Three Essays on Shrinkage Estimation and Model Selection of Linear and Nonlinear Time Series Models

by

Mario Giacomazzo

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved May 2018 by the
Graduate Supervisory Committee:

Yiannis Kamarianakis, Chair
Mark Reiser
Rob McCulloch
Richard Hahn
John Fricks

ARIZONA STATE UNIVERSITY

May 2018

ABSTRACT

The primary objective in time series analysis is forecasting. Raw data often exhibits nonstationary behavior: trends, seasonal cycles, and heteroskedasticity. After data is transformed to a weakly stationary process, autoregressive moving average (ARMA) models may capture the remaining temporal dynamics to improve forecasting. Estimation of ARMA can be performed through regressing current values on previous realizations and proxy innovations. The classic paradigm fails when dynamics are nonlinear; in this case, parametric, regime-switching specifications model changes in level, ARMA dynamics, and volatility, using a finite number of latent states. If the states can be identified using past endogenous or exogenous information, a threshold autoregressive (TAR) or logistic smooth transition autoregressive (LSTAR) model may simplify complex nonlinear associations to conditional weakly stationary processes. For ARMA, TAR, and STAR, order parameters quantify the extent past information is associated with the future. Unfortunately, even if model orders are known *a priori*, the possibility of over-fitting can lead to sub-optimal forecasting performance. By intentionally overestimating these orders, a linear representation of the full model is exploited and Bayesian regularization can be used to achieve sparsity. Global-local shrinkage priors for AR, MA, and exogenous coefficients are adopted to pull posterior means toward 0 without over-shrinking relevant effects. This dissertation introduces, evaluates, and compares Bayesian techniques that automatically perform model selection and coefficient estimation of ARMA, TAR, and STAR models. Multiple Monte Carlo experiments illustrate the accuracy of these methods in finding the ”true” data generating process. Practical applications demonstrate their efficacy in forecasting.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

Table Page

Table Page

LIST OF FIGURES

Chapter 1

INTRODUCTION

In order of historical discovery, the autoregressive moving average, threshold autoregressive, and logistic smooth transition autoregressive processes have been extensively studied for the modeling and forecasting of linear and nonlinear time series. All three models are parametric and easy to interpret making them popular in application. Order parameters control the overall complexity of these models and often require estimation. By intentionally overestimating these order parameters, Bayesian regularization and selection methods can be applied for flexible subset estimation of these parametric models.

The logistic smooth transition autoregressive (LSTAR) model is a regime-switching nonlinear time series specification that has been adopted in a wide variety of applications. LSTAR is formulated as a weighted combination of two or more linear autoregressive (AR) processes. In Chapter 2, LSTAR models are estimated using Bayesian shrinkage (*Laplace* and $Horseshoe$) priors on the autoregressive coefficients of each regime and *Dirichlet* priors are employed to identify composite threshold variables in the transition function. The proposed specification provides a flexible alternative to time-consuming stepwise model building procedures and to computationally intensive reversible jump Markov chain Monte Carlo (RJMCMC) schemes. A series of experiments is presented to demonstrate the efficacy of the methodology, which can be applied in existing Bayesian software packages. Application to a classic nonlinear time series illustrates the ability to achieve superior forecasting performance. Finally, the capability to handle multiple input exogenous time series is exemplified through forecasting daily maximum water temperatures: for 31 Spanish rivers, Bayesian estimates of linear and nonlinear river-specific models are evaluated with regard to their 3-step and 7-step ahead forecasting performance.

Urban traffic patterns naturally change with the growing populations of metropolitan areas. Real-time management systems capture high frequency traffic data to obtain short-term forecasts of critical traffic variables. For example, traffic occupancy measures vehicular density in an arterial through the percentage of time a sensor detects a vehicle. Major research over the last 20 years focused primarily on the modeling and forecasting of traffic volume. Like traffic volume, occupancy is a useful metric for quantifying traffic concentration that exhibits weekly seasonal patterns, nonlinear dynamics, and heteroskedasticity. Multiple-regime threshold autoregressive models (TAR), reformulated as high dimensional linear regressions, help understand the changing temporal dynamics as traffic flows between different levels of congestion. In Chapter 3, a Bayesian three step model building procedure is used for parsimonious estimation of subset TAR models designed for day-specific and horizon-specific (1-step, 3-step, and 5-step ahead) forecasting of traffic occupancy at 7 detector locations. In the first step, fully saturated multiple regime TAR models are fitted using Bayesian horseshoe priors for sparse estimation. Next, regimes are selected through a forward stepwise selection algorithm based on the Kullback-Leibler (KL) distance between the posterior predictive distribution of the full reference model and a TAR model with fewer regimes. Given the regimes, the forward selection algorithm is repeated to ensure the most parsimonious model is selected. Empirical results applied to traffic data from Athens, Greece, establish the efficacy of these procedures in obtaining interpretable models designed to produce point and density forecasts at multiple horizons.

The autoregressive moving average (ARMA) model is valuable in describing and forecasting weakly stationary stochastic processes. Classic ARMA model selection relies on choosing AR order p and MA order q to minimize prediction error (PE). Information criteria such as AIC or BIC discourage overfitting to estimate PE. The subset ARMA(p,q) model is more flexible but often involves computationally intensive methods for model selection. Treating ARMA as a linear regression model, regularization techniques are explored and evaluated to automatically select and estimate subset ARMA(p,q) in Chapter 4. Because of temporal dependence, procedures considered are capabable of handling the natural multicollinearity existant in AR and MA predictors. Extended from the adaptive LASSO (ADLASSO) used in Chen and Chan (2011), the adaptive elastic net (ADENET) which combines ℓ_1 and ℓ_2 regularization is considered. Beyond AIC and BIC, cross-validation techniques estimate PE and aid in final model selection. Under the Bayesian framework, horseshoe (HS) priors are valuable in sparse estimation of a full ARMA(p,q) reference model. Posterior distributions of sub models are quickly obtainable through projection, and discrepancy is measured by the Kullback-Leibler distance. A forward selection algorithm identifies the best nested sequence of subset ARMA(p,q) models, and the final model is chosen based on estimated PE. For the full library of methods discussed, model selection is evaluated via simulation and forecasting performance via practical application.

Chapter 2

BAYESIAN SHRINKAGE ESTIMATION OF LOGISTIC SMOOTH TRANSITION AUTOREGRESSIONS

2.1 Introduction

Occasionally, classical linear time series models inadequately capture temporal dynamics in the expected levels of a process, resulting in suboptimal forecasting performance (Lee *et al.*, 1993). The presence of significant nonlinear associations leads to the daunting task of selecting an adequate model from an expanding library of nonlinear specifications. A parametric subset from the aforementioned library extends autoregressive processes to account for changes in regimes or states (Priestley, 1988); nonlinear phenomena in financial and economic time-series motivated the majority of research in this area (Teräsvirta *et al.*, 2010; Zivot and Wang, 2006; Franses and Van Dijk, 2000). For example, asymmetries in quarterly national industrial production indexes can be explained by differentiating dynamics in two regimes: recessions and expansions (Terasvirta and Anderson, 1992). Although popularized by econometricians, regime-switching nonlinear time series models have been applied in a variety of research problems related for instance to the dynamics of network flows (Kamarianakis *et al.*, 2010) and the dynamics of air (Battaglia and Protopapas, 2012) and stream-water temperatures (Kamarianakis *et al.*, 2016).

In what follows, the univariate time series of interest is denoted by y_t. Let $\boldsymbol{x}_t' = [1, y_{t-1}, y_{t-2}, \cdots, y_{t-p}]$, $\boldsymbol{\alpha} = [\alpha_0, \alpha_1, \cdots, \alpha_p]$ and $\boldsymbol{\beta} = [\beta_0, \beta_1, \cdots, \beta_p]$. A parametric regime-switching time series model is formulated as:

$$y_t = (\boldsymbol{x}_t'\boldsymbol{\alpha})(1 - G(z_t, \gamma, \delta)) + (\boldsymbol{x}_t'\boldsymbol{\beta})G(z_t, \gamma, \delta) + \epsilon_t \text{ where } \epsilon_t \sim \text{i.i.d. } N(0, \sigma^2) \quad (2.1)$$

If $0 \leq G(z_t, \gamma, \delta) \leq 1$, Equation 2.1 represents a weighted average of two autoregressive processes of order p (AR(p)), with weights depending on the value of the transition variable z_t. When $z_t = y_{t-d}$ the model is a "self-exciting autoregression" and an additional delay parameter d is introduced (Petruccelli and Woolford, 1984). Equation 2.1 represents a logistic smooth transition autoregressive model of order p (LSTAR(p)) when $G(y_{t-d}, \gamma, \delta) = \{1 + \exp[-(\gamma/s_y)(y_{t-d} - \delta)]\}^{-1}$. If $y_{t-d} < \delta$, $G(y_{t-d}, \gamma, \delta) < 1/2$ and the AR(p) model $\boldsymbol{x}_t'\boldsymbol{\alpha}$ in the "low regime" is favored, whereas when $y_{t-d} > \delta$, $G(y_{t-d}, \gamma, \delta) > 1/2$ and the AR(p) model $\boldsymbol{x}_t'\boldsymbol{\beta}$ in the "high regime" receives larger weights. The slope γ determines the speed of transition between regimes; scaling γ by the sample standard deviation of the transition variable s_y allows for scale-free comparisons across competing STAR models with differing transition variables (Deschamps, 2008). As $\gamma \to \infty$, $G(y_{t-d}, \gamma, \delta) \to \mathbb{1}_{\{y_{t-d} > \delta\}}(y_{t-d})$ that evaluates to 1 if $y_{t-d} > \delta$ and 0 otherwise. Hence, in the limiting case when regime changes are abrupt, Equation 2.1 is equivalent to a threshold autoregressive model of order p (TAR(p)). Although this work focuses on the homoskedastic case, it is not hard to fathom the variance of y_t exhibiting regime switching dynamics along with the mean of y_t. Most research regarding STAR models revolves around the two regime case; however, extensions have been made to account for multiple (>2) regimes (MR-STAR) (Teräsvirta *et al.*, 2010).

Bayesian estimation of two-regime LSTAR(p) models was initially developed by (Lubrano, 2000). (Lopes and Salazar, 2006) expanded Lubrano's methodology to include the model order p in the vector of unknown parameters, using the reversible jump markov chain monte carlo (RJMCMC) algorithm presented in Green (1995). These changes were inspired by (Troughton and Godsill, 1997) who applied RJMCMC to AR(p) models. Further work by (Gerlach and Chen, 2008) accounted for regime-specific heteroskedasticity. Current Bayesian estimation methods of the LSTAR(p) typically assume that the autoregressive order p is the same in both regimes and estimate coefficients corresponding to all autoregressive terms y_{t-k} for $k \in \{1, 2, ..., p\}$. If the true nonlinear data generating process (DGP) has regime-specific orders with some autoregressive terms being non-significant, the above-mentioned method is expected to be suboptimal in terms of out-of-sample predictive accuracy as it is not flexible enough to capture the data generating process.

Section 2 explains how Bayesian estimation methods for sparse signals can be incorporated in the sampling algorithm for LSTAR models and how a *Dirichlet* prior may be used to estimate a generalized variant of the transition function. The proposed methodology is an alternative to existing stepwise model building strategies and to RJMCMC schemes: it estimates in a single step, specifications which encompass LSTAR and it may identify complex data generating mechanisms in which the values of transition function are determined by more than one threshold variable. Section 3 provides results from a series of Monte Carlo experiments showing the efficacy of the proposed methods. Section 4 presents a forecasting exercise based on benchmark data analyzed extensively in previous studies. Section 5 gives a positive outlook on how these methods may further advance Bayesian estimation of more complicated nonlinear processes and the last Section concludes the paper.

2.2 Methodology

2.2.1 Bayesian Estimation of LSTAR

For the 2-regime LSTAR(p) model in Equation 2.1 define the full vector of unknown parameters $\boldsymbol{\theta} = [\alpha_0, \alpha_1, \cdots, \alpha_p, \beta_0, \cdots, \beta_p, \gamma, \delta, \sigma, d, p]'$ where $\gamma = \gamma^*/s_y$. In regards to $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and σ^2, the prior specifications presented in previous research works are $\alpha_k \sim N(\mu_\alpha, \sigma^2_\alpha)$, $\beta_k \sim N(\mu_\beta, \sigma^2_\beta)$, $1/\sigma^2 \sim IG(a_{\sigma^2}, b_{\sigma^2})$, where N and IG respectively denote normal and inverse-gamma distributions. To ensure that sufficient representation exists in both regimes, the prior for δ is defined as $\delta \sim U[q_Y(0.15), q_Y(0.85)]$, where $q_Y(.)$ is the empirical quantile function of the observed transition variable and $U[a, b]$ represents the uniform distribution bounded on $[a, b]$. Using the 15^{th} and 85^{th} percentiles ensures that at least 15% of the data belongs to each regime. The parameter d is typically given a discrete uniform prior $P(d = \tilde{d}) = 1/d_{max}$ for $\tilde{d} \in \{1, 2, \cdots, d_{max}\}$, where d_{max} is chosen *a priori*.

