian method and the frequentist version were more comparable. However, different from the previous function, the `dexp` prior appears better at covering the true function when applied to dampened harmonic motion, especially as the hyperparameter ρ decreases. This may be due to the sharper tails of the double exponential prior distribution compared to the generalized double Pareto tails. The sharper tails more heavily penalize wiggliness, which favors the flat right tail. Though, `dexp` is significantly worse at covering the true variance, in fact much worse at covering the variance than `gdp`.

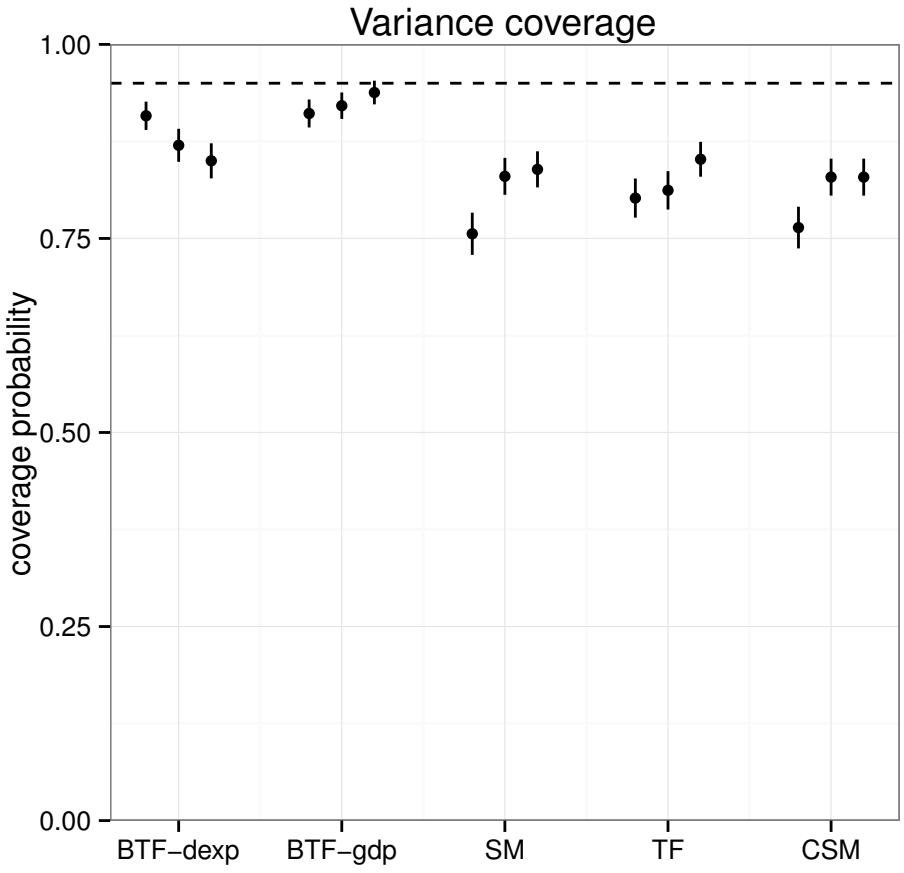

Figure 4.14: Overall variance coverage, for the dampened harmonic motion function, of all the methods at each level of noise, where the `BTF` methods set the hyperparameters to be $\alpha = 1$ and $\rho = 10^{-2}$ and `TF` used 10-fold cross validation.

The picture of how well the Bayesian trend filtering methods perform on the dampened harmonic motion function is less clear than for the piecewise cubic function. At certain values of the hyperparameter ρ, the variance coverage is significantly worse than the other methods. Still though, when judged by small mse and function coverage, `BTF-gdp` proves to be a strong competitor even with semi-arbitrary choices of the hyperparameters.

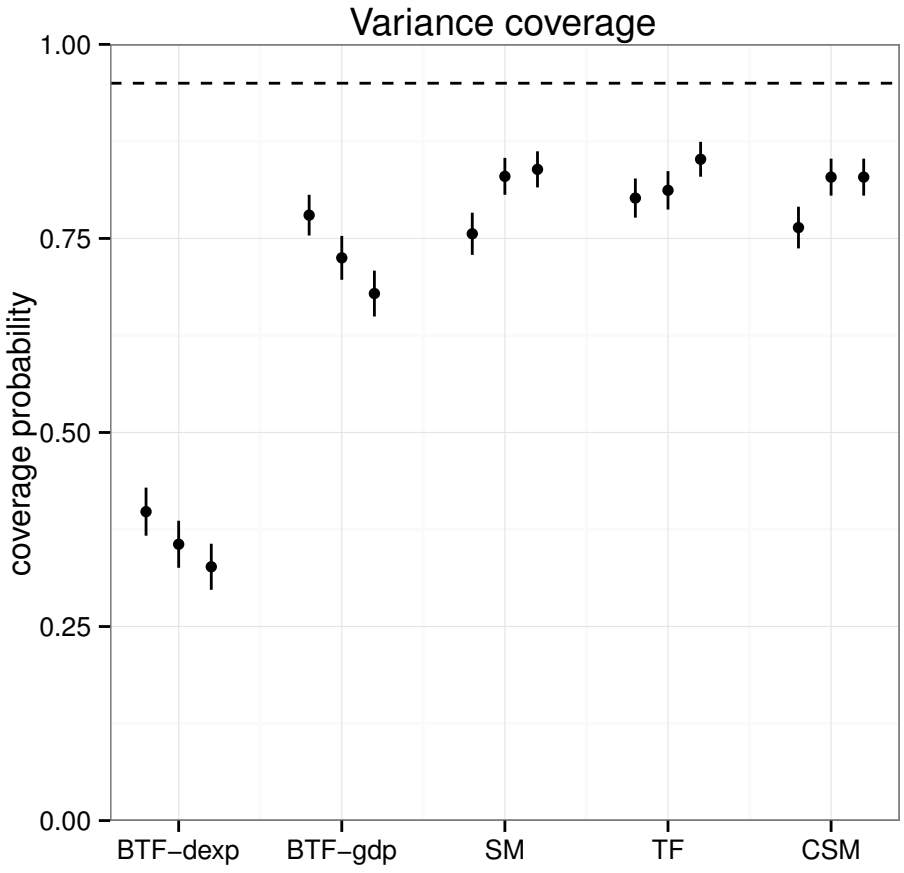

Figure 4.15: Overall variance coverage, for the dampened harmonic motion function, of all the methods at each level of noise, where the `BTF` methods set the hyperparameters to be $\alpha = 1$ and $\rho = 1$ and `TF` used 10-fold cross validation.

4.2 Real Data

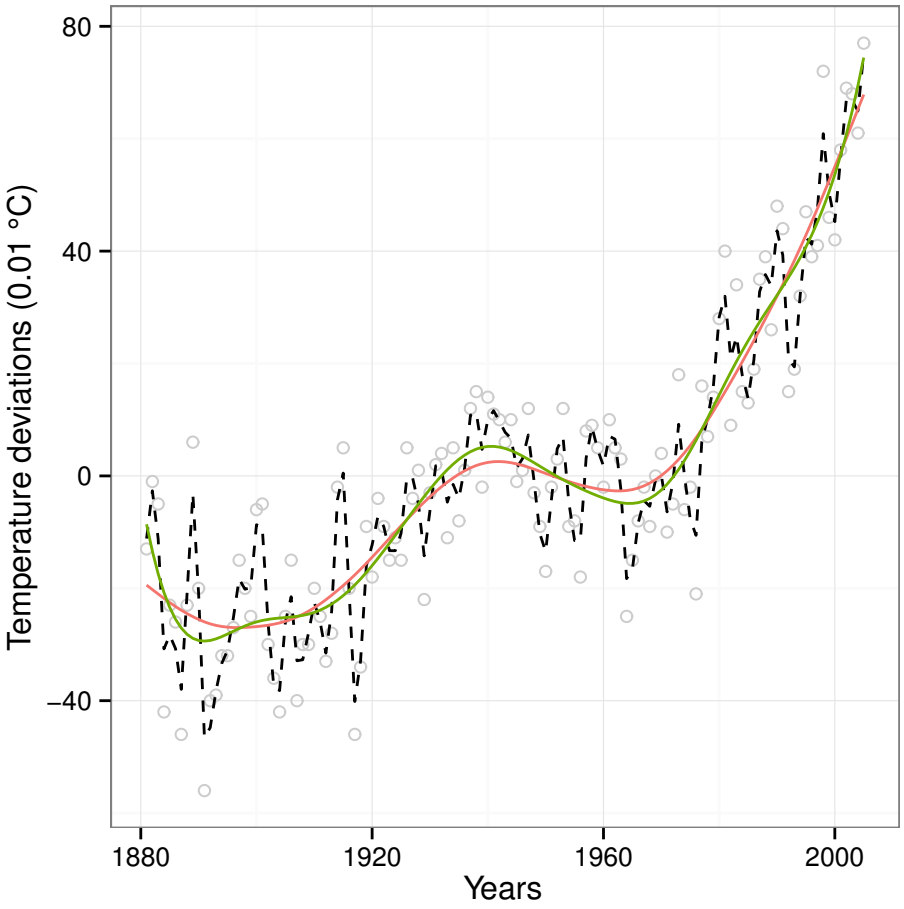

Figure 4.16: Methods `CSM` (dash), `BTF-gdp` (solid green) with $\alpha = 1$ and $\rho = 10^{-2}$, and cubic smoothing spline (solid red) with AR(1) error structure fit to the global mean surface temperature deviations data.

Surface Temperatures The global mean surface temperature deviations data consist of yearly measurements, $[1881, 2005]$, in units of $0.01\,^{\circ}$C. The global mean surface temperature data proves to be a problem for both 5- and 10-fold cross validation used by `TF`. Figure 4.17 shows that trend filtering (dash black) nearly fits every data point, and that `BTree` (solid red) arguable provides a reasonable, albeit not smoothed, fit. Figure 4.16 shows that `BTF` smooths these data quite a lot compared to the cubic smoothing spline method `CSM`. It is difficult to say whether or not the smoothing done by `BTF` is too much. `SM` (not shown) provides a very similar fit to that of `BTF-gdp`. The credible intervals (not shown) from `BTF-gdp` completely contain the fit from `SM`.

Interestingly, a cubic smoothing spline with an autoregressive one, denoted AR(1), error structure provides a fit very close to that of `BTF-gdp`. The `CSM`-AR(1) fit is completely contained within the credible intervals (not shown) of `BTF-gdp`. It should be emphasized though, that these two methods hypothesize different underlying structures. The cubic smoothing spline is correlating the error structure of the observations $\mathbf{y}$, while `BTF` is correlating adjacent values of the underlying function itself and stipulating no correlation across the observations. From the estimates of these methods, it is unclear which model is a more appropriate fit to these data.

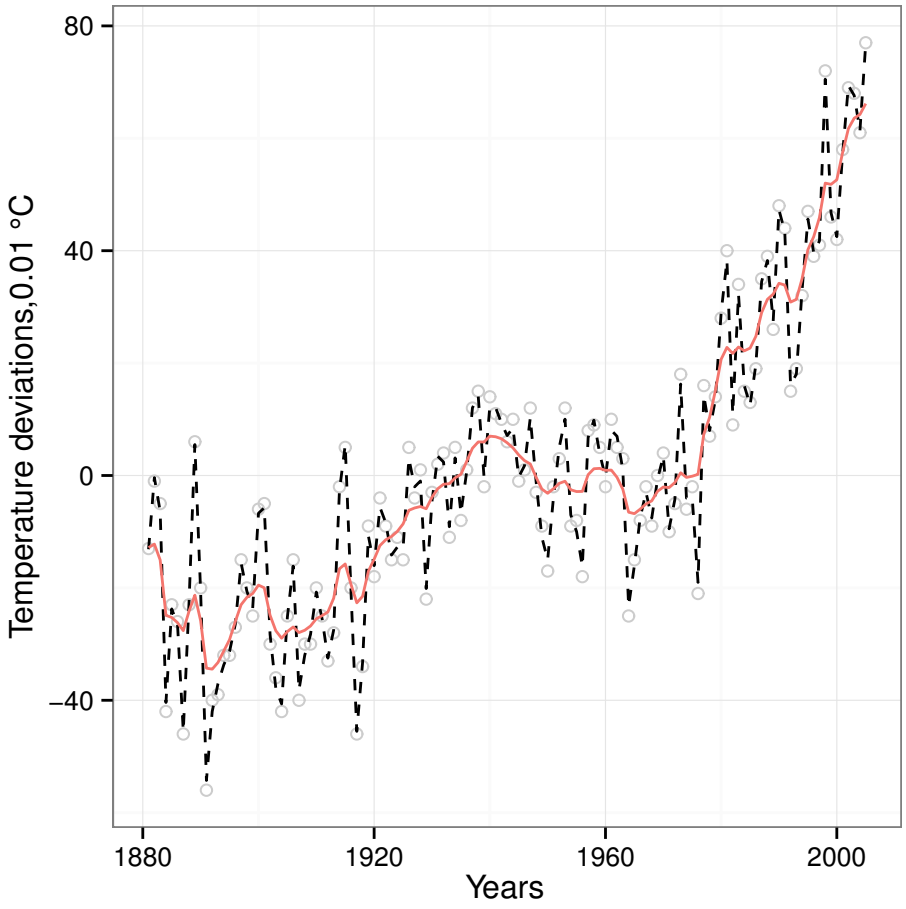

Figure 4.17: Methods `BTree` and `TF` fit to the global mean surface temperature deviations data, with `TF` using 10-fold cross validation.