Difficulties in the estimation of $\gamma = \gamma^*/s_y$ have led to a variety of prior proposals: $Cauchy$ (Lubrano, 2000), $Gamma$ (Lopes and Salazar, 2006), $truncated-Normal$ (Livingston Jr. and Nur, 2017), and $log-Normal$ (Gerlach and Chen, 2008). Livingston Jr. and Nur (2017) compared $Gamma$ and $truncated-Normal$ and demonstrated that computational time and posterior results are mainly influenced by starting values and prior information rather than distributional choice. Gerlach and Chen (2008) favored the $log-Normal$ (LN) prior $\gamma^* \sim LN(\mu_\gamma, \sigma^2_\gamma)$ over $Cauchy$, since it leads to an integrable posterior for γ and removes unnecessary prior weight placed at 0. Our preliminary analyses showed that the choice of prior had little effect on posterior distributions, confirming the findings by Livingston Jr. and Nur (2017). In what follows, $log-Normal$ priors are adopted for γ^* since it resulted in low computational times relative to $Gamma$, $Cauchy$ and $truncated-Normal$.

Sampling algorithms of the joint posterior $f(\boldsymbol{\theta}|\boldsymbol{y})$ exploit that the LSTAR(p) model is conditionally linear given γ^* and δ. Specifically, Gibbs sampling is applied for $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and σ^2 (Gelfand and Smith, 1990) and Metropolis-Hastings (Metropolis *et al.*, 1953; Hastings, 1970) for γ^* and δ. Since the length of $\boldsymbol{\theta}$ increases with the model order p, Lopes and Salazar (2006) extended the sampling algorithm outlined by Lubrano (2000) to incorporate a reversible jump step, which includes the model order p in $\boldsymbol{\theta}$. RJMCMC allows the dimension of the sampled vector $\boldsymbol{\theta}$ to change from $2(p+1)+3$

to $2(p' + 1) + 3$ whenever proposed changes from p to p' are accepted. Posterior assessment with regard to p relies on comparing posterior model probabilities. The most likely model order $\hat{p}$ is defined as $\hat{p} = \max_p \#\{p_s = p | s \in 1, \cdots, S\}/S$ where S represents the number of samples from the joint posterior distribution after burn-in and p_s represents the sampled value at iteration s.

2.2.2 *Sparse Estimation via Bayesian Shrinkage*

Let p_1 be the true linear AR model order in the low regime and p_2 be its equivalent in the high regime. Furthermore, let $p = \max\{p_1, p_2\}$. Two cases where RJMCMC may never sample from the correct parameter space are when $p_1 \neq p_2$ or when $\exists\ j < p$ such that $\alpha_j = 0 \cup \beta_j = 0$. The conditional linear nature of the LSTAR(p) model invites a plethora of Bayesian techniques for model selection through the editing of the priors for $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. For insights into the different varieties, the interested reader may consult O'Hara and Sillanpaa (2009).

Stochastic search variable selection (SSVS) builds a mixture of two normal distributions centered at 0, one with small variance and one with large variance, using indicator variables as the mixing weights (George and McCulloch, 1993). As different subsets of predictors are identified, coefficients of non-significant predictors are drawn toward 0 by the conditional nature of the priors in SSVS. Adaptive shrinkage methods, that achieve sparsity by using priors represented as scale mixtures of normal distributions, are shaped similarly to SSVS. An expanding list of hierarchical prior representations incorporates tuning parameters to perform shrinkage, by manipulating the amount of prior mass at zero and the shape of the tails (Polson and Scott, 2010). These methods are Bayesian analogs to penalized (regularized) estimates (Tibshirani, 1996), which have been employed to estimate linear time series models (Konzen and Ziegelmann, 2016; Nardi and Rinaldo, 2011).

Once a maximum order p is specified *a priori*, Bayesian shrinkage provides a flexible model building alternative; unlike RJMCMC, LSTAR(p) estimation can be performed using popular Bayesian software such as JAGS (Plummer, 2003). Furthermore, these methods may be applied to all models expressible by Equation 2.1, including exponential smooth transition autoregressive (ESTAR) and TAR models. Future discussion is limited to four prior hierarchical representations, varying in shrinkage flexibility.

Bayesian LASSO (BLASSO)

Andrews and Mallows (1974) demonstrated that the double-exponential distribution can be expressed as a scale-mixture of normal distributions. Their work leads to the two-level hierarchical representation depicted in Equation 2.2, where EXP denotes the *Exponential* distribution.

$$\begin{aligned} \alpha_j|\sigma^2, \tau^2_{\alpha_j} &\sim N(0, \sigma^2\tau^2_{\alpha_j}),\ \tau^2_{\alpha_j}| \sim EXP(\lambda^2/2) \\ \beta_j|\sigma^2, \tau^2_{\beta_j} &\sim N(0, \sigma^2\tau^2_{\beta_j}),\ \tau^2_{\beta_j}| \sim EXP(\lambda^2/2) \end{aligned} \tag{2.2}$$

The hyperparameter λ controls shrinkage across both regimes. In a linear context, as $\lambda \to \infty$ the path of posterior medians settles between the regularization paths under L_1 and L_2 penalties (Park and Casella, 2008); therefore, this method is often called Bayesian LASSO (BLASSO). In the LASSO of Tibshirani (1996), the tuning parameter λ is chosen via generalized cross validation. Rather than selecting a fixed λ, Bayesian procedures update this hyperparameter as MCMC moves through the posterior distribution (George and Foster, 2000; Casella, 2001; Yuan and Lin, 2005) using a *Gamma* hyperprior $\lambda^2 \sim G(a_\lambda, b_\lambda)$. The full Gibbs sampler outlined by Park and Casella (2008) can be extended to the LSTAR(p) model using a Metropolis-Hastings scheme for parameters $\{\gamma^*, \delta\}$.

Regime-Specific Bayesian LASSSO (RS-BLASSO)

A regime-specific variant of BLASSO, named (RS-BLASSO), employs two regime-specific tuning parameters λ_1 and λ_2, with independent gamma hyperpriors. The motivation for using two shrinkage parameters arrives from the understanding that sparseness may differ between the two regimes. A later simulation will identify a situation where this added flexibility is necessary for convergence. The corresponding modification to the BLASSO hierarchy is shown in Equation 2.3.

$$\tau^2_{\alpha_j}| \sim EXP(\lambda_1^2/2),\, \tau^2_{\beta_j}| \sim EXP(\lambda_2^2/2) \tag{2.3}$$

Variable Selection with Bayesian LASSO (VS-BLASSO)

A popular Bayesian subset selection method for linear models uses independent *Bernoulli* ($BERN$) distributed variables to indicate either inclusion or exclusion of a covariate (Kuo and Mallick, 1998). Lykou and Ntzoufras (2013) combined this subset selection method with the *double exponential* ($DEXP$) priors of BLASSO (VS-BLASSO). The BLASSO of Yuan and Lin (2005) also employs binary selection variables, but only in a SSVS context. Introducing latent binary variables ζ_j and η_j for $j \in \{1, 2, \cdots, p\}$, the autoregressive coefficients are reparamaterized to $\alpha_j = \zeta_j \alpha_j^*$ and $\beta_j = \eta_j \beta_j^*$ and the alternative prior hierarchy is seen in Equation 2.4. The tuning parameter λ handles global shrinkage, while the independent binary variables provide local variable selection. It is not unusual here for posterior medians of unnecessary parameters to be exactly zero. Combining these ideas opens the door to posterior comparisons of model probabilities and the easy incorporation of RJMCMC for faster convergence (Dellaportas *et al.*, 2002).

$$\begin{aligned} \zeta_j &\sim BERN(0.5),\, \alpha_j^*|\sigma^2 \sim DEXP\Big(0, \frac{\sigma^2}{\lambda}\Big) \\ \eta_j &\sim BERN(0.5),\, \beta_j^*|\sigma^2 \sim DEXP\Big(0, \frac{\sigma^2}{\lambda}\Big) \end{aligned} \tag{2.4}$$

Bayesian Horseshoe (BHS)

The horseshoe prior of Carvalho *et al.* (2009) can also be expressed as a scale-mixture of normals. Along with a global shrinkage parameter λ, Bayesian horseshoe adds local shrinkage parameters λ_{α_j} and λ_{β_j}. This change allows finer discrimination between relevant and non-significant autoregressive parameters by preventing the simultaneous over-shrinking that may occur to the parameter space in BLASSO. The Bayesian horseshoe (BHS) prior hierarchy described by (Carvalho *et al.*, 2010) is presented in Equation 2.5 with C^+ denoting the *half-Cauchy* distribution. Unfortunately, posterior sampling of α_j and β_j does not compare to the ease of the Gibbs sampler for BLASSO since full conditional distributions cannot be found analytically; however, fast slice sampling methods have been developed (Hahn *et al.*, 2016).

$$\begin{aligned} \alpha_j|\lambda_{\alpha_j} &\sim N(0, \lambda_{\alpha_j}),\ \beta_j|\lambda_{\beta_j} \sim N(0, \lambda_{\beta_j}) \\ \lambda_{\alpha_j} &\sim C^+(0, \lambda),\ \lambda_{\beta_j} \sim C^+(0, \lambda) \\ \lambda|\sigma^2 &\sim C^+(0, \sigma) \end{aligned} \tag{2.5}$$

2.2.3 Estimating the Delay Parameter

Till now, the delay parameter d was assumed known as in Gerlach and Chen (2008) whose focus was on regime-specific heteroskedasticity. However, this assumption is unreasonable in applications. The discrete uniform prior has been used for d since Lubrano (2000) and Lopes and Salazar (2006). The discrete uniform prior restricts popular Bayesian MCMC software from incorporating the delay parameter in MCMC posterior sampling. This problem can be circumvented by building LSTAR specifications for a finite set of prospective values of d and then choosing the model with the highest posterior probability (Deschamps, 2008). For model order p, one may consider all possible threshold variables y_{t-d} for $d \in \{1, 2, \cdots, p\}$; purposefully overestimating p can lead to a tedious procedure for choosing the delay.

Let $\boldsymbol{y} = [y_{t-1}, y_{t-2}, \cdots, y_{t-p}]'$, $\boldsymbol{\phi} = [\phi_1, \phi_2, \cdots, \phi_p]'$ and recall the transition function $G(z_t, \gamma, \delta) = \{1 + \exp[-(\gamma^*/s_y)(z_t - \delta)]\}^{-1}$ in Equation 2.1. A specification that nests LSTAR contains a new threshold variable $z_t = \boldsymbol{\phi}'\boldsymbol{y} = \sum_{k=1}^{p} \phi_k y_{t-k}$ expressed as a linear combination of possible threshold variables. The vector $\boldsymbol{\phi}$ adds p new parameters to $\boldsymbol{\theta}$ while providing flexibility in the selection of z_t. A naive estimation approach is to let $\phi_j \sim$ i.i.d. $BERN(1/p)$ for $j \in \{1, 2, \cdots, p\}$. This leads to the possibility that the threshold variable z_t is expressed as the sum of multiple lags of the endogenous series y_t, in contrast with conventional LSTAR where $\phi_k = 0\ \forall k \neq d$. Since prior of δ is chosen conditionally on the empirical distribution of y_t, fair representation in regimes cannot be enforced when $\phi_k = 1$ for more than one value of $k \in \{1, 2, \cdots, p\}$. A possible remedy is to let $\delta = \phi^*\delta^*$ where $\phi^* = \sum \phi_k$ and the prior for $\delta^* \sim U[q_Y(0.15), q_Y(0.85)]$. Simulation results, not shown here for brevity, reveal that full Bayesian estimation is possible, but extremely slow, making this method impractical.

Applying the constraint $\sum \phi_k = 1$ eliminates the previous issues involving δ: z_t becomes a weighted average of multiple lags of y_t rather than a summation. Consider the following p-dimensional *Dirichlet* (Dir) prior distribution: $\boldsymbol{\phi} \sim Dir([\frac{1}{p}, \frac{1}{p}, \cdots, \frac{1}{p}])$. Often times the *Dirichlet* distribution is used for its conjugacy in multinomial and categorical models. The application in this context relates more to the usage in multivariate regressions on compositional data (Campbell and Mosimann, 1987; Hijazi and Jernigan, 2009). Posterior assessment of $\boldsymbol{\phi}$ can either heavily point to a specific delay parameter or provide evidence of a composite threshold variable. The next Section shows that combining this prior specification for $\boldsymbol{\phi}$ with Bayesian shrinkage provides accurate signal detection without causing a significant drop in convergence speed.

2.3 Monte Carlo Simulations

2.3.1 Simulation 1: Well-Behaved LSTAR

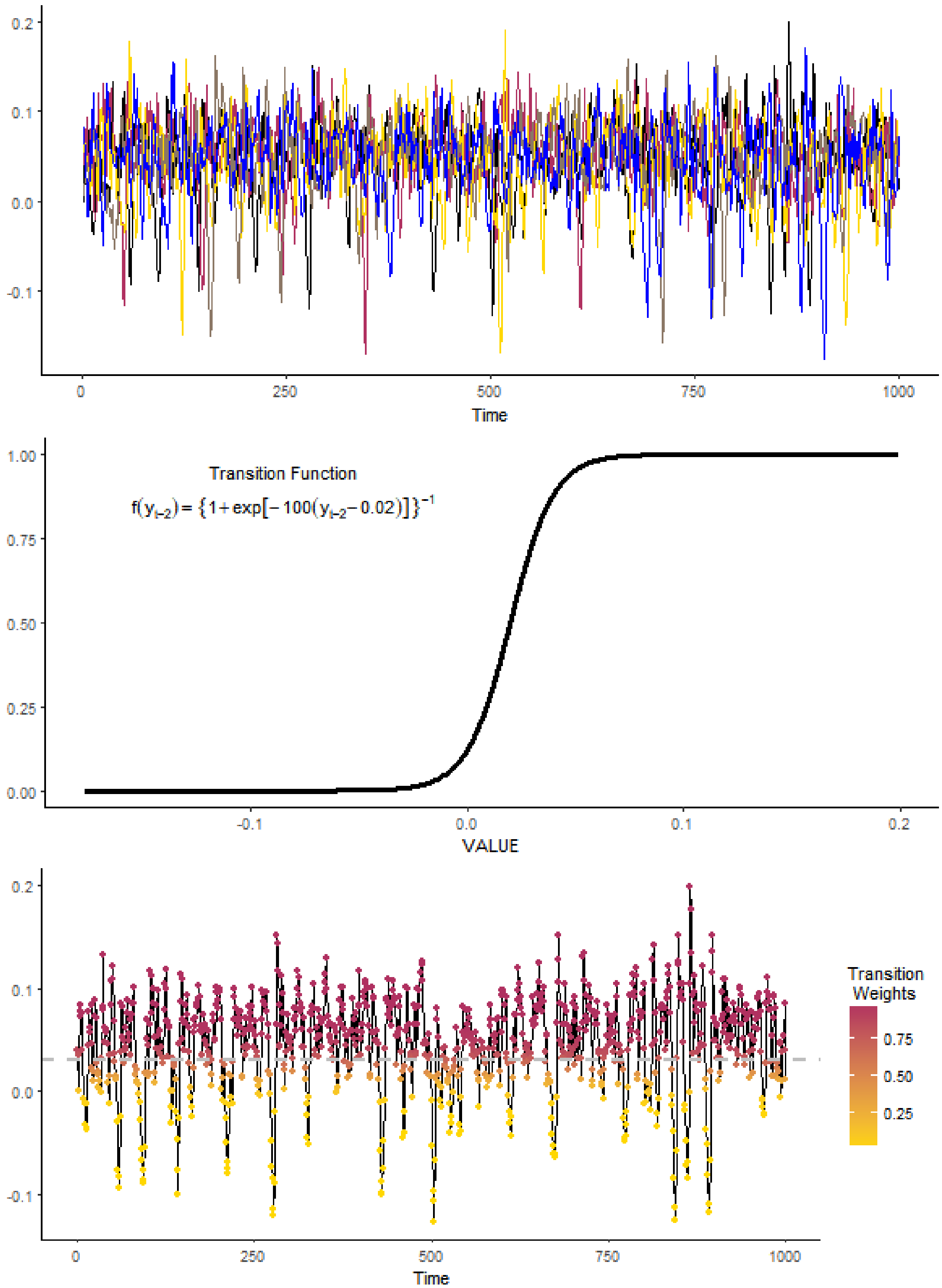
The first experiment is based on 100 replicates of the LSTAR(2) model in Equation 2.6, each of length 1000, after a burn-in period of 500. Figure 2.1

$$
\begin{aligned}
y_t &= (1.8y_{t-1} - 1.06y_{t-2})[1 - G(y_{t-2})] \\
&\quad + (0.02 + 0.9y_{t-1} - 0.265y_{t-2})[G(y_{t-2})] + \epsilon_t \\
G(y_{t-2}) &= \Big\{1 + \exp\big[-100(y_{t-2} - 0.02)\big]\Big\}^{-1} \\
\epsilon_t &\sim \text{i.i.d. } N(0, 0.02^2)
\end{aligned}
\tag{2.6}
$$

This model is identical to the one presented in Lopes and Salazar (2006) where RJMCMC is used to select model order and a discrete $Uniform$ is adopted for d. If $p = 4$ is known *a priori*, the true parameter vector $\boldsymbol{\theta} = [\alpha_0, \alpha_1, \cdots, \alpha_4, \beta_0, \cdots, \beta_4, \gamma, \delta, \sigma]'$ contains 5 zero parameters. Until further notice, d is assumed to be known while the focus is on the ability of Bayesian shrinkage to combat over-fitting.

Bayesian estimation of the underlying LSTAR(2) model compares BLASSO, RS-BLASSO, VS-BLASSO, and BHS shrinkage priors to conventional Normal priors. For the variations of BLASSO, posterior medians are obtained (Park and Casella, 2008), whereas when BHS (Carvalho *et al.*, 2009) or Normal priors are used, point estimates are based on posterior means. After a burn-in period of 15,000 and a thinning of 10 to reduce autocorrelation and control computer memory usage, 1,000 initial samples are obtained for 3 chains from the joint posterior distribution. The "potential scale reduction factor" (PSRF(θ)) of Gelman and Rubin (1992) evaluates convergence across the three chains, and effective sample size (ESS(θ)) measures mixing efficiency for each parameter $\theta \in \boldsymbol{\theta}$. If $\max_\theta$ PSRF(θ) < 1.05 and $\min_\theta$ ESS(θ) > 150, convergence criteria is met and our initial sample is sufficient; otherwise, posterior samples

Figure 2.1: Ten Random Replications (top), Transition Function (middle), and Illustration of Regime-switching Behavior for Simulation 1



are added, a maximum of 20 times, with intermittent convergence checks. Posterior simulations are only considered valid whenever the convergence criteria is met. Upcoming sections will follow the same convergence and reporting standards. Prior hyper-parameters are intentionally chosen to be non-informative and starting values are either randomly chosen or over-dispersed. Specifically for γ^*, a non-informative log normal prior $LN(3,1)$ is selected for all simulation experiments (Gerlach and Chen, 2008).

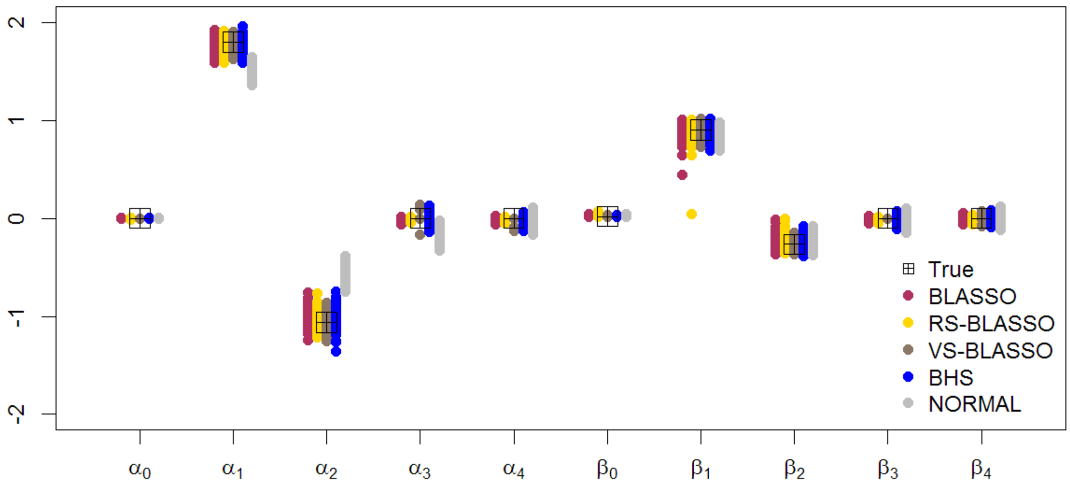
Table 2.1 provides summary statistics of the posterior estimates from replications that converged using all five prior specifications. Rather than reporting the standard deviation of the estimates, estimation error is summarized using $RMSE(\theta) = \sqrt{\sum(\hat{\theta}-\theta)^2/n}$. There is consistent overestimation and large uncertainty for γ — commonly reported in literature (Livingston Jr. and Nur, 2017) — with the worst results for BHS and Normal priors. Figure 2.2 plots posterior estimates of the autoregressive parameters $\hat{\boldsymbol{\alpha}}$ and $\hat{\boldsymbol{\beta}}$. Discerning between shrinkage estimation methods is difficult since signal detection is satisfactory for all 4 methods. The optimal choice may be determined solely on computational efficiency which is left for discussion in a future subsection. Clearly, substantial improvements can be seen over the default normal prior choice.

Comparing our results to Lopes and Salazar (2006) is difficult for a number of reasons. For 50 replications, they obtain one MCMC chain of 2,500 posterior samples after a burn-in of 5,000; prior hyper-parameters are not specified and initial values are fixed. In this well-behaved case, posterior model probabilities pointed to the correct model 49 out of 50 times. For each replication, estimates are only based on the posterior samples where RJMCMC visited the correct LSTAR(2) model; no information is given on how many of the 2,500 posterior samples come from the correct parameter space. In Lopes and Salazar (2006), overall summaries are based on the

Table 2.1: Posterior Estimate Summary for Simulation 1

		BLASSO		RS-BLASSO		VS-BLASSO		BHS		Normal	
Parameter	Actual	Mean	RMSE	Mean	RMSE	Mean	RMSE	Mean	RMSE	Mean	RMSE
α_0	0	0.0011	0.0024	0.0011	0.0025	0	0	0.0007	0.002	0.0049	0.0054
α_1	1.8	1.7666	0.0715	1.7696	0.068	1.7765	0.0624	1.768	0.0746	1.5165	0.2898
α_2	-1.06	-1.0046	0.1003	-1.0108	0.0924	-1.0367	0.081	-1.0104	0.1159	-0.5786	0.4888
α_3	0	-0.0042	0.0103	-0.0026	0.0066	0.0007	0.0237	-0.0125	0.0434	-0.1745	0.1878
α_4	0	-0.004	0.0142	-0.0027	0.0105	-0.0022	0.0155	-0.0028	0.031	-0.0125	0.0618
β_0	0.02	0.0199	0.0039	0.0201	0.0067	0.0205	0.0032	0.0202	0.0035	0.021	0.0043
β_1	0.9	0.8714	0.0772	0.8697	0.1069	0.8987	0.0475	0.8899	0.0528	0.8635	0.0661
β_2	-0.265	-0.2246	0.0732	-0.2253	0.0748	-0.2684	0.0406	-0.2496	0.053	-0.2366	0.0584
β_3	0	-0.0075	0.0139	-0.0057	0.0114	0	0	-0.0081	0.0296	-0.0083	0.0504
β_4	0	-0.0005	0.0151	-0.0009	0.012	-0.0003	0.0132	0.0019	0.0253	0.0033	0.0404
σ	0.02	0.0202	0.0004	0.0202	0.0004	0.0201	0.0004	0.0201	0.0004	0.021	0.0011
γ	100	131.0624	80.8034	131.5355	82.1307	131.0224	77.9754	174.15	148.8093	225.5381	204.5109
δ	0.02	0.0218	0.0049	0.0218	0.0056	0.0204	0.0035	0.0208	0.0038	0.0261	0.0083

Figure 2.2: Plot of $\hat{\boldsymbol{\alpha}}$ and $\hat{\boldsymbol{\beta}}$ for Simulation 1



standard deviation (SD) of posterior estimates, whereas RMSE is evaluated here. The SD summarizes how much the posterior estimates differed from each other, while RMSE shows how much the posterior estimates differed from the truth. The main purpose for repeating this study is not to compare RJMCMC to Bayesian shrinkage but to establish the efficacy of these alternative methods for estimating a relatively simple LSTAR model.

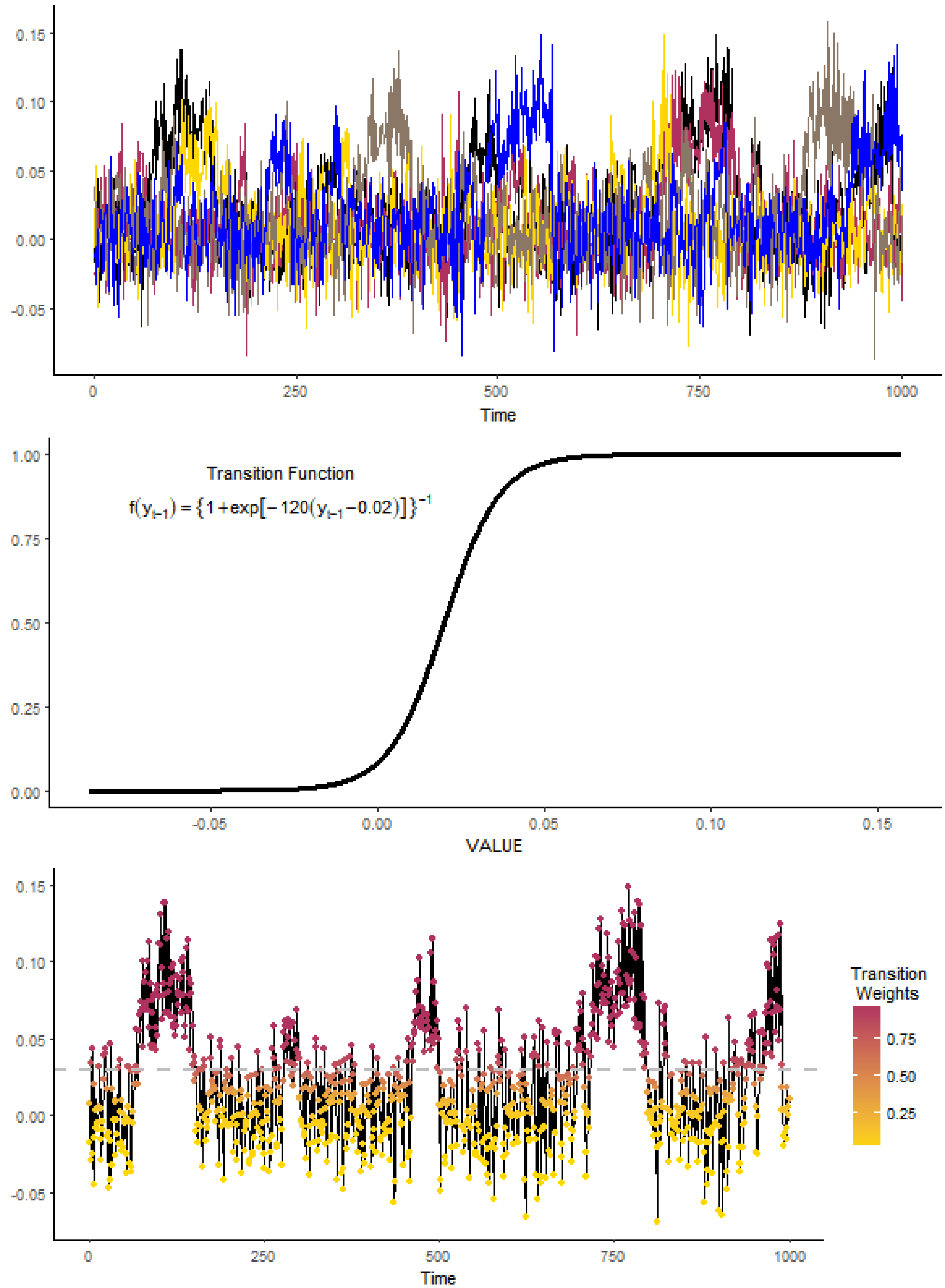
2.3.2 Simulation 2: LSTAR with Gaps and Incremental Changes to Error Variance