The trace plots in Figure 4.18 show that convergence is quite fast. All convergence diagnostics are shown for `BTF-gdp` with $\alpha = 1$ and $\rho = 10^{-2}$, but similar plots are produced for all levels of the hyperparameters tested. After discarding a burn-in of 1000 iterations, Figure 4.19 overlays density plots of the first 4000 and remaining 5000 of the 10000 samples from the posterior. Since the density plots have similar location and shape, this shows evidence of rapid convergence. The density plots show that, at least for the posterior of the function evaluations, 5000 total iterations is generally sufficient.

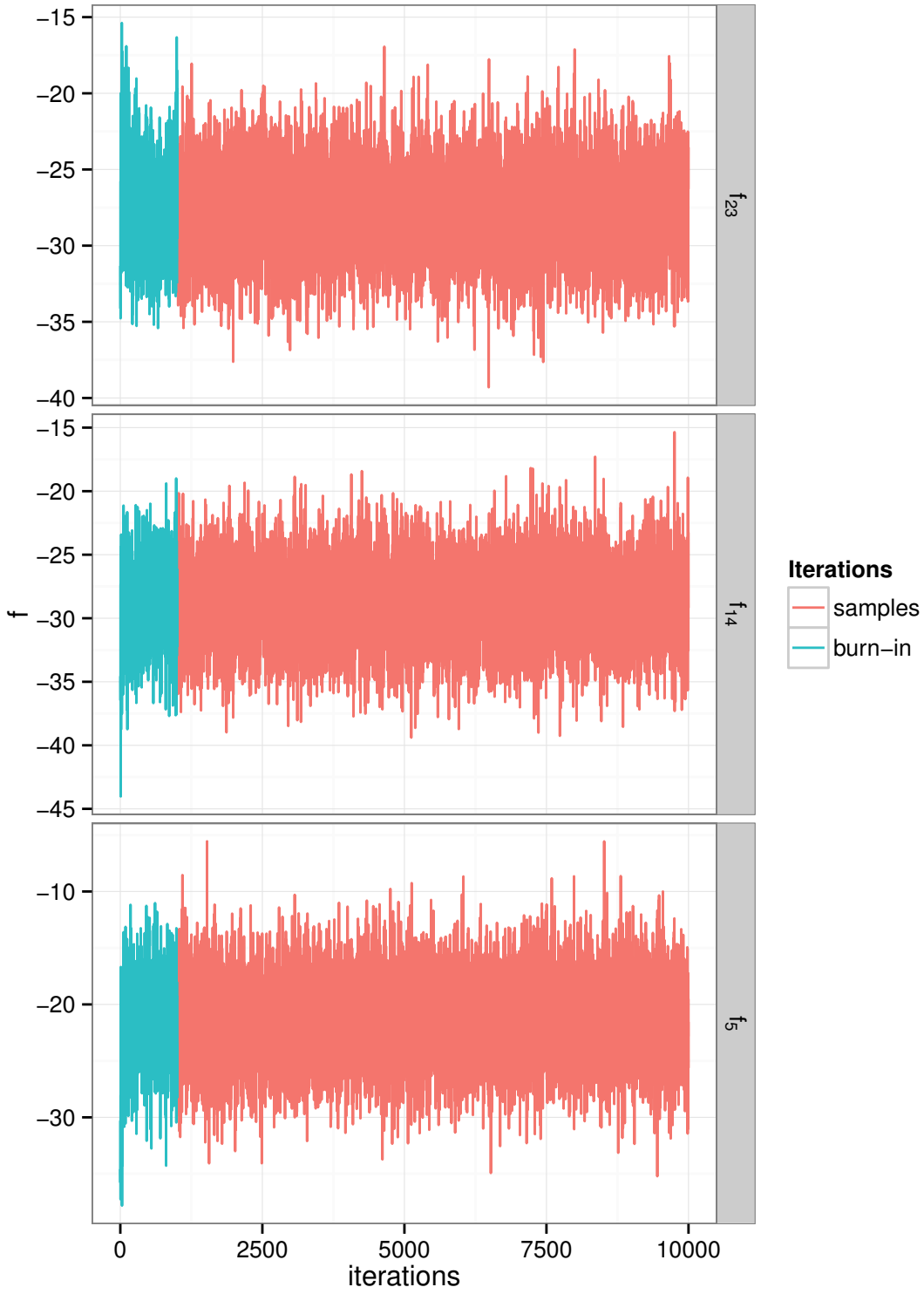

Figure 4.18: Trace plots of three randomly selected function evaluations for the global mean surface temperature deviations data.

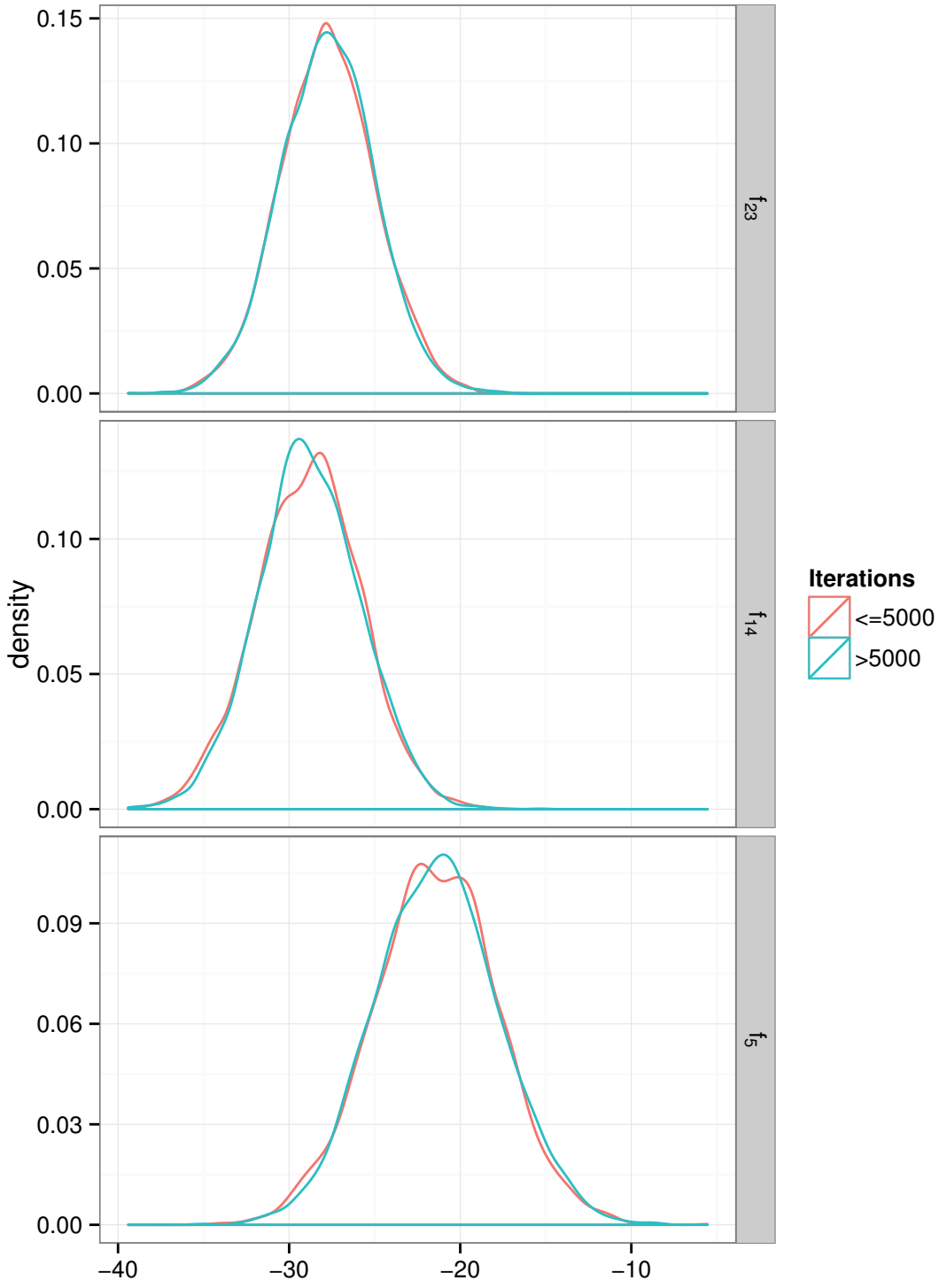

Figure 4.19: Density plots of three randomly selected function evaluations for the global mean surface temperature deviations data.

Trace and density plots of σ^2 and λ provide some evidence of the convergence rates for these other important parameters; see Figures 4.20 through 4.23. Since the two density plots, for σ^2 in

Figure 4.23, are quite similar, we have some indication that this full conditional mixes well. The effective sample size for σ^2 is $n_{eff} = 3209$. The penalty parameter λ appears to mix the slowest, and has the smallest effective sample size, $n_{eff} = 188$, for the global temperature data.

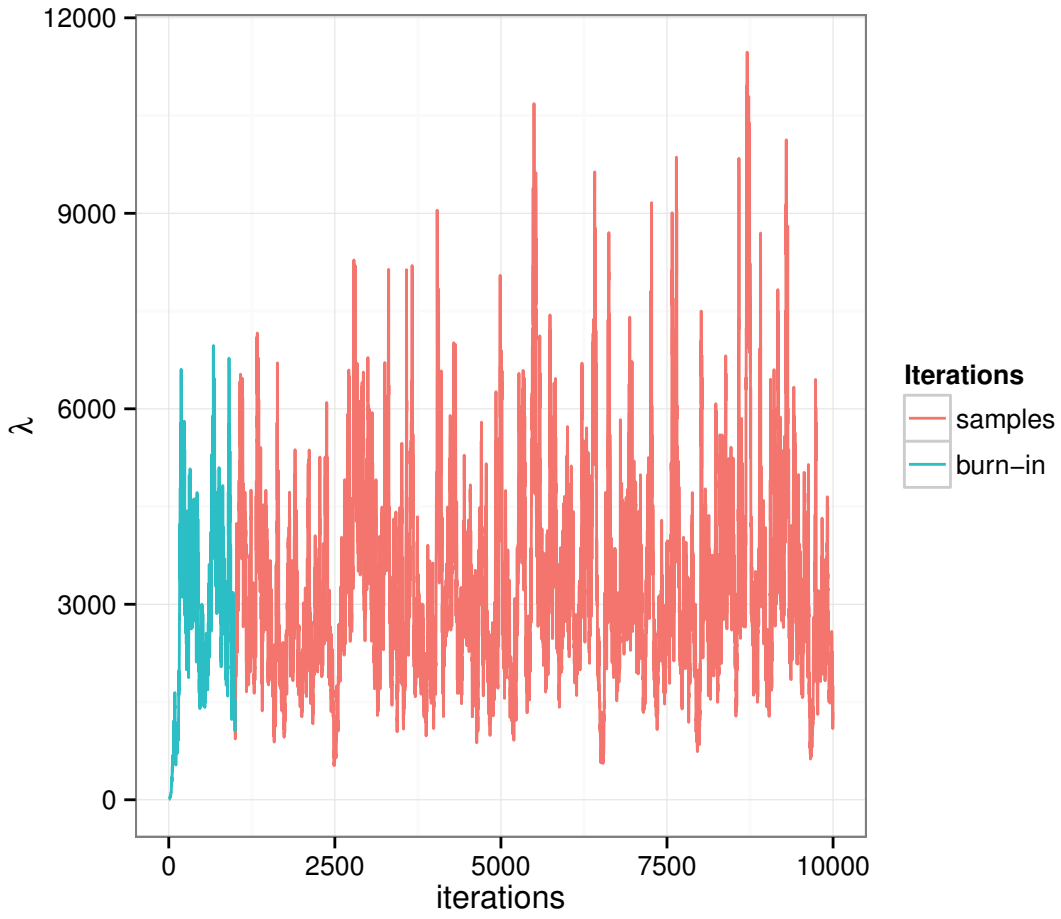

Figure 4.20: Trace plot of the penalty parameter λ for the global mean surface temperature deviations data.