The next experiment is based on 100 replications of the LSTAR(3) model described in Equation 2.7.

$$\begin{aligned} y_t &= (-0.6y_{t-3})[1-G(y_{t-1})] + (0.02 + 0.75y_{t-3})[G(y_{t-1})] + \epsilon_t \\ &\text{where: } G(y_{t-1}) = \left\{1 + \exp\Big[-120(y_{t-1} - 0.02)\Big]\right\}^{-1} \\ &\text{and } \epsilon_t \sim \text{i.i.d. } N(0, 0.02^2) \end{aligned} \tag{2.7}$$

Under the assumption that $p = 4$, coefficients $\boldsymbol{\theta}$ for autoregressive lags less than and larger than 3 are truly zero. This is a situation where even if RJMCMC visits

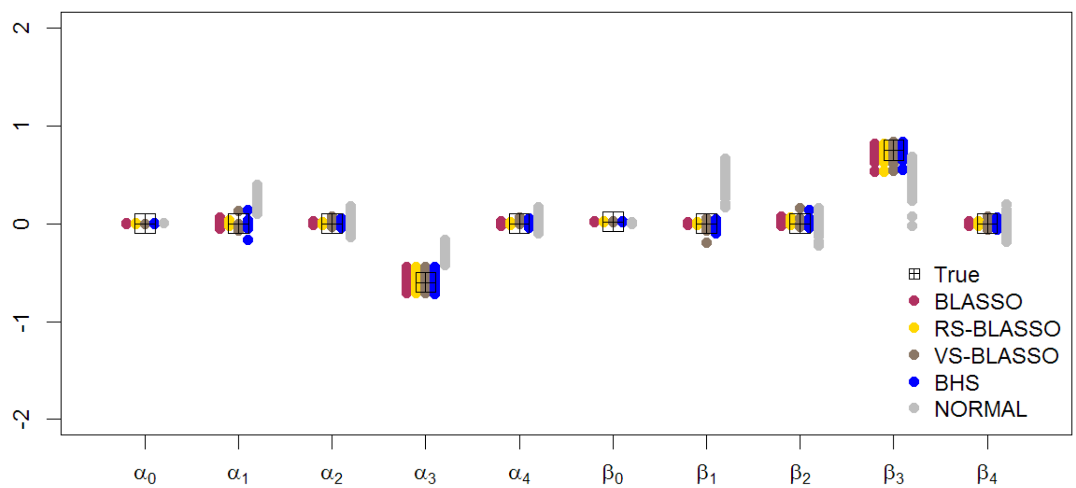
Figure 2.3: Ten Random Replications (top), Transition Function (middle), and Illustration of Regime-switching Behavior for Simulation 2



the correct parameter space, normal priors will result in over-fitting. Motivation is geared towards nonlinear models where seasonal dynamics, of any period length, exhibit nonlinearities through dependence on some threshold variable.

Table 2.2 summarizes and Figure 2.4 illustrates the estimation accuracy of each method. Again, the normal priors result in unsatisfactory estimation accuracy. The simplest shrinkage methods, BLASSO and RS-BLASSO, consistently identify the true signal slightly better than the other shrinkage methods.

Figure 2.4: Plot of $\hat{\boldsymbol{\alpha}}$ and $\hat{\boldsymbol{\beta}}$ for Simulation 2



Additionally, Simulation 2 is modified to allow $\sigma_k = 0.02k\ \forall k \in \{1, 2, \cdots, 5\}$. For 50 replicates under each proposed σ, BLASSO and BHS methods are applied. RMSE(θ) is naturally expected to increase with σ. The desire is to explore the sensitivity of RMSE(θ) as the noise is amplified. Under a fixed transition slope $\gamma = 120$, a contradictory trend was initially observed for known nonzero parameters α_3 and β_3. In Table 2.3, RMSE(α_3) and RMSE(β_3) gradually decline when γ is fixed implying improved estimation. Increasing σ naturally increases the sample standard deviation s_y. Under the reparameterization $\gamma = \gamma^*/s_y$, the unscaled transition slope γ^* naturally must increase with s_y to obtain the predetermined $\gamma = 120$.

Table 2.2: Posterior Estimate Summary for Simulation 2

		BLASSO		RS-BLASSO		VS-BLASSO		BHS		Normal	
Parameter	Actual	Mean	RMSE	Mean	RMSE	Mean	RMSE	Mean	RMSE	Mean	RMSE
α_0	0	0.0003	0.0011	0.0002	0.0011	0	0	0.0002	0.001	0.0041	0.0042
α_1	0	0.0016	0.0097	0.0008	0.0051	-0.0003	0.0157	0.0018	0.0273	0.2484	0.2556
α_2	0	0.0004	0.007	0.0003	0.0041	0.0011	0.0094	0.0008	0.0154	-0.0014	0.0517
α_3	-0.6	-0.583	0.0547	-0.5838	0.0544	-0.5849	0.0548	-0.5896	0.0537	-0.2916	0.313
α_4	0	-0.0012	0.0073	-0.0007	0.0044	0.0011	0.0075	-0.0016	0.0149	0.0114	0.0529
β_0	0.02	0.0201	0.0019	0.0201	0.0018	0.0206	0.0021	0.0203	0.0021	0.0011	0.0195
β_1	0	0.0004	0.0051	0.0004	0.0044	-0.0032	0.022	-0.0022	0.0214	0.4269	0.4386
β_2	0	0.0005	0.0097	0.0006	0.0086	0.0016	0.0167	0.0002	0.0204	0.0008	0.0673
β_3	0.75	0.7307	0.046	0.73	0.0463	0.7306	0.0467	0.734	0.0439	0.5085	0.2764
β_4	0	-0.0001	0.0067	0.0001	0.006	-0.0004	0.0103	-0.0012	0.0159	-0.0113	0.0648
σ	0.02	0.0201	0.0004	0.0201	0.0004	0.0201	0.0004	0.02	0.0004	0.0232	0.0033
γ	120	130.5062	19.0649	130.3244	18.9201	128.4834	17.3982	130.9414	19.6161	686.7313	908.9353
δ	0.02	0.0202	0.0013	0.0202	0.0013	0.0202	0.0013	0.0202	0.0013	0.025	0.007

Clearly, changing σ has an impact on γ^* through s_y. Although the actual transition function is not changing with σ since it is fully determined by γ and δ, the speed of transition is increasing due to the natural modifications in the scope of the simulated data. The change is more visual in this regard. Therefore, for target $\gamma^* \approx 4$, data is simulated with $\gamma_k \approx 4/s_y$. Since σ will naturally not equal s_y, the initial replications for each σ_k under fixed $\gamma = 120$ were used to obtain a mean estimate of s_y. Then, an appropriate γ_k is determined for each σ_k, and 50 new replications are obtained. Table 2.3 shows the RMSE(θ) of each parameter for the specified options of σ under fixed $\gamma_k = 120$ and modified γ_k to target $\gamma^* = 4$. From these changes to the simulated data, RMSE(α_3) and RMSE(β_3) increase with σ_k. Interestingly, the pattern for RMSE(γ) is now reversed. These observations are prevalent under both BLASSO and BHS priors; nevertheless, Bayesian shrinkage priors efficiently identify the nonlinear signal under gradual increases to the noise.

2.3.3 *Simulation 3: LSTAR with Regime-Specific Sparsity*

The effectiveness of the proposed methods is evaluated via simulating 100 replicates of the LSTAR(3) model in Equation 2.8, exhibiting regime-specific complexity: the autoregressive dynamics are far simpler in the low regime relative to the high regime.

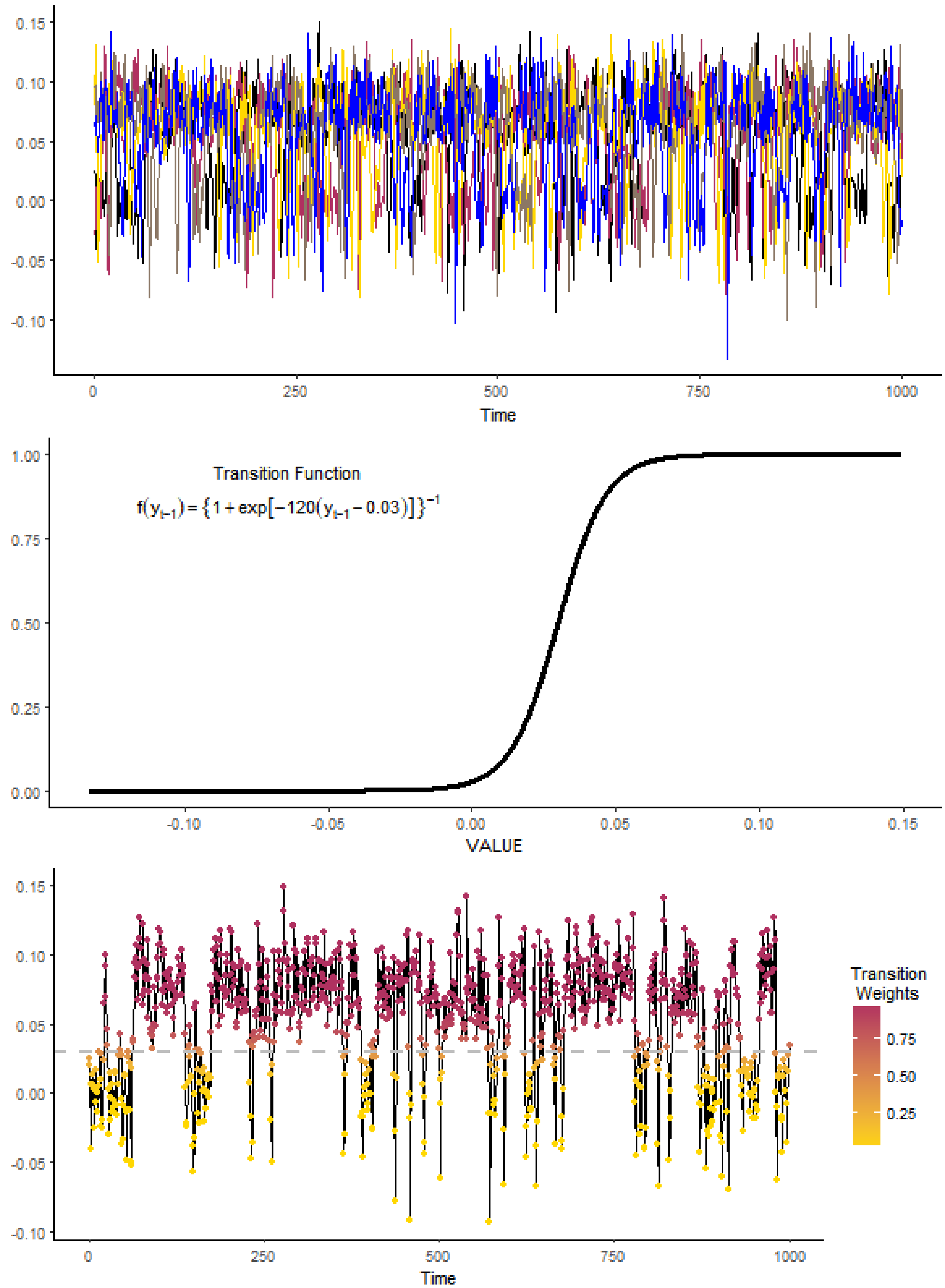
$$
\begin{aligned}
y_t = {} & (-0.7y_{t-3})[1 - G(y_{t-1})] \\
& + (0.06 + 0.4y_{t-1} - 0.35y_{t-2} + 0.2y_{t-3})[G(y_{t-1})] + \epsilon_t \\
& \text{where: } G(y_{t-1}) = \Big\{1 + \exp\big[-120(y_{t-1} - 0.03)\big]\Big\}^{-1} \\
& \text{and } \epsilon_t \sim \text{i.i.d. } N(0, 0.02^2)
\end{aligned}
\tag{2.8}
$$

From Table 2.4 and Figure 2.6, one observes that estimation accuracy is satisfactory for all three shrinkage methods, whereas *Normal* priors continue to lead to poor

Table 2.3: Sensitivity Analysis of RMSE($\boldsymbol{\theta}$) to σ in Simulation 2

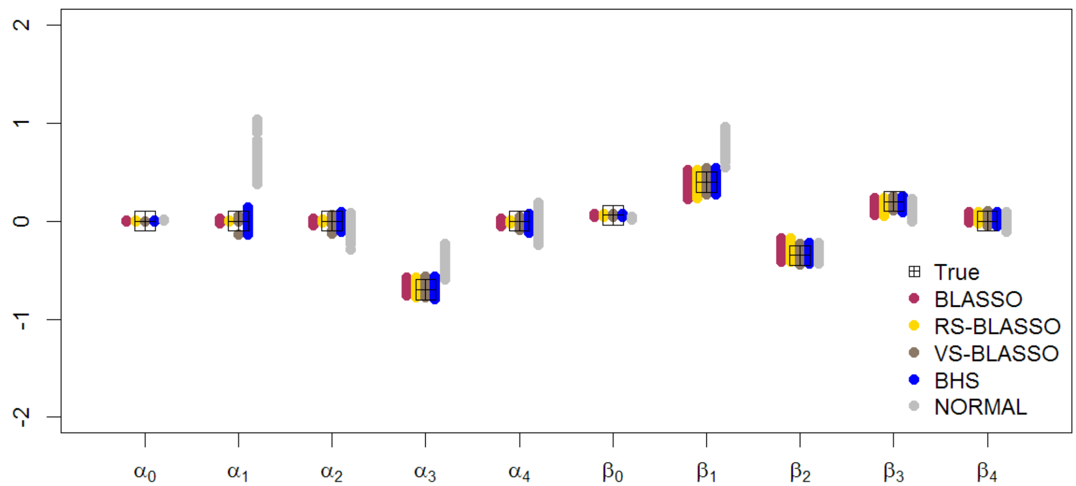
Method	Parameter	Fixed Transition Slope					Modified Transition Slope				
	Choice of σ	0.02	0.04	0.06	0.08	0.1	0.02	0.04	0.06	0.08	0.1
	Choice of γ			120			109.60	71.47	49.50	37.46	30.02
BLASSO	α_0	0.0009	0.0016	0.002	0.0025	0.0032	0.001	0.0016	0.0027	0.004	0.0049
	α_1	0.0068	0.0104	0.0123	0.0145	0.013	0.0091	0.0141	0.0205	0.0248	0.0258
	α_2	0.0125	0.0116	0.0102	0.0108	0.0102	0.0121	0.0136	0.0117	0.0127	0.0136
	α_3	0.0479	0.0443	0.0383	0.0324	0.0328	0.0501	0.0522	0.055	0.0552	0.0543
	α_4	0.0119	0.0128	0.008	0.0089	0.0106	0.012	0.0109	0.0097	0.0097	0.0099
	β_0	0.0019	0.0025	0.0038	0.0048	0.0059	0.0019	0.0027	0.0042	0.0057	0.0069
	β_1	0.006	0.0099	0.0152	0.0171	0.0177	0.0069	0.0066	0.0195	0.0227	0.0218
	β_2	0.0091	0.0147	0.0165	0.0154	0.0151	0.0127	0.0141	0.0154	0.0181	0.02
	β_3	0.0494	0.0429	0.0427	0.0438	0.0403	0.0579	0.061	0.0651	0.0683	0.0707
	β_4	0.008	0.0163	0.0136	0.0148	0.0204	0.0093	0.0202	0.0209	0.0195	0.0193
	σ	0.0005	0.0009	0.0014	0.0018	0.0023	0.0005	0.0009	0.0014	0.0018	0.0023
	γ	18.1887	23.4943	32.6301	32.8937	45.224	17.961	12.7347	9.5208	8.1114	7.1989
	δ	0.0013	0.0019	0.0021	0.0025	0.0025	0.0013	0.0022	0.0038	0.0052	0.0065
HS	α_0	0.0011	0.002	0.0028	0.0036	0.0049	0.0011	0.002	0.0033	0.0049	0.0065
	α_1	0.0054	0.0127	0.0186	0.0236	0.0251	0.0063	0.0144	0.0233	0.03	0.0345
	α_2	0.0075	0.013	0.0157	0.018	0.0181	0.0074	0.0141	0.0162	0.0192	0.0223
	α_3	0.0485	0.0451	0.0386	0.0329	0.0334	0.0508	0.0545	0.0575	0.0576	0.0568
	α_4	0.0079	0.0139	0.0138	0.0155	0.0174	0.008	0.0134	0.0159	0.0181	0.02
	β_0	0.0018	0.0024	0.0037	0.0046	0.0058	0.0018	0.0026	0.0039	0.0053	0.0067
	β_1	0.004	0.0111	0.0151	0.0204	0.0242	0.0039	0.0081	0.0162	0.022	0.0258
	β_2	0.0068	0.0152	0.0196	0.0214	0.0237	0.0071	0.014	0.0189	0.0226	0.0261
	β_3	0.0524	0.044	0.0442	0.0451	0.042	0.0608	0.0642	0.0688	0.072	0.0739
	β_4	0.0062	0.0164	0.0181	0.0215	0.0258	0.0063	0.0183	0.0227	0.024	0.0259
	σ	0.0005	0.0009	0.0014	0.0018	0.0023	0.0005	0.0009	0.0014	0.0018	0.0023
	γ	19.0596	24.0976	33.568	33.4267	45.3324	19.0212	13.5966	10.205	8.596	7.5575
	δ	0.0013	0.0019	0.0021	0.0025	0.0025	0.0014	0.0022	0.0038	0.0052	0.0065

Figure 2.5: Ten Random Replications (top), Transition Function (middle), and Illustration of Regime-switching Behavior for Simulation 3



estimates. The motivation for regime-specific shrinkage parameters λ_1 and λ_2 is illustrated in Figure 2.7, which presents histograms of posterior median estimates for the tuning parameters of BLASSO vs RS-BLASSO. The visual disparity between λ_1 and λ_2 is a result of the regime-specific sparsity patterns: $\lambda_1 > \lambda_2$ necessitates from the lower regime requiring relatively more shrinkage to identify the underlying signal.

Figure 2.6: Plot of $\hat{\boldsymbol{\alpha}}$ and $\hat{\boldsymbol{\beta}}$ for Simulation 3



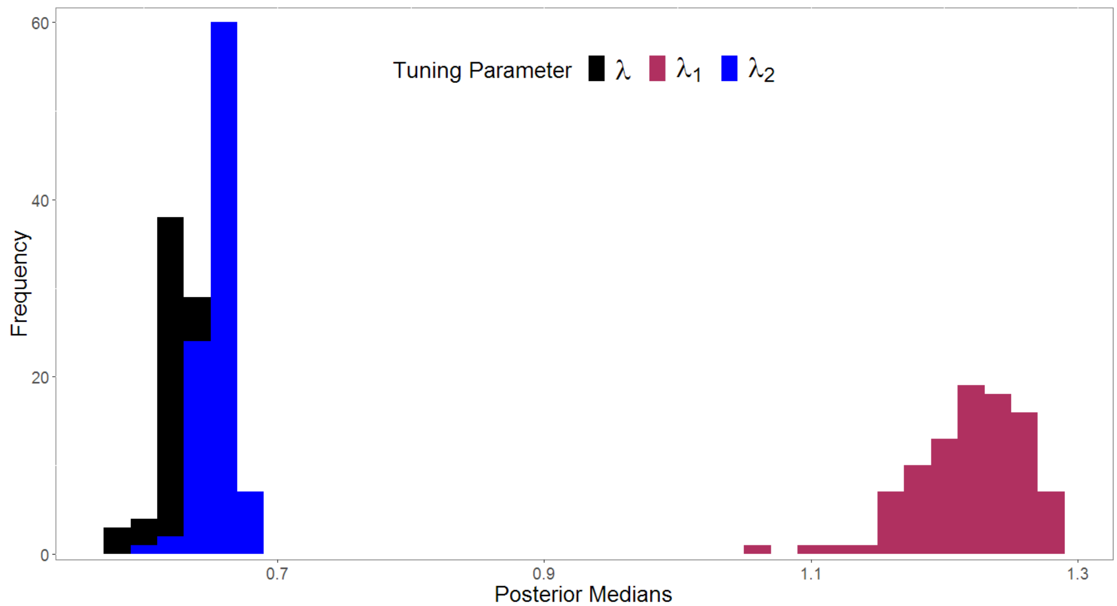
2.3.4 Convergence Analysis

Simulations 1-3 were conducted on an Intel(R) Xeon (R) CPU E5-2697 v3 @ 2.60 GHz server with 132GB of RAM and 56 cores. Both the replications and the MCMC chains were parallel-processed within limitations of the server. The resources were often shared amongst colleagues, hence computational times can be misleading as a measure of efficiency. Since the parameter space is fixed for each MCMC routine, efficiency can be measured by the number of posterior samples required to attain the convergence criteria. Table 2.5 reports the percent of the 100 replications that converged along with the mean, median, and extreme percentiles of the samples required for the replications that converged.

Table 2.4: Posterior Estimate Summary for Simulation 3

		BLASSO		RS-BLASSO		VS-BLASSO		BHS		Normal	
Parameter	Actual	Mean	RMSE	Mean	RMSE	Mean	RMSE	Mean	RMSE	Mean	RMSE
α_0	0	-0.0003	0.0019	0	0.0017	0	0.0003	0	0.002	0.0125	0.0126
α_1	0	0.0011	0.0092	0.0003	0.0026	-0.0004	0.0151	0.003	0.0431	0.6015	0.6135
α_2	0	-0.0031	0.0123	-0.0006	0.0035	-0.0035	0.0244	-0.0058	0.0375	-0.093	0.1118
α_3	-0.7	-0.6785	0.0448	-0.6816	0.0439	-0.6754	0.052	-0.6803	0.0553	-0.4609	0.2483
α_4	0	-0.0016	0.0131	-0.0005	0.0044	-0.0008	0.0118	-0.0026	0.0311	-0.056	0.1002
β_0	0.06	0.0612	0.0046	0.0613	0.0042	0.061	0.0042	0.0609	0.0042	0.0301	0.0305
β_1	0.4	0.3692	0.0674	0.37	0.0613	0.3802	0.0545	0.3809	0.056	0.7792	0.3863
β_2	-0.35	-0.3054	0.0695	-0.3005	0.0717	-0.3292	0.0508	-0.3242	0.0536	-0.3312	0.0501
β_3	0.2	0.1597	0.0596	0.1538	0.0643	0.1844	0.0375	0.1766	0.0454	0.1371	0.0771
β_4	0	0.0104	0.0202	0.0101	0.0198	0.0015	0.0128	0.0063	0.0225	-0.0087	0.039
σ	0.02	0.0202	0.0005	0.0202	0.0004	0.0201	0.0004	0.0201	0.0004	0.0239	0.0039
γ	120	120.5942	9.7438	121.8142	10.8705	121.3552	10.2547	123.1839	11.4902	617.4922	740.3125
δ	0.03	0.0302	0.0011	0.0303	0.0011	0.0303	0.001	0.0302	0.0011	0.0378	0.0089

Figure 2.7: Posterior Shrinkage Comparison for BLASSO vs RS-BLASSO



When the model order p is overestimated, the four shrinkage methods resist over-fitting to identify the true nonlinear process; therefore, choosing a method in practice ultimately depends on computational feasibility. All methods were equally efficient for Simulation 2 and unaffected by changes in noise. The methods were organized in order of regularization flexibility. For Simulations 1 and 3, the percent of converged replicates increased with the aforementioned flexibility. Specifically for Simulation 3, the additional tuning parameter in RS-BLASSO increased this percentage by 19%, identifying the true advantage for regime-specific shrinkage. The BHS hierarchy is commended for being consistently efficient.

2.3.5 Bayesian Selection of the Threshold Variable

To incorporate the uncertainty for the delay d, Simulation 1 is revisited where the true threshold variable y_{t-2} was assumed to be known. Maintaining the assumption $p = 4$, the vector $\boldsymbol{y} = [y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4}]'$ contains the threshold variables of interest.

Table 2.5: Convergence Statistics for Estimation Methods From Simulations 1-3

		Percent of Replications	Summary Statistics of Samples Required			
Simulation	Method	Converged	5th Percentile	Mean	Median	95th Percentile
1	BLASSO	91%	1000	11615	2000	67000
1	RS-BLASSO	96%	1000	10188	2000	110125
1	VS-BLASSO	99%	1000	3202	2000	11000
1	BHS	100%	1000	1600	1000	4000
1	Normal	100%	1000	1360	1000	4000
2	BLASSO	100%	1000	1000	1000	1000
2	RS-BLASSO	100%	1000	1000	1000	1000
2	VS-BLASSO	100%	1000	2120	1000	4000
2	BHS	100%	1000	1010	1000	1000
2	Normal	100%	1000	1150	1000	3050
3	BLASSO	75%	1000	15800	2000	131500
3	RS-BLASSO	94%	1000	1723	1000	4000
3	VS-BLASSO	100%	1000	1380	1000	4000
3	BHS	99%	1000	1010	1000	1000
3	Normal	99%	2000	8232	4000	46000

The re-paramaterized transition function $G(\boldsymbol{y}) = \{1 + \exp[-100(\boldsymbol{\phi}'\boldsymbol{y} - 0.02)]\}^{-1}$ is equivalent to the transition function in Equation 2.6 when $\boldsymbol{\phi} = [\phi_1, \phi_2, \phi_3, \phi_4]' = [0, 1, 0, 0]'$. Posterior sampling of $\boldsymbol{\phi}$ is combined with BLASSO and BHS under the previously stated convergence requirements.