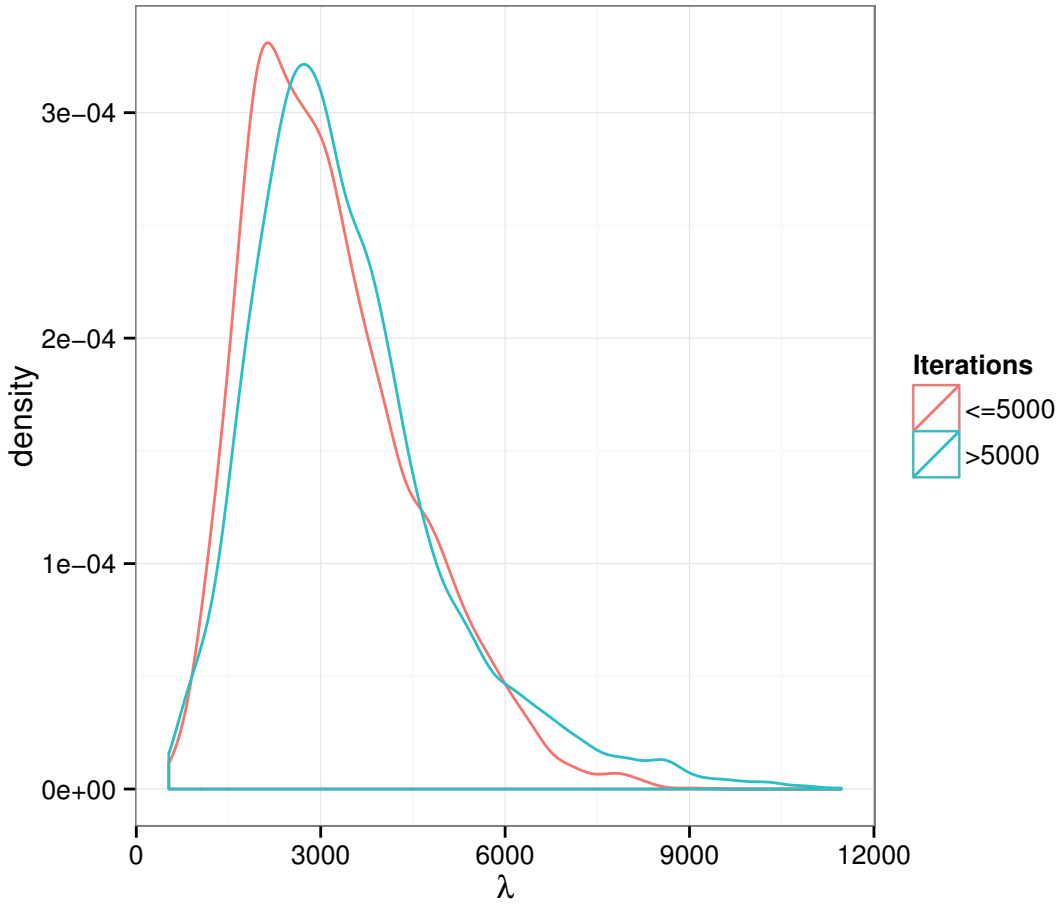

Figure 4.21: Density plot of the penalty parameter λ for the global mean surface temperature deviations data.

It is to the practitioner's benefit that the function evaluations, generally the parameters of most interest when fitting `BTF`, mix more quickly than the other parameters. As mentioned in Section 3.6 a possible strategy to speed up Bayesian trend filtering is to sample from the posterior for f every mth iteration. Because the matrix inversion involved with sampling from the posterior of f comprises the vast majority of the computational complexity involved in sampling, such a strategy could greatly benefit the speed of fitting `BTF`.

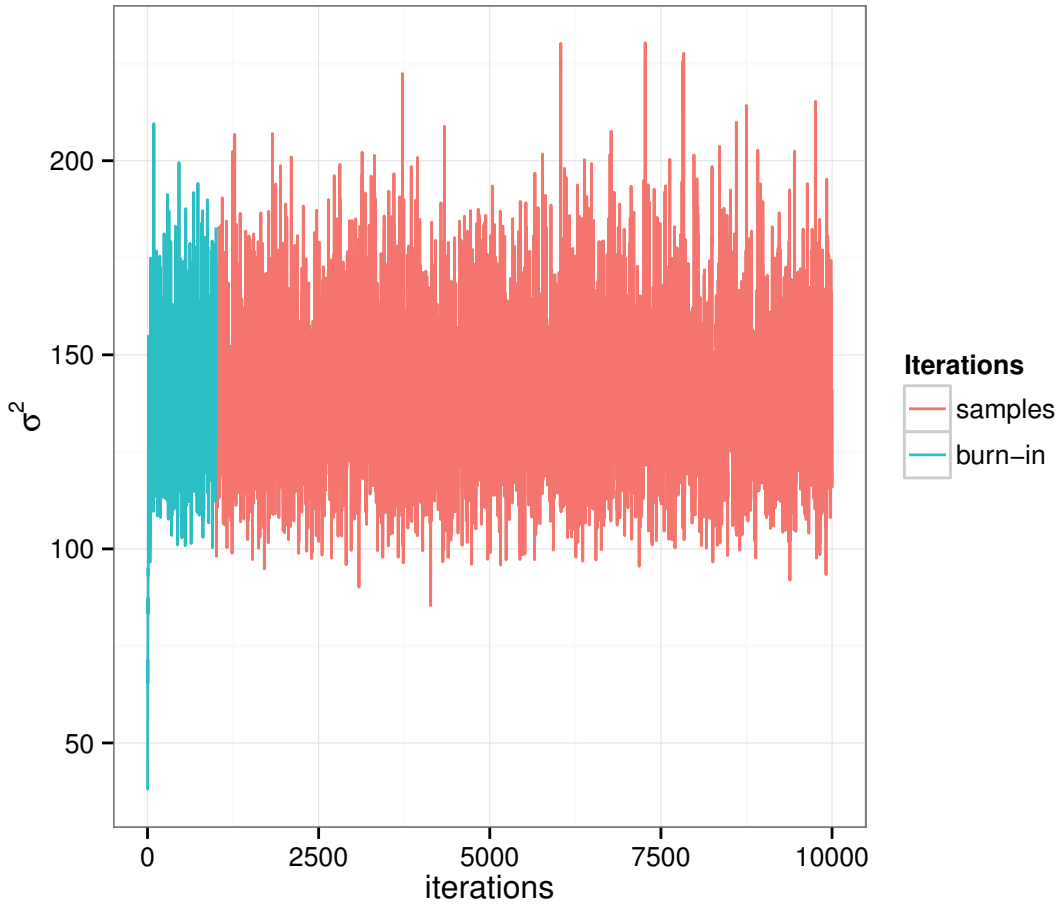

Figure 4.22: Trace plot of the estimate of the variance parameter σ^2 for the global mean surface temperature deviations data.

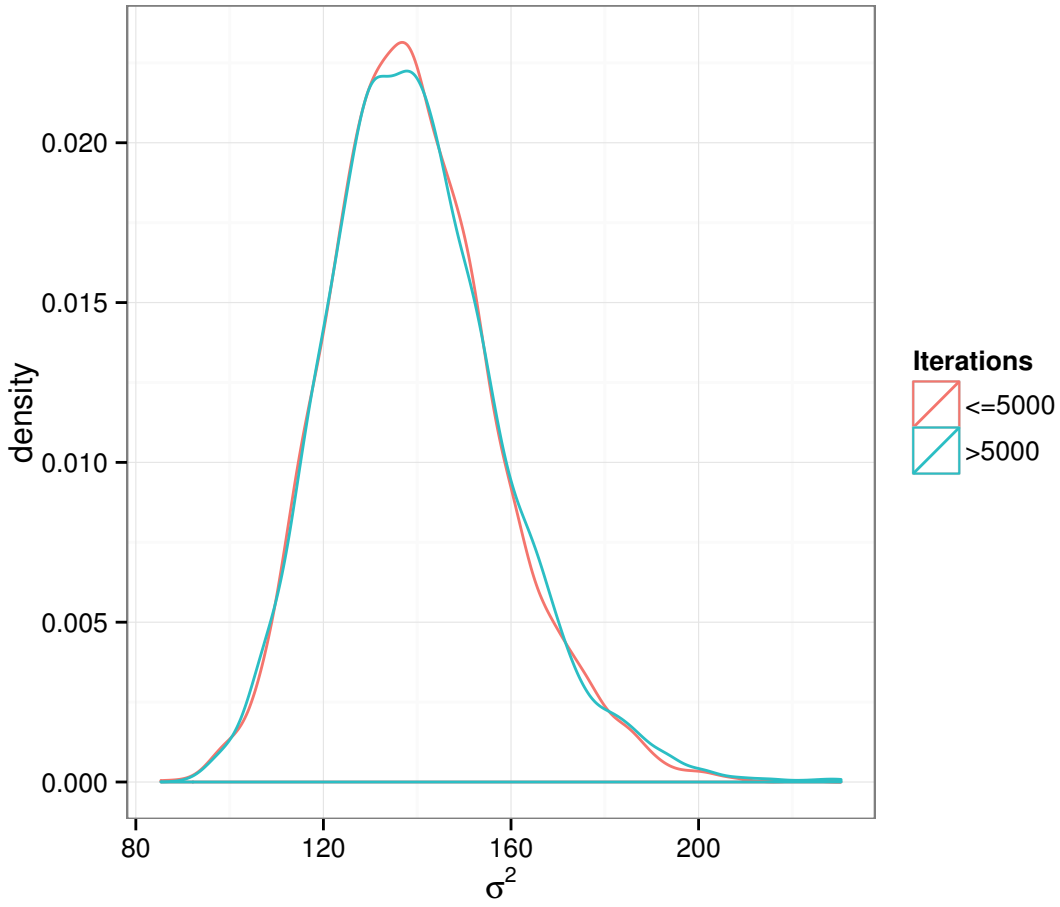

Figure 4.23: Density plot of the estimate of the variance parameter σ^2 for the global mean surface temperature deviations data.

Sunspots The second applied data set we investigate is the sunspot data. Since both `TF` and `BTF` are quite a bit slower than `SM` and `CSM`, we dropped data prior to 1980, leaving 402 observations. Figure 4.24 shows the fits of `BTF-gdp` (solid greed), `CSM` (solid red), and `CSM`-AR(1) (dash black). The Bayesian trend filtering and cubic smoothing spline with AR(1) errors, smooth estimates quite a bit more than any other method. `BTF-gdp` and `CSM`-AR(1) provide very similar fits. In Figure 4.25 we see the fit from `TF` (dash black) using 10-fold cross validation and the fit from `BTree` (solid red), both of which provide a much more noisy fit to the data than do the other methods; 5- and 10-fold cross validation under `TF` yield visually identical results.

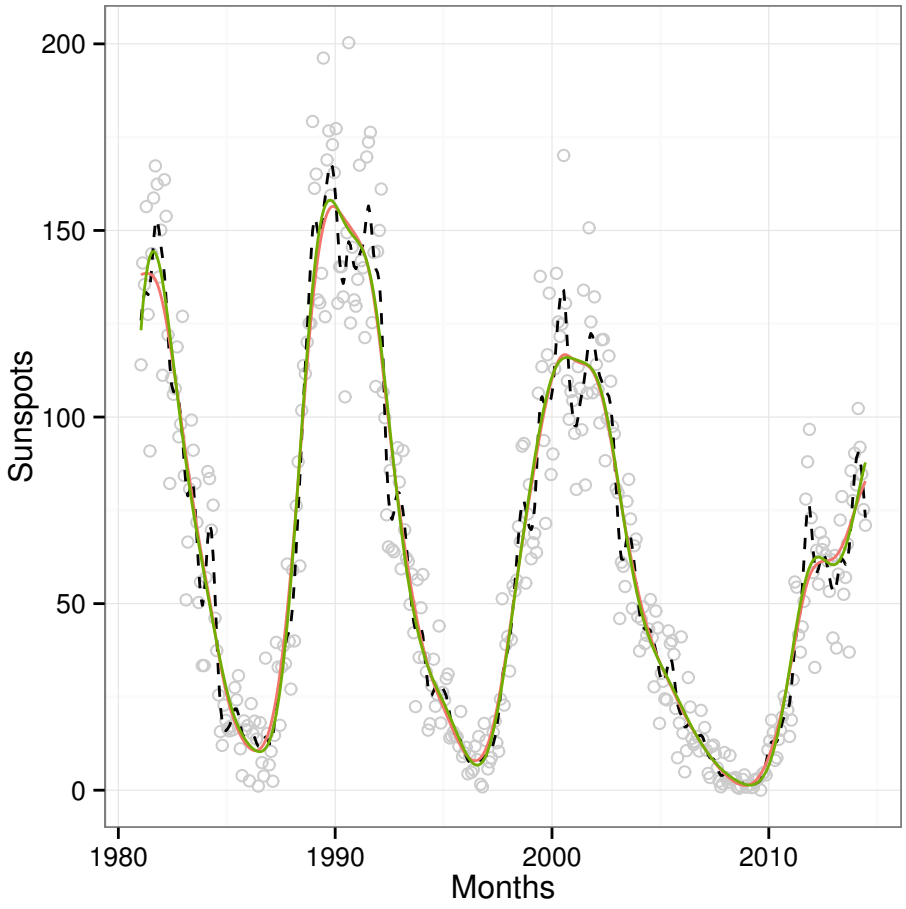

Figure 4.24: Methods `CSM` (dash), `BTF-gdp` (solid green) with $\alpha = 1$ and $\rho = 10^{-2}$, and cubic smoothing spline with an AR(1) error structure fit to the sunspot data.