First, independent Bernoulli priors were used for ϕ_k along with BLASSO. Only 43% of the replications converged compared to 91% when $d = 2$ was known. The average of the 43 posterior means for $\boldsymbol{\phi}$ was $[0.223, 0.988, 0.206, 0.040]'$; the independent Bernoulli priors do not limit the threshold variable to one choice in $\{y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4}\}$ since $\sum \phi_k \neq 1$ is not enforced. Estimation accuracy for non-$\boldsymbol{\phi}$ parameters was similar to the results presented in the previous Sections, but Bernoulli priors will not be discussed further due to computational deficiencies.

Next, let $\boldsymbol{\phi} \sim Dir([0.25, 0.25, 0.25, 0.25]')$; the uninformative hyper-parameter demonstrates prior impartiality regarding d. BLASSO and BHS are combined with the *Dirichlet* prior to re-estimate the 100 replications in Simulation 1, of which 98% and 100% converged, respectively. Table 2.6 uses the RMSE of non-$\boldsymbol{\phi}$ parameters to show that MCMC sampling for $\boldsymbol{\phi}$ does not render the previous estimation methods useless. Table 2.7 depicts summary statistics of the posterior means for $\boldsymbol{\phi}$ from the replications that converged. Figure 2.8 overlays the posterior means summarized in Table 2.7. Both star plots heavily point toward the correct threshold variable indicating accurate estimation of $\boldsymbol{\phi}$.

Table 2.6: Sensitivity of RMSE($\boldsymbol{\theta}$) to Uncertainty about $\boldsymbol{\phi}$ in Simulation 1

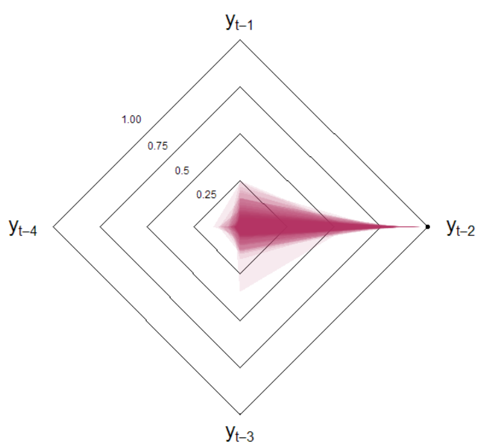
	$\boldsymbol{\phi} \sim Dir$		$\boldsymbol{\phi}$ Known	
Parameter	BLASSO	BHS	BLASSO	BHS
α_0	0.0027	0.002	0.0024	0.002
α_1	0.0763	0.0827	0.0715	0.0746
α_2	0.1048	0.1269	0.1003	0.1159
α_3	0.0103	0.038	0.0103	0.0434
α_4	0.0153	0.0265	0.0142	0.031
β_0	0.0072	0.0034	0.0039	0.0035
β_1	0.1091	0.0541	0.0772	0.0528
β_2	0.0745	0.0549	0.0732	0.053
β_3	0.0147	0.0271	0.0139	0.0296
β_4	0.0156	0.0245	0.0151	0.0253
σ	0.0004	0.0004	0.0004	0.0004
γ	87.1606	91.2016	80.8034	148.8093
δ	0.0058	0.004	0.0049	0.0038

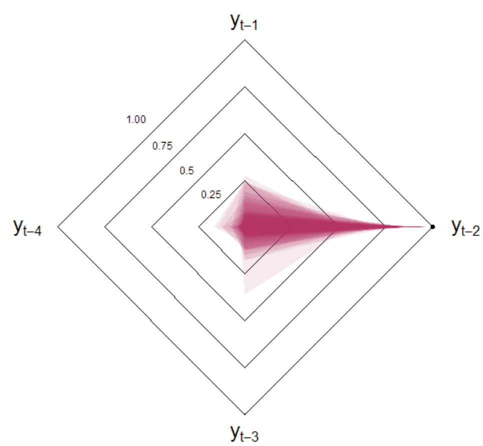
Simulation 2 is repeated for three threshold variables denoted $z_{1,t}$, $z_{2,t}$, and $z_{3,t}$ and identified in Equation 2.9. The first two threshold variables conform to the classic LSTAR structure; however, $z_{3,t}$ is an average of the first three lags of the endogenous

Table 2.7: Posterior Estimate Summary for $\boldsymbol{\phi}$ in Simulation 1

		BLASSO			BHS		
Parameter	Actual	5th %-ile	Mean	95th %-ile	5th %-ile	Mean	95th %-ile
ϕ_1	0	0.0114	0.0557	0.1618	0.0110	0.0581	0.1920
ϕ_2	1	0.6501	0.8686	0.9536	0.6008	0.8653	0.9527
ϕ_3	0	0.0144	0.0465	0.1301	0.0143	0.0475	0.1254
ϕ_4	0	0.0079	0.0292	0.0897	0.0082	0.0290	0.0810

Figure 2.8: Posterior Means of $\boldsymbol{\phi}$ for Simulation 1 Using BLASSO (left) and BHS (right)





time series $\boldsymbol{y_t}$. Conventional estimation of the delay d would be unable to correctly identify $z_{3,t}$. Using BHS only, all 3 modifications are identifiable when a 4-dimensional *Dirichlet* prior is used for $\boldsymbol{\phi}$.

$$\begin{aligned}
z_{1,t} &= y_{t-1} = \boldsymbol{\phi_1'y} = [1,0,0,0]\boldsymbol{y} \\
z_{2,t} &= y_{t-2} = \boldsymbol{\phi_2'y} = [0,1,0,0]\boldsymbol{y} \\
z_{3,t} &= \frac{y_{t-1}+y_{t-2}+y_{t-3}}{3} = \boldsymbol{\phi_3'y} = \left[\frac{1}{3},\frac{1}{3},\frac{1}{3},0\right]\boldsymbol{y}
\end{aligned} \tag{2.9}$$

Acknowledged uncertainty about the threshold variable manifests lower convergence rates than the original 100% seen in Simulation 2. Based on 100 replications, convergence rates were 86%, 75%, and 87% for $z_{1,t}$, $z_{2,t}$, and $z_{3,t}$, respectively. For replications that converged, Figures 6-8 present posterior means of $\boldsymbol{\phi_k}$ for $k \in \{1, 2, 3\}$. Figure 2.9 shows almost perfect posterior weighting towards the true $z_{1,t} = y_{t-1}$ while Figure 2.10 provides evidence of occasional mis-identification of $z_{2,t} = y_{t-2}$. Figure 2.11 for $z_{3,t}$ shows almost equal favor for y_{t-1}, y_{t-2}, and y_{t-3} while severely down-weighting y_{t-4}.

Figure 2.9: Posterior Means of $\boldsymbol{\phi_1}$ When $z_{1,t} = y_{t-1}$

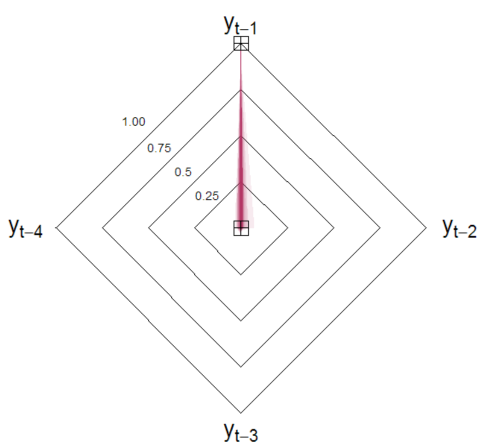


Figure 2.10: Posterior Means of $\boldsymbol{\phi_2}$ When $z_{2,t} = y_{t-2}$

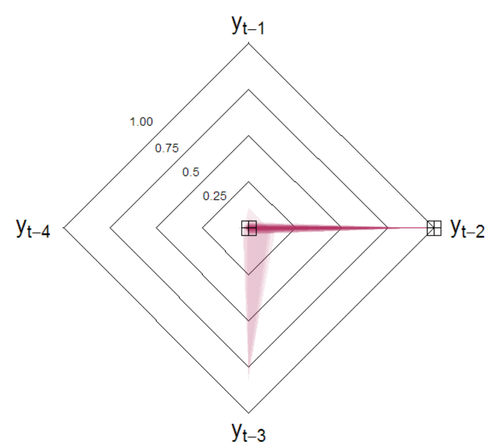
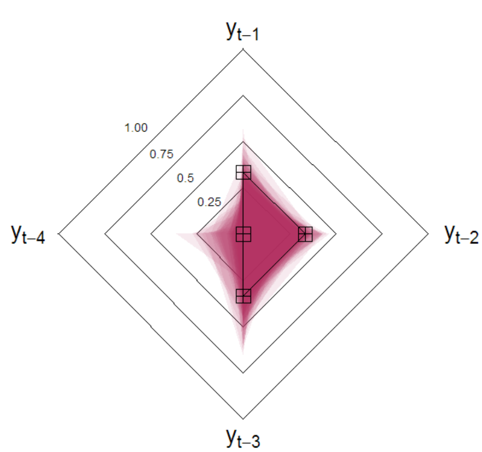


Figure 2.11: Posterior Means of $\boldsymbol{\phi_3}$ When $z_{3,t} = \frac{y_{t-1}+y_{t-2}+y_{t-3}}{3}$



2.4 Forecasting Annual Sunspot Numbers

International sunspot numbers are gathered and updated by the World Data Center SILSO, Royal Observatory of Belgium, Brussels. Since Granger (1957) , this data has served as an example in nonlinear time series literature. Letting x_t represent the annual sunspot number at time t, the square root transformation $y_t = 2[\sqrt{1+x_t} - 1]$ following Ghaddar and Tong (1981) is applied. Data from 1700 to 1979 are used to estimate models while data from 1980 to 2006 are used to evaluate their forecasting accuracy. Teräsvirta *et al.* (2010) compares three nonlinear time series models, namely STAR, TAR, and Artificial Neural Nets (AR-NN), to the baseline linear AR model. The LSTAR model in Equation 2.10 had optimal h-step ahead forecasting performance for horizons $h \in \{1, 2, \cdots, 5\}$. Sparsity is achieved through a stepwise frequentist procedure; henceforth, this model abbreviates to F_T.

$$\begin{aligned} y_t = &(1.46y_{t-1} - 0.76y_{t-2} + 0.17y_{t-7} + 0.11y_{t-9})[1 - G(y_{t-2}, 5.5, 7.9)] \\ &+ (2.7 + 0.92y_{t-1} - 0.01y_{t-2} - 0.47y_{t-3} + 0.32y_{t-4} - 0.26y_{t-5} \\ &+ 0.17y_{t-7} - 0.24y_{t-8} + 0.11y_{t-9} + 0.17y_{t-10})G(y_{t-2}, 5.5, 7.9) + \hat{\epsilon}_t \end{aligned} \tag{2.10}$$

where: $\hat{\epsilon}_t \sim N(0, 1.898^2)$.

Simulation results indicate the difficulty of normal priors in combating over-fitting: even if RJMCMC directed to the correct model order $p = 10$, current Bayesian approaches are incapable of estimating the model in Equation 2.10. Assuming the delay $d = 2$, a fully saturated LSTAR(10) model, denoted F_S, is estimated for a baseline comparison.