As before, the cubic smoothing spline with an AR(1) error structure and `BTF-gdp` methods offer very similar results. The predicted values between these two methods leave little to discern which method is more appropriate for these data. It would be interesting to discuss this issue with somebody more familiar with the underlying physical process of sunspots.

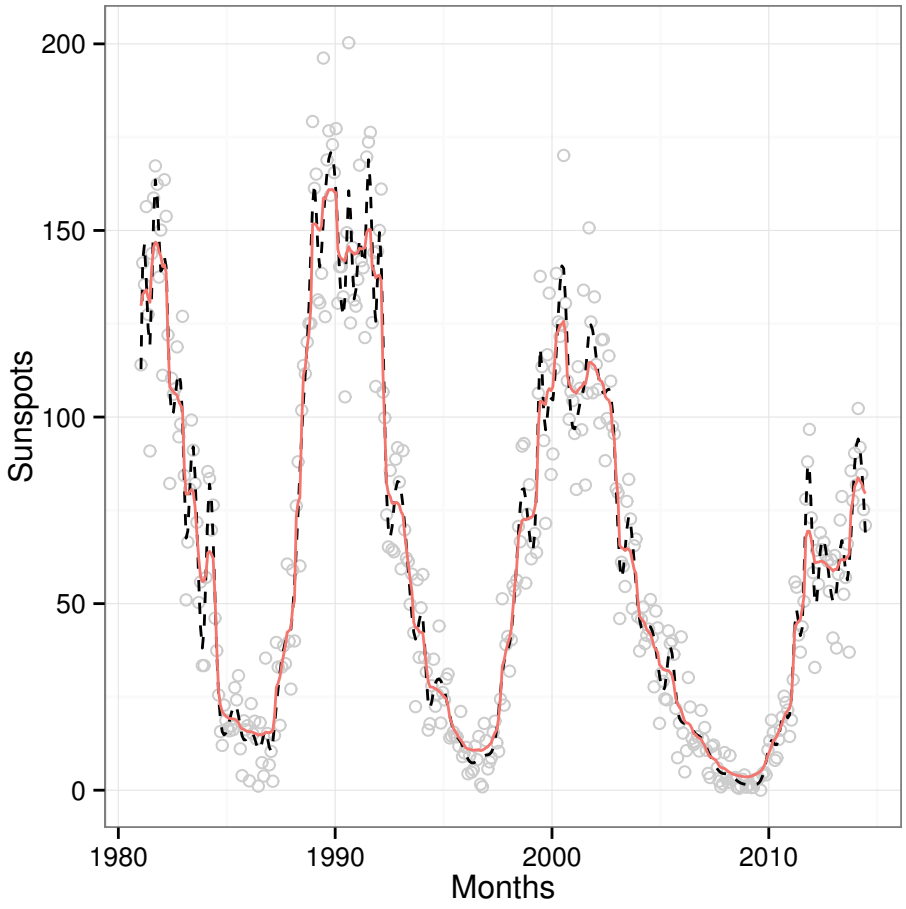

Figure 4.25: Methods `BTree` (solid) and `TF` (dash) fit to the sunspot data, with `TF` using 10-fold cross validation.

The trace plots with respect to the sunspot data tell much of the same story as before. Mixing speeds for the function evaluations f and the variance σ^2 are quite quick. The range of the effective sample sizes for the function evaluations is 314 to 9409 with a median of 3688. The variance σ^2 also mixes quite well with an effective sample size $n_{eff} = 1761$. On the other hand, mixing was a bit slower for the penalty parameter λ; effective sample size for λ was $n_{eff} = 92$. An effective sample size of 92 is unfortunately small, especially as we are performing 10^4 iterations.

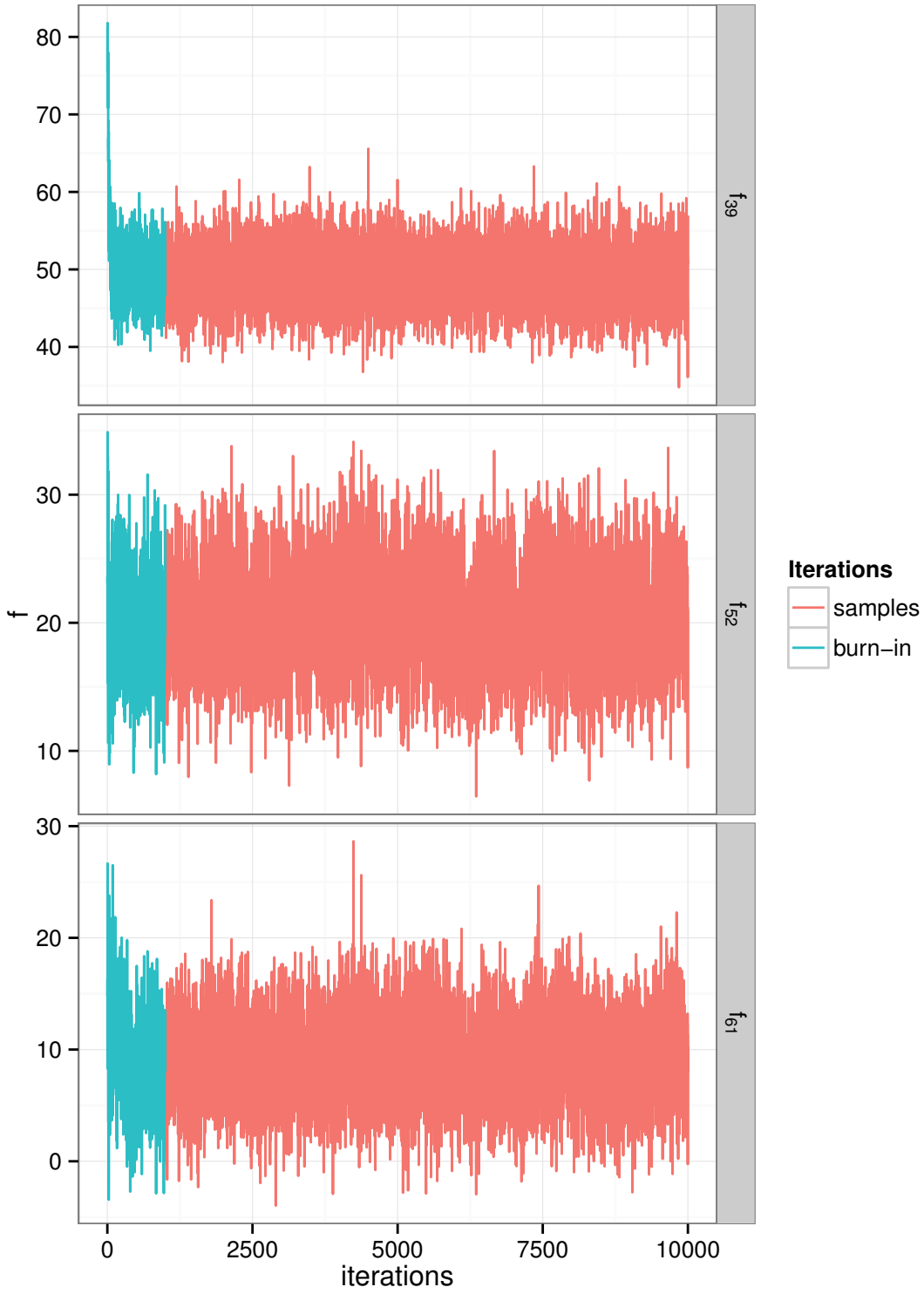

Figure 4.26: Trace plots of three randomly selected function evaluations for the global mean surface temperature deviations data.

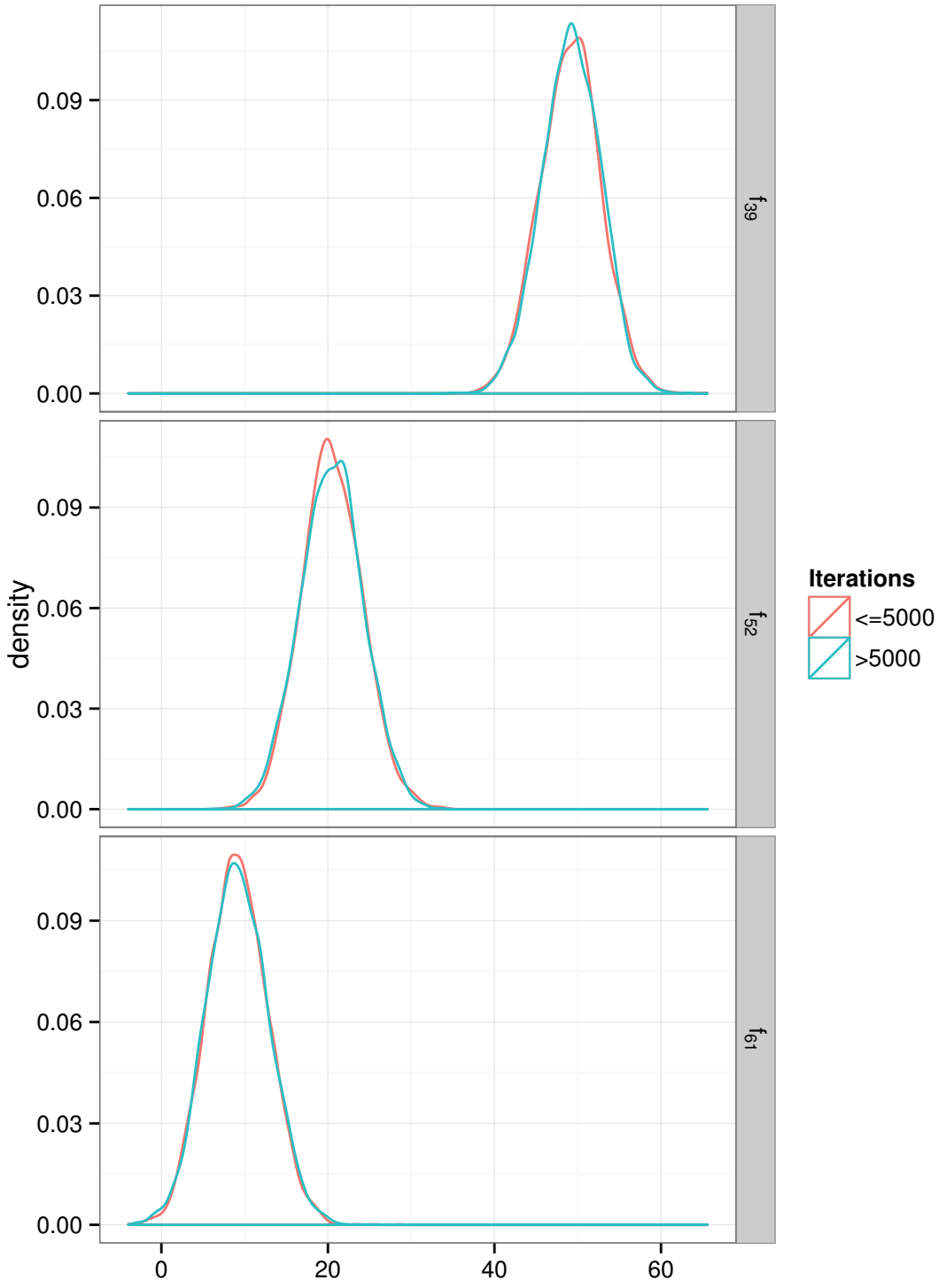

Figure 4.27: Density plots of three randomly selected function evaluations for the sunspot data.