Hypothesis testing for the threshold variable produced ambiguous results as nonlinearity was rejected for multiple delay parameters. Teräsvirta *et al.* (2010) chose $d = 2$ based on p-value magnitude, but recommended LSTAR modeling for other values of d. Assuming $p = 10$ and $d = 2$, BHS priors estimate the LSTAR model, denoted B_2, in Equation 2.11. Posterior standard deviations are provided below the corresponding regime-specific AR coefficients. Parameter estimates for α_6 and β_6 round to zero and are ignored from the model representation.

$$
\begin{aligned}
y_t = (&\underset{(1.14)}{-0.56} + \underset{(0.20)}{1.56} y_{t-1} - \underset{(0.29)}{0.52} y_{t-2} + \underset{(0.15)}{0.01} y_{t-3} - \underset{(0.14)}{0.06} y_{t-4} \\
&- \underset{(0.12)}{0.03} y_{t-5} + \underset{(0.17)}{0.18} y_{t-7} + \underset{(0.13)}{0.05} y_{t-8} + \underset{(0.15)}{0.14} y_{t-9} - \underset{(0.10)}{0.04} y_{t-10}) \\
&\times [1 - G(y_{t-2}, 5.21, 8.37)] + (\underset{(1.11)}{0.43} + \underset{(0.25)}{0.83} y_{t-1} + \underset{(0.28)}{0.14} y_{t-2} \\
&- \underset{(0.14)}{0.24} y_{t-3} + \underset{(0.11)}{0.06} y_{t-4} - \underset{(0.10)}{0.1} y_{t-5} + \underset{(0.09)}{0.04} y_{t-7} - \underset{(0.14)}{0.13} y_{t-8} \\
&+ \underset{(0.11)}{0.06} y_{t-9} + \underset{(0.10)}{0.14} y_{t-10})[G(y_{t-2}, 5.21, 8.37)] + \hat{\epsilon}_t
\end{aligned}
\tag{2.11}
$$

where: $\hat{\epsilon}_t \sim N(0, 1.94^2)$

Along with BHS priors, $\boldsymbol{\phi} \sim Dir([\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}]')$ for Bayesian estimation of the threshold variable $z_t = \boldsymbol{\phi}'\boldsymbol{y}$. Results from Teräsvirta *et al.* (2010) indicate that $z_t \in \{y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4}\}$ is likely; therefore in the re-parameterization, let $\boldsymbol{y} = [y_{t-1}, y_{t-2}, y_{t-3}, y_{t-4}]'$. The estimated model in Equation 2.12 provides conflicting results for z_t; this model is denoted B_D. Posterior mean of $\boldsymbol{\phi}$ places more weight on y_{t-3} than the assumed threshold variable y_{t-2}.

$$
\begin{aligned}
y_t = (&\underset{(0.26)}{-0.04} + \underset{(0.08)}{1.36} y_{t-1} - \underset{(0.12)}{0.55} y_{t-2} + \underset{(0.10)}{0.06} y_{t-3} - \underset{(0.08)}{0.01} y_{t-4} \\
&- \underset{(0.08)}{0.02} y_{t-5} - \underset{(0.09)}{0.02} y_{t-6} + \underset{(0.12)}{0.16} y_{t-7} + \underset{(0.09)}{0.03} y_{t-8} + \underset{(0.08)}{0.06} y_{t-9} \\
&+ \underset{(0.05)}{0.02} y_{t-10})[1 - G(z_t, 31.87, 9.66)] + (\underset{(0.48)}{0.18} + \underset{(0.08)}{0.91} y_{t-1} \\
&+ \underset{(0.08)}{0.02} y_{t-2} - \underset{(0.08)}{0.07} y_{t-3} - \underset{(0.08)}{0.11} y_{t-5} + \underset{(0.06)}{0.01} y_{t-6} + \underset{(0.08)}{0.06} y_{t-7} \\
&- \underset{(0.09)}{0.07} y_{t-8} + \underset{(0.08)}{0.04} y_{t-9} + \underset{(0.06)}{0.06} y_{t-10})[G(z_t, 31.87, 9.66)] + \hat{\epsilon}_t
\end{aligned}
\tag{2.12}
$$

where: $z_t = [0.16, 0.15, 0.62, 0.08]\boldsymbol{y}$ and $\hat{\epsilon}_t \sim N(0, 1.88^2)$

Using the *Dirichlet* prior not only allows a compositional threshold variable, but can be used to shorten the list of possible lags. Under the assumption $d = 3$, the estimated model (B_3) is seen in Equation 2.13.

$$
\begin{aligned}
y_t = (&\underset{(0.22)}{0.03} + \underset{(0.07)}{1.32}y_{t-1} - \underset{(0.11)}{0.6}\,y_{t-2} + \underset{(0.1)}{0.08}y_{t-3} + \underset{(0.09)}{0.06}y_{t-4} \\
&- \underset{(0.08)}{0.02}y_{t-5} - \underset{(0.09)}{0.04}y_{t-6} + \underset{(0.11)}{0.1}\,y_{t-7} + \underset{(0.09)}{0.05}y_{t-8} + \underset{(0.09)}{0.09}y_{t-9} \\
&+ \underset{(0.05)}{0.03}y_{t-10})[1 - G(y_{t-3}, 32.42, 10.6)] + (\underset{(0.39)}{0.1} + \underset{(0.08)}{0.87}y_{t-1} \\
&+ \underset{(0.08)}{0.03}y_{t-2} - \underset{(0.06)}{0.02}y_{t-3} + \underset{(0.07)}{0.01}y_{t-4} - \underset{(0.09)}{0.13}y_{t-5} - \underset{(0.06)}{0.01}y_{t-6} \\
&+ \underset{(0.07)}{0.04}y_{t-7} - \underset{(0.08)}{0.05}y_{t-8} + \underset{(0.06)}{0.02}y_{t-9} + \underset{(0.05)}{0.04}y_{t-10}) \\
&\times [G(y_{t-3}, 32.42, 10.6)] + \hat{\epsilon}_t
\end{aligned}
\tag{2.13}
$$

where: $\hat{\epsilon}_t \sim N(0, 1.90^2)$

The posterior standard deviations and means of autoregressive coefficients in models B_2, B_D, and B_3 suggest simpler LSTAR models than Terasvirta's in Equation 2.10. Even simpler is the linear AR(10) model in Equation 2.14, also estimated using BHS priors. Evidence of nonlinearity does not always guarantee that nonlinear specifications will outperform linear ones in forecasting accuracy(Montgomery *et al.*, 1998; Tersvirta, 2005); thus the linear AR(10) model, denoted B_L serves as a benchmark in the evaluation.

$$
\begin{aligned}
y_t = &\underset{(0.64)}{0.83} + \underset{(0.06)}{1.22}y_{t-1} - \underset{(0.09)}{0.48}y_{t-2} - \underset{(0.09)}{0.08}y_{t-3} + \underset{(0.11)}{0.13}y_{t-4} \\
&- \underset{(0.09)}{0.1}\,y_{t-5} + \underset{(0.08)}{0.07}y_{t-7} - \underset{(0.09)}{0.07}y_{t-8} + \underset{(0.09)}{0.21}y_{t-9} + \underset{(0.06)}{0.03}y_{t-10}
\end{aligned}
\tag{2.14}
$$

where: $\hat{\epsilon}_t \sim N(0, 2.08^2)$

Consistent with Terasvirta's textbook (2010), the evaluation is based on h-step ahead root mean squared forecast error ($RMSFE(h)$) for horizons $h \in \{1, 2, \cdots, 5\}$. Out-of-sample forecasts are obtained recursively using a rolling window without re-estimation. One-step ahead forecasts are directly obtainable. The nonlinear nature of LSTAR requires Monte Carlo sampling of the theoretical error distribution (Peguin-Feissolle,

1994) or bootstrap sampling of the empirical error distribution for multi-step ahead forecasts (van Dijk *et al.*, 2002; Lundbergh and Teräsvirta, 2002). Robustness against distributional assumptions tilts favor toward bootstrapped forecasts (Lin and Granger, 1994).

Table 2.8 compares $RMSFE(h)$ of the two frequentist and four Bayesian estimated models. Models, F_T and B_2, estimated under assumption $d = 2$, perform clearly better than B_D and B_3. This contradicts the evidence for $d = 3$ seen in the training period. Efficacy of BHS shrinkage is illustrated through this extensively studied data: the best models, highlighted in bold, have almost identical forecasting performance for horizons 1 and 2, but B_2 starts outperforming at $h = 3$.

Table 2.8: $RMSFE(h)$ for Horizons $h \in \{1, 2, \cdots, 5\}$

Model	Horizon				
	1	2	3	4	5
F_T	**1.42**	**2**	**2.36**	**2.51**	**2.35**
F_S	1.86	3.21	3.7	3.63	3.16
B_L	1.73	2.3	2.54	2.53	2.56
B_2	**1.42**	**1.96**	**2.29**	**2.19**	**2.19**
B_D	1.77	2.83	3.38	3.5	3.29
B_3	1.86	3.11	3.58	3.62	3.58

2.5 Forecasting Daily Maximum Stream Water Temperatures

2.5.1 Background

Climate change has been proven to have a negative effect on cold water species. As habitats become less suitable, the natural biodiversity in streams is altered. In a study of salmonid population in a mountain river network, rainbow trout migrated toward higher, colder elevations, while the bull trout significantly adjusted as the percent of the network suitable for habitation declined tremendously from 1993 to 2006 (Isaak *et al.*, 2010). Furthermore, many nonnative invasive species inclined to warm water areas are infiltrating previously uninhabitable areas (Rahel and Olden, 2008). These distributional changes in streams alter localized food chains and thereby the entire ecosystem (Albouy *et al.*, 2014). Letting $T_w(t)$ and $T_a(t)$ represent daily maximum water and air temperatures on day t, predictive models assist environmental authorities in assessing when water temperatures are expected to exceed certain species-specific thresholds.

Mohseni *et al.* (1998) exploited the S-curve shaped association between water and air temperatures using the nonlinear logistic model seen in Equation 2.15. The lower asymptote β_0 represents the theoretical $\min T_w(t)$ and β_1 represents the theoretical range $\max T_w(t) - \min T_w(t)$. Parameters β_2 and β_3 control how fast water temperatures react to air temperature changes. The error term E_t represents the deviation from the equilibrium profile at time t.

$$T_w(t) = \beta_0 + \frac{\beta_1}{1 + \exp[\beta_2 - \beta_3 T_a(t)]} + E_t \tag{2.15}$$

Caissie *et al.* (1998) employed harmonic regression models using Fourier series, to capture the annual cycles natural to water and air temperatures. The seasonality of daily maximum water and air temperatures is sufficiently captured by the first harmonic as seen in Equations 2.16 and 2.17; the error terms W_t and A_t represent the deviations from seasonal maximum water and air temperature profiles at time t, respectively.

$$T_w(t) = \beta_0 + \beta_1 \sin\left(\frac{2\pi t}{365.25}\right) + \beta_2 \cos\left(\frac{2\pi t}{365.25}\right) + W_t \tag{2.16}$$

$$T_a(t) = \beta_0 + \beta_1 \sin\left(\frac{2\pi t}{365.25}\right) + \beta_2 \cos\left(\frac{2\pi t}{365.25}\right) + A_t \tag{2.17}$$

The three river-specific profiles are estimated using historical data, and deviations are calculated. Instead of forecasting daily maximum water temperatures directly, models are designed to forecast deviations from the seasonal water temperature profiles. Let $\boldsymbol{w_t} = [W_t, W_{t-1}, \cdots, W_{t-p_W}]'$, $\boldsymbol{a_t} = [A_t, A_{t-1}, \cdots, A_{t-p_A}]'$, and $\boldsymbol{e_t} = [E_t, E_{t-1}, \cdots, E_{t-p_E}]'$. Most commonly, subsets of the linear model seen in Equation 2.18 are employed in the literature (Benyahya *et al.*, 2007; Caissie *et al.*, 2001).

$$W_{t+1} = \mu + \boldsymbol{w_t'\alpha} + \boldsymbol{a_t'\beta} + \boldsymbol{e_t'\theta} + \epsilon_t \tag{2.18}$$

Previous research focused on forecasting 1-step ahead where subsets of the previous model perform competitively. Our interest is on 3-step and 7-step ahead forecasts; exploited nonlinearity may improve performance at longer horizons. The two exogenous time series, A_t and E_t, complicate the multi-step ahead forecast of W_{t+h}, which not only depends on future unknown values $\{W_{t+1}, W_{t+2}, \cdots, W_{t+h-1}\}$ but also on both $\{A_{t+1}, A_{t+2}, \cdots, A_{t+h-1}\}$ and $\{E_{t+1}, E_{t+2}, \cdots, E_{t+h-1}\}$. The remedy is horizon-specific models where forecasting W_{t+h}, requires information at or before time t. For the basic LSTAR(p) model, the iterative (Monte Carlo or bootstrap) approaches were shown to forecast better than this more direct approach on average (Lin and Granger, 1994). Nevertheless, computational advantages outweigh forecasting disadvantages, and horizon specific nonlinear models are seen throughout literature (Stock and Watson, 1998; Marcellino *et al.*, 2006).

Consider the three following horizon-specific models: Linear, shown in Equation 2.19, nonlinear LSTAR with fixed threshold variable, depicted in Equation 2.20, and nonlinear LSTAR with unknown threshold variable delineated in Equation 2.21. Given horizon h, the aforementioned specifications are respectively denoted $L(h)$, $N_1(h)$, and $N_2(h)$. Each model is developed from the same information and depends on the three model orders p_W, p_A, and p_E. Assumptions about the order parameters, such as $p_W = p_A = p_E = p$, simplify the MCMC sampling algorithm at a loss of model flexibility.

$$W_{t+h} = \mu + \boldsymbol{w_t'\alpha} + \boldsymbol{a_t'\beta} + \boldsymbol{e_t'\theta} + \epsilon_t \tag{2.19}$$

$$\begin{aligned} W_{t+h} = {} & (\mu_1 + \boldsymbol{w_t'\alpha_1} + \boldsymbol{a_t'\beta_1} + \boldsymbol{e_t'\theta_1})[1 - G(W_t, \gamma, \delta)] \\ & + (\mu_2 + \boldsymbol{w_t'\alpha_2} + \boldsymbol{a_t'\beta_2} + \boldsymbol{e_t'\theta_2})[G(W_t, \gamma, \delta)] + \epsilon_t \end{aligned} \tag{2.20}$$

$$\begin{aligned} W_{t+h} = {} & (\mu_1 + \boldsymbol{w_t'\alpha_1} + \boldsymbol{a_t'\beta_1} + \boldsymbol{e_t'\theta_1})[1 - G(z_t, \gamma, \delta)] \\ & + (\mu_2 + \boldsymbol{w_t'\alpha_2} + \boldsymbol{a_t'\beta_2} + \boldsymbol{e_t'\theta_2})[G(z_t, \gamma, \delta)] + \epsilon_t \\ & \text{where: } z_t = \boldsymbol{\phi' w_t} \end{aligned} \tag{2.21}$$

2.5.2 The Data

Four years of daily maximum water temperatures and maximum air temperatures were collected from 31 rivers in Spain. The Spanish Environmental Department is credited for the water temperatures and the Spanish Meteorological Agency is credited for the air temperatures. Pairs of measurement stations were chosen for each river under strict guidelines to limit the impact from dams, cities, and fuel/nuclear power stations. The full data set is not limited to just daily maximum water and air temperatures; for further information, see Kamarianakis *et al.* (2016).