Effect of Speeding Up BTF There is significant variation amongst the computation times for the methods fit here. By far the fastest methods are the cubic smoothing splines, `SM` and `CSM`, as they

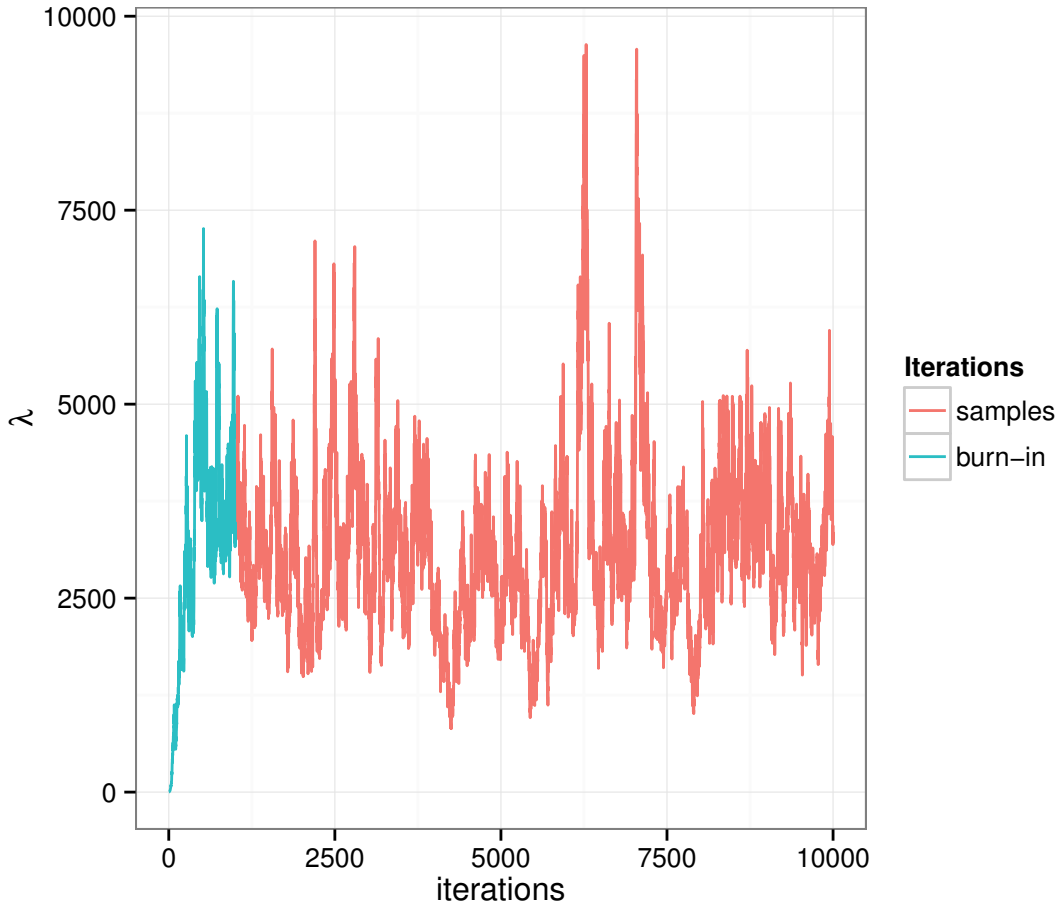

Figure 4.28: Trace plot of the penalty parameter λ for the sunspot data.

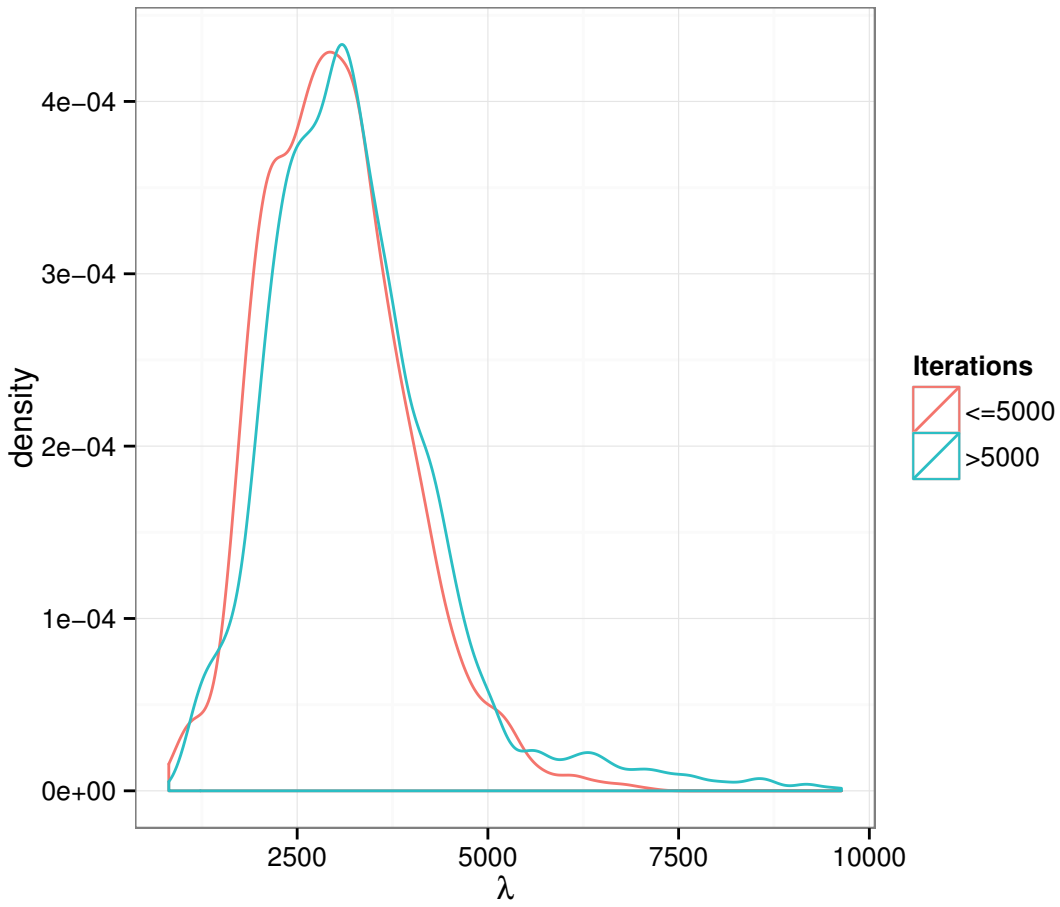

Figure 4.29: Density plot of the penalty parameter λ for the sunspot data.

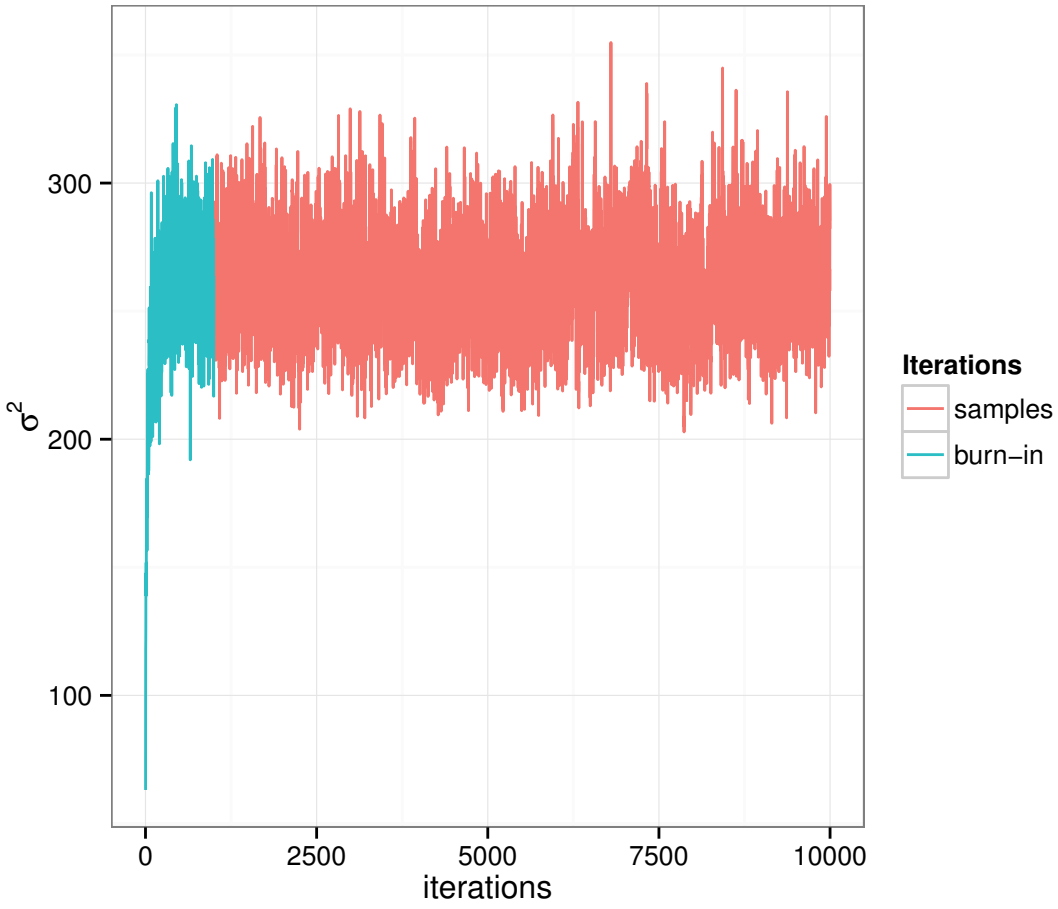

Figure 4.30: Trace plot of the estimate of the variance parameter σ^2 for the sunspot data.

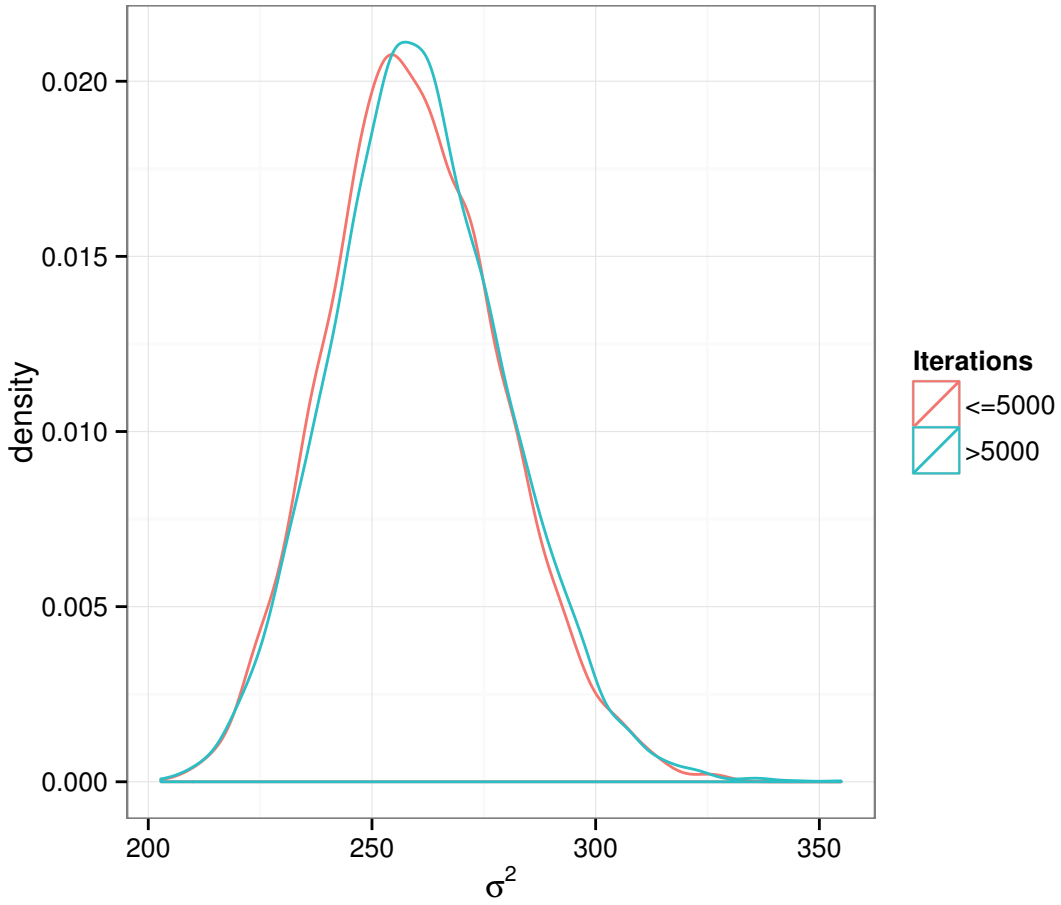

Figure 4.31: Density plot of the estimate of the variance parameter σ^2 for the sunspot data.

do not require iterative solutions and instead rely on matrix calculus to estimate the parameters of interest. The slowest methods are the trend filtering methods. Table 4.3 contains run times for each method for each real data set. Two times for frequentist trend filtering are given, one for each of 5- and 10-fold cross validation. Bayesian trend filtering is known to scale very poorly. This is due to the matrix inversion that is required for each sample from the full conditional for f. Sampling from `BTF` methods every 2nd iteration drops the computation almost in half, as expected. With respect to the global mean temperature data, the run time dropped almost in half from 7 seconds down to 4 seconds with thinned sampling from $[f|\cdot]$. We see similar results relative to the sunspots data. Sampling from $[f|\cdot]$ only every 2nd iteration decreased the run time from 98 to 51 seconds.

method	temperature (n=125)	sunspots (n=402)
BTF	7	98
BTF (m=2)	4	51
SM	0.001	0.003
TF (k=5)	10	46
TF (k=10)	20	94
CSM	0.2	7
CSM (AR(1))	0.5	13
BTree	9	19

Table 4.3: Computation times in seconds for each method against each real data set.

Re-fitting Bayesian trend filtering to the above data sets when sampling from the full conditional of f every 2nd iteration had little effect on the location of the posterior distribution. We find that estimates of location don't change very much. By evaluating the mean absolute difference of the means and medians of the posterior's function evaluations at each input x_i, we can compare an overall location change of thinned sampling. Relative to the temperature data, the mean absolute difference of the mean of the posterior for function evaluations is 0.0028, and for the posterior medians, 0.0036. Further, the effective sample sizes for σ^2 changed little, from 3180 to 3058. On the other hand, the effective sample sizes for the penalty parameter λ dropped a bit more than one would hope, 244 down to 156. We calculated the same summary statistics for the sunspot data. The mean absolute difference of the means and medians of the posterior for function evaluations was 0.0009 and 0.0012, respectively. The effective sample size for σ^2 changes from 2862 to 2312, while it drops from 108 to 74 for λ.

Drawing samples of the full conditional for f only from the mth iteration of the Gibbs sampler proves to be an easy way to speed up Bayesian trend filtering. This idea relies on the fact that the full conditional for f quickly converges to the posterior distribution. In cases for which convergence is not as fast as in the examples provided here, or for which the other parameters are more highly correlated with the parameters f, such a strategy may not be as successful. Still, with few alternative methods available from the literature, this remains a fairly simple method to speed up the fit of Bayesian trend filtering.

4.3 Discussion

Sections 4.1 and 4.2 both offer encouraging results for Bayesian trend filtering. Bayesian trend filtering achieved the smallest mean and standard deviations of the mean squared errors of all the methods from the simulated data. This confirms the theoretical results proved by Tibshirani [2014], that with an appropriate choice of the penalty parameter trend filtering methods converge to the true underlying function at the minimax rate – a rate unmatched by methods linear in the observations $\mathbf{y}$. Bayesian trend filtering also appears to fit time series data well. We considered two real data sets, and Bayesian trend filtering performs as well as methods intentionally designed to handle the correlation across time. Compared to other methods which suppose uncorrelated errors, Bayesian trend filtering appears more robust to deviations from its underlying model assumptions.

Chapter 5

Future Work

We introduced a Bayesian version of the generalized lasso. The hierarchical model allows for a tractable Gibbs sampler to simulate draws from the posterior. Unfortunately, a matrix inversion is unavoidable. The matrix inversion, with $\mathcal{O}(n^3)$ computational complexity, is a limitation for this method. Some work is already ongoing to speed up the frequentist generalized lasso, but no easy solution eagerly awaits us under the Bayesian perspective. To solve this issue, some new sampling techniques come to mind; e.g. hybrid Monte Carlo or variational Bayes. But for much of the same reasons discussed in Section 3.6, these techniques are not intuitively good fits for this problem. An easy first step that is guaranteed to speed up the Bayesian generalized lasso, is to push off all computation to graphics processing units. Though this strategy will surely decrease run times, it does not overcome the inherent $\mathcal{O}(n^3)$ complexity of a matrix inversion.

Beyond speeding up the fit of the Bayesian generalized lasso, Bayesian trend filtering offers ample room for both computational and theoretical work. For instance, the hierarchical model underlying Bayesian trend filtering is composed of a Gaussian distribution on the observations and a Gaussian process prior on the underlying function of interest. This is the exact structure of Gaussian process regression methods. A connection between these methods already exists for the ℓ_2 norm, but previous work has already shown that estimators linear in the response vector $\mathbf{y}$ are not optimal. Mimicking this literature with the ℓ_1 norm, though the nondifferentiability of the ℓ_1 norm likely causes some serious issues, could open a world minimax convergent estimators. We already know Bayesian trend filtering, with its simple hierarchical model, is a minimax convergent estimator. Thus, the theoretical work to find optimal estimators would be nicely coupled with the tractable form of the Bayesian hierarhcical models.

Of course, Bayesian trend filtering is young and much testing and experimentation is necessary to better understand when and where this method best applies. For instance, we already know that Bayesian trend filtering is mostly restricted to smoothing. In the case of piecewise constant or linear function, Bayesian trend filtering may guide frequentist trend filtering's choice of penalty parameter. For instance, one could insist that the frequentist fit is completely contained within credible intervals, thereby guiding the choice of the penalty parameter. However, frequentist trend filtering's knot detection is a prized asset.

Beyond testing the performance of Bayesian trend filtering, there is also room to expand the model. For instance, it is easy enough to code up the elastic net penalty, or combinations of other penalty functions. A desirable example is the additive model, such that $\mathbb{E}y = \sum_{l=1}^{L} f_l(x)$. In this case, each function f_l could be penalized with a different, user specified derivative k_l. Further, the Bayesian perspective of trend filtering is better suited to modify its assumptions than is the frequentist version. One could easily allow for t-distributed or correlated errors, to increase Bayesian trend filtering's applicability.

Appendix A

Appendix

A.1 Full Conditionals

We provide justifications of the full conditionals presented in Section 3.6. The full conditional for f involves little more than completing the square with matrices. Start with the likelihood and the priors that involve only terms relating to f directly.

$$\begin{aligned}[\beta|\cdot] &\propto \mathcal{N}_n(\mathbf{y}|X\beta, \sigma^2 I_n) \cdot \mathcal{N}_p(\beta|0, \sigma^2\Sigma_\beta) \\ &\propto \exp\left\{\frac{-1}{2\sigma^2}[\beta^t(\mathbf{X}^t\mathbf{X} + \Sigma_\beta^{-1})\beta - 2\mathbf{y}^t\mathbf{X}\beta]\right\}\end{aligned}$$

Completing the square gives

$$[\beta|\cdot] \propto \exp\left\{\frac{-1}{2\sigma^2}[(\beta - (\mathbf{X}^t\mathbf{X} + \Sigma_\beta^{-1})^{-1}\mathbf{y})^t(\mathbf{X}^t\mathbf{X} + \Sigma_\beta^{-1})^{-1}(\beta - (\mathbf{X}^t\mathbf{X} + \Sigma_\beta^{-1})^{-1}\mathbf{y})]\right\}.$$

The full conditional for σ^2 is amongst the easiest to find

$$\begin{aligned}[\sigma^2|\cdot] &\propto \mathcal{N}_n(\mathbf{y}|\mathbf{X}\beta, \sigma^2 I_n) \cdot \mathcal{N}_p(\beta|0, \sigma^2\Sigma_\beta)) \cdot \sigma^{-2} \\ &\propto (\sigma^2)^{n-1+p} \exp\left\{\frac{-1}{2\sigma^2}[(\mathbf{y} - \mathbf{X}\beta)^t(y - \mathbf{X}\beta) + \beta^t\Sigma_\beta^{-1}\beta]\right\}.\end{aligned}$$

The full conditional for ω_j^{-2} takes quite a bit of manipulation. Because of this, we work our the full conditional for Bayesian trend filtering, where we have a specific case of the penalty matrix D to work with. Insight from previous papers allows us to see the connection to the inverse Gaussian distribution Park and Casella [2008]; Kyung *et al.* [2010]. For cleaner notation, define $\eta_j := \omega_j^{-2}$ and $\gamma(\cdot, f)$. Then we can write the penalty term as a double sum,

$$||D^{(x,k+1)}f||_1 = \sum_{i=1}^{n-k-1}\left|\sum_{j=i}^{i+k+1}(-1)^{j-i}\binom{k+1}{j-i}f(x_j)\right| = \sum_{i=1}^{n-k-1}|\gamma(i,f)|,$$

where dependence on k is implicit. After making the appropriate transformation of ω_j^{-2}, we find

$$\begin{aligned}[\eta_j|\cdot] &\propto \mathcal{N}_n(f|0, \sigma^2\Sigma_f(\eta_j^{-1})) \cdot [\eta_j]|\eta_j^{-2}| \\ &\propto |\Sigma_f(\eta_j^{-1})|^{-1/2} \exp\left\{\frac{-1}{2\sigma^2}\sum_{l=1}^{n-k-1}\eta_l\gamma^2(l,f)\right\}\exp\left\{\frac{-\lambda^2}{2\eta_j}\right\}\eta_j^{-2}.\end{aligned}$$

We are only concerned with $\eta_l = \eta_j$ and can thus reduce notation further; put $\gamma := \gamma(j, f)$. Collect η_js and take out of the proportionality constant $\exp\{\lambda\gamma/\sigma\}$ to get

$$\begin{aligned}[\eta_j|\cdot] &\propto \eta_j^{-3/2}\exp\left\{\frac{-\eta_j\gamma^2}{2\sigma^2} - \frac{\lambda^2}{2\eta_j}\right\} \\ &\propto \eta_j^{-3/2}\exp\left\{\frac{-\eta_j\gamma^2}{2\sigma^2} + \frac{\lambda\gamma}{\sigma} - \frac{\lambda^2}{2\eta_j}\right\}.\end{aligned}$$

After multiplying the first two terms in the exponential by λ^2/λ^2, and λ/λ, respectively, and rearranging, put $\mu := \lambda\sigma/\gamma$ and $\zeta := \lambda$ to highlight the kernel of the inverse Gaussian distribution.

$$
\begin{aligned}
[\eta_j|\cdot] &\propto \eta_j^{-3/2} \exp\left\{ \frac{-\lambda^2\gamma^2}{2\sigma^2\eta_j\lambda^2} \left(\eta_j^2 - \frac{2\lambda\sigma\eta_j}{\gamma} + \left(\frac{\lambda\sigma}{\gamma}\right)^2 \right) \right\} \\
&= \eta_j^{-3/2} \exp\left\{ \frac{-\zeta^2}{2\mu^2\eta_j} (\eta_j - \mu)^2 \right\}.
\end{aligned}
$$

The full conditional for λ^2 is quite easy to find

$$
\begin{aligned}
[\lambda^2|\cdot] &\propto \Gamma(\alpha, \rho) \cdot [\omega_1^2, \ldots, \omega_m^2|\lambda^2] \\
&\propto (\lambda^2)^{m+\alpha-1} \exp\left\{ -\lambda^2(\rho + \sum_{j=1}^{m} \omega_j^2/2) \right\}.
\end{aligned}
$$

A.2 Code

Below is the `C++` class developed as the base of all the `R` functions to fit Bayesian trend filtering. This code uses the linear algebra template library `Eigen` [Guennebaud *et al.*, 2010].

```
#include <RcppEigen.h>
#include <vector>
#include <random>

// [[Rcpp::depends(RcppEigen)]]

typedef Eigen::VectorXd Vec;
typedef Eigen::Map<Vec> MVec;
typedef Eigen::MappedSparseMatrix<double> MspMat;
typedef Eigen::SparseMatrix<double> spMat;

class individual {
private:
  /* fields */
  MVec y;
  int max_draws;
  MspMat D;
  spMat sigma;
  spMat I;
  Eigen::SimplicialLLT<spMat > LLt;

  /* random */
  Vec rndNorm(const int& n_) {
    Rcpp::RNGScope scope;
    Rcpp::NumericVector x = Rcpp::rnorm(n_);
    Vec out(Rcpp::as<Vec>(x));
    return out;
  }
  double rInvGauss(const double& nu_, const double& lambda_) {
    Vec z = rndNorm(1);          // one N(0,1)
    double z2 = z(0)*z(0);
    double nu2 = nu_*nu_;
    double c = 0.5*nu_/lambda_;
    double x = nu_ + c*z2*nu_ - c*std::sqrt(4.0*nu_*lambda_*z2 + nu2*z2*z2);
    Vec u = rndUniform(1);
    double out = (u(0) < nu_/(nu_+x)) ? x : nu2/x;
    return out;
```

```
}
template <typename T>
Vec rndInvGauss(const Eigen::DenseBase<T>& nu_, const double& lambda_) {
  Vec out(nk); int draws;
  for (int i=0; i<nk; i++) {
    draws = 0;
    do {
      out(i) = rInvGauss(nu_(i), lambda_);
      ++draws;
    } while ( out(i) < 1e-11 && draws < max_draws );
    if ( draws > 1 ) Rcpp::Rcout << "Note: InvGauss resampled." << std::↩
        endl;
  }
  return out;
}
// rndMVNorm templated?
Vec rndMVNorm(const Vec& mu_, const spMat& sqrtCov_, const double& scale) ↩
    {
  return (scale*(sqrtCov_*rndNorm(sqrtCov_.cols()))+mu_).eval();
}
Vec rndGamma(const int& n_, const double& shape_, const double& scale_) {
  Rcpp::RNGScope scope; Rcpp::NumericVector x;
  int draws = 0; bool toosmall;
  do {                          // cut off tail
    x = Rcpp::rgamma(n_, shape_, scale_);
    toosmall = Rcpp::is_true( Rcpp::any( x < 1e-11) );
    ++draws;
  } while ( toosmall && draws < max_draws );
  if ( draws > 1 ) Rcpp::Rcout << "Note: Gamma resampled." << std::endl;
  Vec out(Rcpp::as<Vec>(x));        // convert to Eigen::VectorXd
  return out;
}
Vec rndUniform(const int& n_) {
  Rcpp::RNGScope scope;
  Rcpp::NumericVector x = Rcpp::runif(n_);
  Vec out(Rcpp::as<Vec>(x));
  return out;
}

/* initializers */
void init_l2() {
  Vec L = rndGamma(1, nk+alpha, 2.0/(1.0+rho));
  l2 = L(0);
}
void init_l() {
  Vec L = rndGamma(1, nk+alpha, 2.0/(1.0+rho));
  l = L(0);
}
void init_o2() {
  Vec O = rndGamma(nk, 1.0, 2.0);
  sigma = D.transpose()*mkDiag(O.cwiseInverse())*D;
  o2 = O;
}
void init_s2() {
  Vec S = rndGamma(1, 1.0, 0.5);
  s2 = 1.0/S(0);
}
void init_beta() {
  beta = y;
```

```
  }

  /* utility */
  // spMat mkDiag(const int& sz) {
  //   spMat W(sz, sz); W.reserve(sz);
  //   for (int i=0; i<sz; i++) {
  //     W.insert(i,i) = 1.0;
  //   }
  //   return W;
  // }
  template <typename T>
  spMat mkDiag(const Eigen::DenseBase<T>& val) {
    int I = val.size();
    spMat W(I,I); W.reserve(I);
    for (int i=0; i<I; i++) {
      W.insert(i,i) = val(i);     // insert along diagonal
    }
    return W;
  }

public:

  /* fields */
  int n, nk;
  Vec beta, o2;
  double l, l2, s2, alpha, rho, Dbl1;
  Vec Db;

  /* constructor */
  individual(const MVec y_, const  MspMat D_, const double alpha_, const ↩
      double rho_) : y(y_), D(D_), alpha(alpha_), rho(rho_) {

    // general info
    max_draws = 10;
    n = y.size();
    nk = D.rows();

    // initialize
    init_o2();
    init_s2();
    init_beta();
    init_l2();
    init_l();

    Db = D*beta;                // initialize D*beta;
    LLt.analyzePattern(sigma); // symbolic decomposition on the sparsity
    LLt.setShift(1.0, 1.0);     // add I to Sigma_f
  }

  /* update parameters */
  void upBeta() {
    LLt.factorize(sigma);
    spMat L(n,n); spMat Ltinv(n,n);
    LLt.matrixL().twistedBy(LLt.permutationPinv()).evalTo(L);
    Ltinv = LLt.solve(L);
    int draws = 0;
    do {
      beta = rndMVNorm(Ltinv*(Ltinv.transpose()*y), Ltinv, std::sqrt(s2));
      Db = D*beta;
```

```
      Dbl1 = (Db).lpNorm<1>();
      ++draws;
    } while ( (Db.cwiseAbs().array() < 1e-10).any() && draws < max_draws );
    if (draws > 1) Rcpp::Rcout << "Note: Normal resampled." << std::endl;
  }
  void upS2() {
    double rate = (y-beta).squaredNorm() + beta.transpose()*sigma*beta;
    Vec S = rndGamma(1, n, 2.0/rate);
    s2 = 1.0/S(0);
  }

  // dexp
  void upOmega2() {
    Vec eta = rndInvGauss(Db.cwiseAbs().cwiseInverse()*std::sqrt(l2*s2), l2)↩
        ;
    if ( (o2.array() <= 0.).any() ) {
      Rcpp::Rcout << "Warning: At least one omega <= zero..." << std::endl;
      eta = eta.cwiseAbs();
    }
    o2 = eta.cwiseInverse();

    // update sigma_f
    sigma = D.transpose()*mkDiag(eta)*D;
    sigma.makeCompressed();
  }
  void upLambda2() {
    double tmp = o2.sum();
    // Rcpp::Rcout << "sum of omegas = " << tmp << std::endl;
    Vec lambda2 = rndGamma(1, nk+alpha, 2.0/(tmp+2*rho));
    l2 = lambda2(0);
  }

  // gdP
  void upOmega() {
    Vec eta = rndInvGauss(Db.cwiseAbs().cwiseInverse()*(l*std::sqrt(s2)), l*↩
        l);
    o2 = eta.cwiseInverse();
    if ( (o2.array() < 0.).any() ) {
      Rcpp::Rcout << "Warning: At least one omega <= zero..." << std::endl;
      eta = eta.cwiseAbs();
    }

    // update sigma_f
    sigma = D.transpose()*mkDiag(eta)*D;
    sigma.makeCompressed();
  }
  void upLambda() {
    double sig = std::sqrt(s2);
    Vec lambda = rndGamma(1, nk+alpha, sig/((D*beta).lpNorm<1>() + rho*sig))↩
        ;
    l = lambda(0);
  }
};
```

The above class is then called by the following functions. The first one uses the generalized double Pareto conditional prior distribution.

```
#include "individual.cpp"
```

```
// [[Rcpp::export]]
Rcpp::List gdp(const int& iter,
                       const Eigen::Map<Eigen::VectorXd>& y,
                       const Eigen::MappedSparseMatrix<double>& D,
                       const double& alpha, const double& rho,
                       const int& m, const bool& debug) {

  // initialize btf object
  individual *btf;
  btf = new individual(y, D, alpha, rho);
  bool broken = false;

  // initialize matrices of posterior draws
  Eigen::MatrixXd beta_draws = Eigen::MatrixXd::Zero(iter, btf->n);
  Eigen::MatrixXd omega_draws = Eigen::MatrixXd::Zero(iter, btf->nk);
  Eigen::VectorXd s2_draws = Eigen::VectorXd::Zero(iter);
  Eigen::VectorXd lambda_draws = Eigen::VectorXd::Zero(iter);

  // run sampler
  for (int i=0; i<iter; ++i) {
    if (i % m == 0) {
      btf->upBeta();
      beta_draws.row(i) = btf->beta.transpose();
    }
    btf->upS2();
    s2_draws(i) = btf->s2;
    btf->upLambda();
    lambda_draws(i) = btf->l;
    btf->upOmega();
    omega_draws.row(i) = btf->o2.transpose();

    if ( debug ) {
      for (int j=0; j<btf->n; ++j) {
        if (std::isnan(beta_draws(i,j))) {
          Rcpp::Rcout << "Warning: watch out gdp!, nan @ beta("
                      <<  i << "," << j << ")" << std::endl;
          broken = true;
        }
      }
      for (int j=0; j<btf->nk; ++j) {
        if (std::isnan(omega_draws(i,j))) {
          Rcpp::Rcout << "Warning: watch out gdp!, nan @ omega("
                      <<  i << "," << j << ")" << std::endl;
          broken = true;
        }
      }
      if (std::isnan(s2_draws(i)) || std::isnan(lambda_draws(i))) {
          Rcpp::Rcout << "Warning: watch out gdp!, nan @ s2|lambda("
                      <<  i << ")" << std::endl;
          broken = true;
      }
    }
    if (broken) break;
  }
  return Rcpp::List::create(Rcpp::Named("beta") = Rcpp::wrap(beta_draws),
                            Rcpp::Named("s2") = s2_draws,
                            Rcpp::Named("lambda") = lambda_draws,
                            Rcpp::Named("omega") = omega_draws);
}
```

The second function fits the double exponential conditional prior.

```
#include "individual.cpp"

// [[Rcpp::export]]
Rcpp::List dexp(const int& iter,
                    const Eigen::Map<Eigen::VectorXd>& y,
                    const Eigen::MappedSparseMatrix<double>& D,
                    const double& alpha, const double& rho,
                    const int& m, const bool& debug) {

  // initialize btf object
  individual *btf;
  btf = new individual(y, D, alpha, rho);
  bool broken = false;

  // initialize matrix of posterior draws
  Eigen::MatrixXd beta_draws = Eigen::MatrixXd::Zero(iter, btf->n);
  Eigen::MatrixXd omega_draws = Eigen::MatrixXd::Zero(iter, btf->nk);
  Eigen::VectorXd s2_draws = Eigen::VectorXd::Zero(iter);
  Eigen::VectorXd lambda_draws = Eigen::VectorXd::Zero(iter);

  // runs sampler
  for (int i=0; i<iter; ++i) {
    if (i % m == 0) {
      btf->upBeta();
      beta_draws.row(i) = btf->beta.transpose();
    }
    btf->upS2();
    s2_draws(i) = btf->s2;
    btf->upLambda2();
    lambda_draws(i) = btf->l;
    btf->upOmega2();
    omega_draws.row(i) = btf->o2.transpose();

    if ( debug ) {
      for (int j=0; j<btf->n; ++j) {
        if (std::isnan(beta_draws(i,j))) {
          Rcpp::Rcout << "Warning: watch out dexp!, nan @ beta("
                      <<  i << "," << j << ")" << std::endl;
          broken = true;
        }
      }
      for (int j=0; j<btf->nk; ++j) {
        if (std::isnan(omega_draws(i,j))) {
          Rcpp::Rcout << "Warning: watch out dexp!, nan @ omega("
                      <<  i << "," << j << ")" << std::endl;
          broken = true;
        }
      }
      if (std::isnan(s2_draws(i)) || std::isnan(lambda_draws(i))) {
          Rcpp::Rcout << "Warning: watch out dexp!, nan @ s2|lambda("
                      <<  i << ")" << std::endl;
          broken = true;
      }
    }
    if (broken) break;
  }
```

```
  return Rcpp::List::create(Rcpp::Named("beta") = Rcpp::wrap(beta_draws),
                            Rcpp::Named("s2") = s2_draws,
                            Rcpp::Named("lambda") = lambda_draws,
                            Rcpp::Named("omega") = omega_draws);
}
```

The last significant piece of code that is the approximation to a Bayesian trend filtering fit given a user supplied value for the penalty parameter λ. In fact, a user supplied vector of penalty parameters is allowed, in case solution paths are of interest. Sparse matrices are used wherever possible.

```
#include <RcppEigen.h>

using namespace Rcpp;

template <typename T>
Eigen::SparseMatrix<double> mkDiag(const Eigen::DenseBase<T>& val) {
  int I = val.size();
  Eigen::SparseMatrix<double> W(I,I); W.reserve(I);
  for (int i=0; i<I; i++) {
    W.insert(i,i) = val(i);     // insert along diagonal
  }
  return W;
}

// [[Rcpp::export]]
List tf_approx(const Eigen::Map<Eigen::VectorXd>& y,
               const Eigen::Map<Eigen::VectorXd>& l,
               const Eigen::MappedSparseMatrix<double>& D,
               const int& k, const double& eps,
               const double& tau, const int& max_iter) {
  // author Edward A. Roualdes
  // initialize some values
  const int n = y.size();        // sample size
  const int nk = n-k-1;
  const int J = l.size();         // number of lambdas
  Eigen::VectorXd beta = y;                  // initial values
  Eigen::VectorXd beta_old(n);

  // create output containers
  Eigen::MatrixXd beta_out(n, J);         // beta out matrix
  Eigen::SparseMatrix<double> W(nk, nk);
  W = mkDiag(1.0/((D*beta).cwiseAbs().array() + eps));
  Eigen::SimplicialLLT<Eigen::SparseMatrix<double> > LLt; // linear system
  LLt.analyzePattern(D.transpose()*W*D); // symbolic decomposition on the ↩
      sparcity
  LLt.setShift(1.0, 1.0);                 // add Identity
  Eigen::VectorXd iter_out(J);              // store number iterations per ↩
      lambda

  // for each lambda
  for (int j=0; j<J; j++) {
    int iter = 0;                // count iterations until convergence

    // while solution not close enough && under max iterations
    while (iter < max_iter) {
      beta_old = beta;           // store old estimates
      W = mkDiag(1.0/((D*beta).cwiseAbs().array() + eps));
```

```
      // create linear system and solve
      LLt.factorize(l(j)*D.transpose()*W*D); // computational decomp on ↩
          symbolic
      beta = LLt.solve(y);

      // check convergence
      if (beta.isApprox(beta_old, tau)) break;
      ++iter;
    }
    // store estimates for each lambda
    beta_out.col(j) = beta;
    iter_out(j) = iter;
  }

  return List::create(Named("coefficients") = beta_out,
                      Named("iters") = iter_out);
}
```

Appendix B

References

David M Allen. *The prediction sum of squares as a criterion for selecting predictor variables.* University of Kentucky, 1971.

David M Allen. The relationship between variable selection and data agumentation and a method for prediction. *Technometrics*, 16(1):125–127, 1974.

David F Andrews and Colin L Mallows. Scale mixtures of normal distributions. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 99–102, 1974.

Artin Armagan, David B Dunson, and Jaeyong Lee. Generalized double Pareto shrinkage. *Statistica Sinica*, 23(1):119, 2013.

Taylor B. Arnold and Ryan Joseph Tibshirani. *genlasso: Path algorithm for generalized lasso problems*, 2014. R package version 1.3.

Douglas Bates and Dirk Eddelbuettel. Fast and elegant numerical linear algebra using the RcppEigen package. *Journal of Statistical Software*, 52(5):1–24, 2013.

Bernd Bischl, Michel Lang, and Olaf Mersmann. *BatchExperiments: Statistical experiments on batch computing clusters.*, 2014. R package version 1.3.

Patrick Breheny and Jian Huang. Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. *The annals of applied statistics*, 5(1):232, 2011.

Leo Breiman and Philip Spector. Submodel selection and evaluation in regression – the x-random case. *International statistical review/revue internationale de Statistique*, pages 291–319, 1992.

Carlos M Carvalho, Nicholas G Polson, and James G Scott. Handling sparsity via the horseshoe. In *International Conference on Artificial Intelligence and Statistics*, pages 73–80, 2009.

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

SILSO World Data Center. The international sunspot number. *International Sunspot Number Monthly Bulletin and online catalogue*, 2014.

Hugh A Chipman, Edward I George, Robert E McCulloch, et al. Bart: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010.

R Core Team. *R: A language and environment for statistical computing.* R Foundation for Statistical Computing, Vienna, Austria, 2014.

Anthony Christopher Davison. *Bootstrap methods and their application*, volume 1. Cambridge university press, 1997.

Carl de Boor. *A practical guide to splines.* Number v. 27 in Applied Mathematical Sciences. Springer, 2001.

Carl de Boor. Divided differences. *Survey Approximation Theory*, 1:46–69, 2005.

Ronald A DeVore and George G Lorentz. *Constructive approximation*, volume 303. Springer, 1993.

David L Donoho and Iain M Johnstone. Adapting to unknown smoothness via wavelet shrinkage. *Journal of the american statistical association*, 90(432):1200–1224, 1995.

Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222, 1987.

Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

Jianqing Fan and Runze Li. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association*, 96(456):1348–1360, 2001.

Centers for Disease Control and Prevention. United States cancer statistics. 2015.

Jerome Friedman, Trevor Hastie, Holger Höfling, and Robert Tibshirani. Pathwise coordinate optimization. *The Annals of Applied Statistics*, 1(2):302–332, 2007.

Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

Alan E Gelfand and Adrian FM Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American statistical association*, 85(410):398–409, 1990.

Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. *Bayesian Data Analysis*, volume 2. Taylor & Francis, 2014.

Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):721–741, 1984.

Gene H Golub, Michael Heath, and Grace Wahba. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223, 1979.

P.J. Green and B.W. Silverman. *Nonparametric Regression and Generalized Linear Models: A roughness penalty approach*. Chapman & Hall/CRC Monographs on Statistics & Applied Probability. Taylor & Francis, 1993.

Jim E Griffin and Philip J Brown. Bayesian hyper-lassos with non-convex penalization. *Australian & New Zealand Journal of Statistics*, 53(4):423–442, 2011.

Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010.

Trevor Hastie and Robert Tibshirani. Varying-coefficient models. *Journal of the Royal Statistical Society. Series B (Methodological)*, 55(4):pp. 757–796, 1993.

Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: data mining, inference and prediction. *The Mathematical Intelligencer*, 27(2):83–85, 2005.

James S Hodges. *Richly Parameterized Linear Models: Additive, Time Series, and Spatial Models Using Random Effects*. CRC Press, 2013.

Arthur E Hoerl and Robert W Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67, 1970.

David R Hunter and Runze Li. Variable selection using MM algorithms. *Annals of statistics*, 33(4):1617, 2005.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

Adam Kapelner and Justin Bleich. Bartmachine: A powerful tool for machine learning. *arXiv preprint arXiv:1312.2171*, 2013.

Adam Kapelner and Justin Bleich. bartMachine: Machine learning with Bayesian additive regression trees. *ArXiv e-prints*, 2014.

Jafar A Khan, Stefan Van Aelst, and Ruben H Zamar. Robust linear model selection based on least angle regression. *Journal of the American Statistical Association*, 102(480):1289–1299, 2007.

Keith Knight and Wenjiang Fu. Asymptotics for lasso-type estimators. *Annals of Statistics*, pages 1356–1378, 2000.

Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145, 1995.

Anita J. van der Kooij. *Prediction accuracy and stability of regression with optimal scaling transformations*. PhD thesis, Leiden University, 2007.

Minjung Kyung, Jeff Gill, Malay Ghosh, and George Casella. Penalized regression, standard errors, and Bayesian lassos. *Bayesian Analysis*, 5(2):369–411, 2010.

A. Lee, F. Caron, A. Doucet, and C. Holmes. A hierarchical Bayesian framework for constructing sparsity-inducing priors. *ArXiv e-prints*, September 2010.

Anthony Lee, Francois Caron, Arnaud Doucet, Chris Holmes, et al. Bayesian sparsity-path-analysis of genetic association signal using generalized t priors. *Statistical applications in genetics and molecular biology*, 11(2):1–29, 2012.

Jun Liu, Lei Yuan, and Jieping Ye. An efficient algorithm for a class of fused lasso problems. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 323–332. ACM, 2010.

Zhi-Quan Luo and Paul Tseng. On the convergence of the coordinate descent method for convex differentiable minimization. *Journal of Optimization Theory and Applications*, 72(1):7–35, 1992.

Enno Mammen and Sara van de Geer. Locally adaptive regression splines. *The Annals of Statistics*, 25(1):387–413, 1997.

Enno Mammen. Nonparametric regression under qualitative smoothness assumptions. *The Annals of Statistics*, pages 741–759, 1991.

Radford M Neal and Geoffrey E Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

John A Nelder and RJ Baker. Generalized linear models. *Encyclopedia of Statistical Sciences*, 1972.

Michael R Osborne, Brett Presnell, and Berwin A Turlach. On the lasso and its dual. *Journal of Computational and Graphical statistics*, 9(2):319–337, 2000.

Trevor Park and George Casella. The Bayesian lasso. *Journal of the American Statistical Association*, 103(482):681–686, 2008.

Martyn Plummer, Nicky Best, Kate Cowles, and Karen Vines. Coda: Convergence diagnosis and output analysis for MCMC. *R News*, 6(1):7–11, 2006.

Benedikt M Pötscher and Hannes Leeb. On the distribution of penalized maximum likelihood estimators: The lasso, scad, and thresholding. *Journal of Multivariate Analysis*, 100(9):2065–2082, 2009.

Carl Edward Rasmussen. Gaussian processes for machine learning. 2006.

Christian Ritter and Martin A Tanner. Facilitating the Gibbs sampler: the Gibbs stopper and the griddy-Gibbs sampler. *Journal of the American Statistical Association*, 87(419):861–868, 1992.

Saharon Rosset and Ji Zhu. Piecewise linear regularized solution paths. *The Annals of Statistics*, pages 1012–1030, 2007.

Edward A. Roualdes. *btf: Estimates univariate function via Bayesian trend filtering*, 2014. R package version 1.1.

Henry Scheffe. *The Analysis of Variance*, volume 72. John Wiley & Sons, 1960.

Robert Tibshirani, Michael Saunders, Saharon Rosset, Ji Zhu, and Keith Knight. Sparsity and smoothness via the fused lasso. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(1):91–108, 2005.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

Ryan Joseph Tibshirani. *The solution path of the generalized lasso*. Stanford University, 2011.

Ryan Joseph Tibshirani. Adaptive piecewise polynomial estimation via trend filtering. *The Annals of Statistics*, 42(1):285–323, 2014.

Andrey Nikolayevich Tikhonov. On the stability of inverse problems. In *Dokl. Akad. Nauk SSSR*, volume 39, pages 195–198, 1943.

Sara Van de Geer. Estimating a regression function. *The Annals of Statistics*, pages 907–924, 1990.

Aki Vehtari and Jarno Vanhatalo. Sparse log Gaussian processes via MCMC for spatial epidemiology. *Wshop on GP in Practice*, 2007.

Grace Wahba. *Spline Models for observational Data*, volume 59. Siam, 1990.

Bob Wheeler. *SuppDists: Supplementary distributions*, 2013. R package version 1.1-9.1.

Hadley Wickham and Romain Francois. *dplyr: A Grammar of Data Manipulation*, 2015. R package version 0.4.1.

Hadley Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009.

John M Winn and Christopher M Bishop. Variational message passing. In *Journal of Machine Learning Research*, pages 661–694, 2005.

John Michael Winn. *Variational message passing and its applications*. PhD thesis, University of Cambridge, 2004.

S.N Wood. *Generalized Additive Models: An Introduction with R*. Chapman and Hall/CRC, 2006.

Tong Tong Wu and Kenneth Lange. Coordinate descent algorithms for lasso penalized regression. *The Annals of Applied Statistics*, pages 224–244, 2008.

Matt Wytock, Suvrit Sra, and J Zico Kolter. Fast Newton methods for the group fused lasso. In *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, 2014.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

Cun-Hui Zhang. Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, 38(2):894–942, 2010.

Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.

Appendix C

Vita

- Edward A. Roualdes, San Rafael, California
- Education
  - MS Economics, Tufts University 2009-2010
  - BS Mathematics, California State University (CSU), Chico 2003-2008
    BA Economics *with honors*
    Minor Business Administration
- Professional Positions
  - Statistical Consultant for the College of Agriculture 2013-2015
    University of Kentucky
  - Research Assistant for The Weisrock Lab 2012-2014
    Department of Biology, University of Kentucky
  - Primary Instructor 2011-2012
    Department of Statistics, University of Kentucky
  - Teacher's Assistant 2010-2011
    Department of Statistics, University of Kentucky
  - Teacher's Assistant 2009-2010
    Economics Department, Tufts University
- Scholastic Honors
  - R.L. Anderson Research Award, University of Kentucky, Department of Statistics 2014
  - Lt. Robert Merton Rawlins Merit Award, CSU, Chico 2007
- Publications
  - Fenger, C. K., Tobin, T., Casey, P. J., Roualdes, E. A., Langemeier, J., Haines, D. M. *Bovine colostrum supplementation optimises earnings, performance and recovery in racing Thoroughbreds.* Comparative Exercise Physiology 11/2014; 10(4):233-238. DOI: 10.3920/CEP140023
  - Roualdes, E. A. `btf`: Estimates univariate function via Bayesian trend filtering. R package version 1.0. `http://cran.r-project.org/web/packages/btf/index.html`
  - Roualdes, E. A., Bonner, S. `spiders`: fits our predator preferences model. R package version 1.0. `https://github.com/roualdes/spiders`
  - Roualdes, E. A. `NextAllele`: a bioinformatic program written in Python to phase haplotypes of diploid animals. `https://code.google.com/p/nextallele/`