For each river, the four years of data could come from any of the years between 2000 and 2008, inclusive. Typically the four years are often not consecutive and 15% of the data is missing across all the rivers. To evaluate forecasting performance of river-specific linear and nonlinear models, the four years are split into a training and testing set. The training set contains the two, ideally consecutive, years of data with the least amount of missing observations. The two remaining years in the testing set typically are not adjacent and not immediately preceding the training period.

2.5.3 Results

For all 31 rivers, BHS priors result in sparse estimation of river specific models $L(h)$, $N_1(h)$, and $N_2(h)$ for $h \in \{3, 7\}$. The maximum complexity considered is constrained by the assumption that $p = \max\{p_W, p_A, p_E\} = 6$. Table 2.9 shows the percent of times the six models converged across the 31 rivers. Forecasting results from models where convergence was not reached after 20 updates are ignored. Models designed for horizon h are evaluated based on their corresponding $RMSFE(h)$.

Table 2.9: Percentages of River-Specific Models that Achieved Convergence

	Horizon	
Model	3	7
$L(h)$	100%	97%
$N_1(h)$	97%	100%
$N_2(h)$	70%	81%

Horizon specific linear models $L(3)$ and $L(7)$ outperform nonlinear alternatives for 67% and 55% of the rivers, respectively. When the nonlinear models outperform, the advantage is often marginal and insignificant. In these cases, the simpler linear specifications are more practical and therefore recommended. Now the focus is on three scenarios where the improvement from the nonlinear model was unusual relative to the rest of the rivers.

For Guadiana River, model $N_2(7)$ reduced overall RMSFE(7) by 0.098 ^oC. Based on posterior weights 0.423 and 0.301, the threshold variable is approximately an average of W_t and W_{t-6}, respectively and the estimated threshold is 0.06 ^oC. Figures 2.12-2.13 show the posterior means of regime-specific autoregressive coefficients. In both regimes, the majority of information needed to forecast W_{t+7} comes from the known seasonal deviation W_t; this phenomenon is stronger in the high regime.

Figure 2.12: Low Regime Coefficients for Guadiana from $N_1(7)$

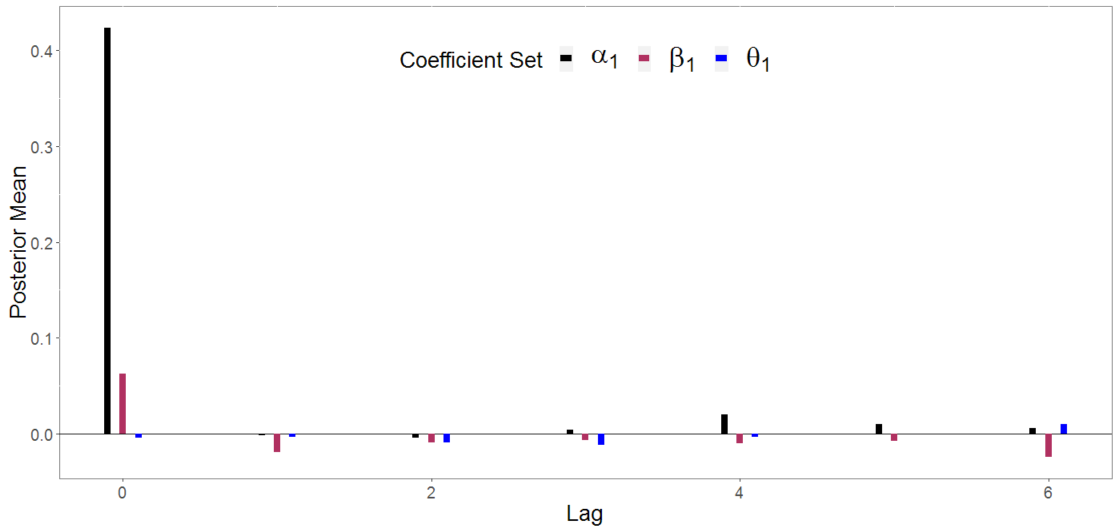
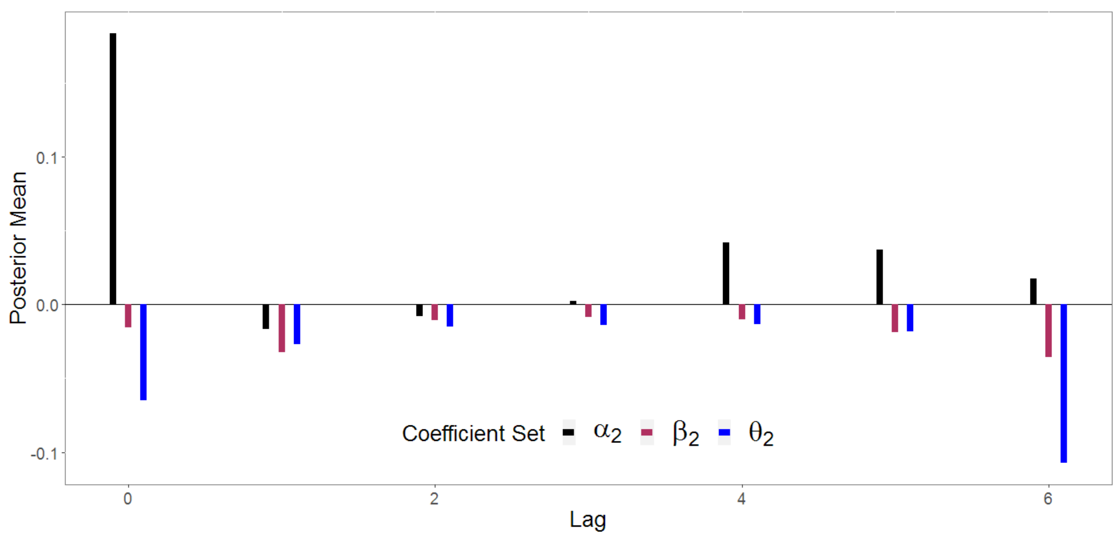


Figure 2.13: High Regime Coefficients for Guadiana from $N_1(7)$



The next two examples involve the Jarama River where nonlinear models provided superior performance. For $h = 3$, $N_2(3)$ reduced $RMSFE(3)$ by 0.165 ^oC; and for $h = 7$, $N_1(7)$ reduced $RMSFE(7)$ by 0.105 ^oC. Posterior expectations of $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\theta}$ for $N_2(3)$ are shown in Figures 2.14-2.15 and for $N_1(7)$ are depicted in Figures 2.16-2.17. For Jarama, it is intriguing that the optimal nonlinear model differs for the two horizons. The threshold variable in $N_2(3)$ places its largest weight of 0.66 on W_{t-6}, representing information one week prior. These nonlinear models change dynamics around different thresholds: for $N_2(3)$, regime switching occurs when maximum water temperature at time t surpasses its seasonal average at time t by 1.20 ^oC; and for $N_1(7)$, this change occurs for 0.65 ^oC. The nonlinear dynamics exhibited in the low and high regimes also change with the horizon h. When forecasting W_{t+3}, the realization W_t provides the most information when in the low regime; but in the high regime, none of the known information up to time t is helpful. The model for forecasting W_{t+7} is even more interesting since the AR dynamics in both regimes are similar to the high regime of $N_2(3)$. Knowing information at time t, specifically W_t, is only helpful in determining when to jump between means $\mu_L = 0.002$ and $\mu_H = 0.03$ to forecast 7-steps ahead. Obvious differences between these two horizon-specific models illustrate that for longer horizons, currently known data provide less useful information in forecasting.

2.6 Conclusion

Bayesian shrinkage priors for the AR coefficients and the *Dirichlet* prior for a composite threshold variable are employed to estimate a flexible specification that nests the classic LSTAR model. Although simulation experiments show *Dirichlet* priors to be an adequate alternative for estimating composite threshold variables, improved forecasting performance is not guaranteed. An advantage of the proposed methods is that practitioners can immediately apply them using common statistical software. Detailed code is provided with this paper for a tutorial in using Bayesian horseshoe to estimate nonlinear LSTAR models.

Recent alternatives to BLASSO and BHS based on the *double-Pareto* (Armagan *et al.*, 2013) and *Dirichlet-Laplace* priors (Bhattacharya *et al.*, 2015) may also be employed to estimate regime-specific autoregressive terms. Besides the jump from a

Figure 2.14: Low Regime Coefficients for Jarama from $N_1(3)$

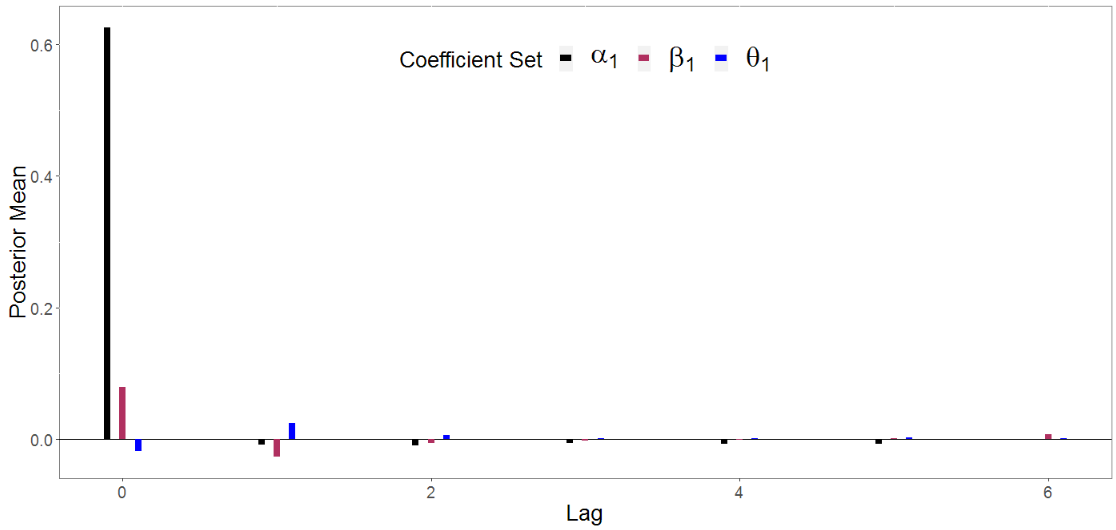
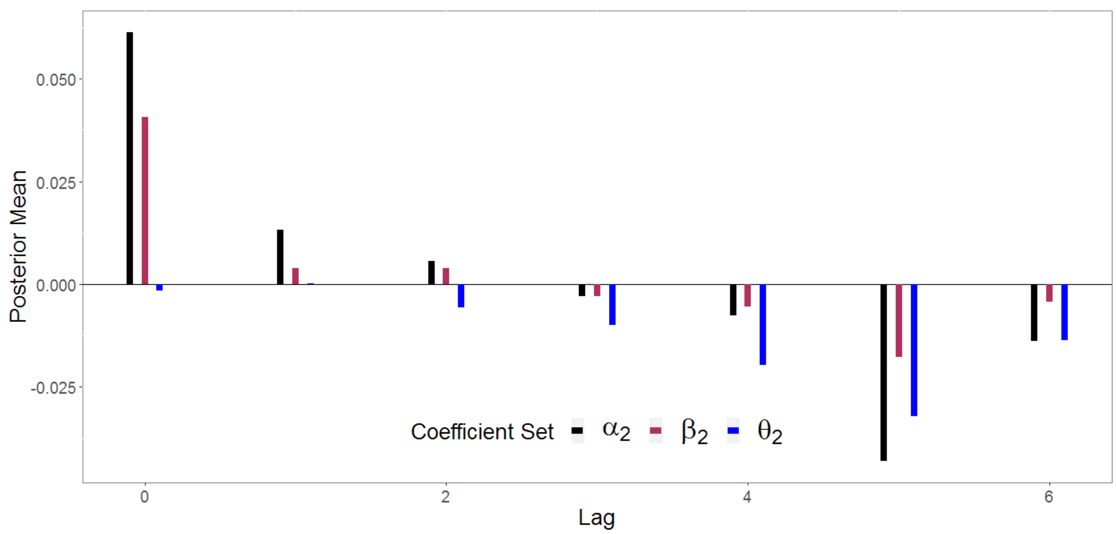


Figure 2.15: High Regime Coefficients for Jarama from $N_1(3)$



linear to nonlinear LSTAR, which more than doubles the number of estimated parameters, increasing the assumed autoregressive orders in the regimes and the compositional threshold variable, further expands the parameter space causing slower convergence. For these reasons, a modified horseshoe representation is recommended for extremely sparse signals (Bhadra *et al.*, 2016).

The proposed methods can be easily employed to estimate nonlinear models rep-

Figure 2.16: Low Regime Coefficients for Jarama from $N_1(7)$

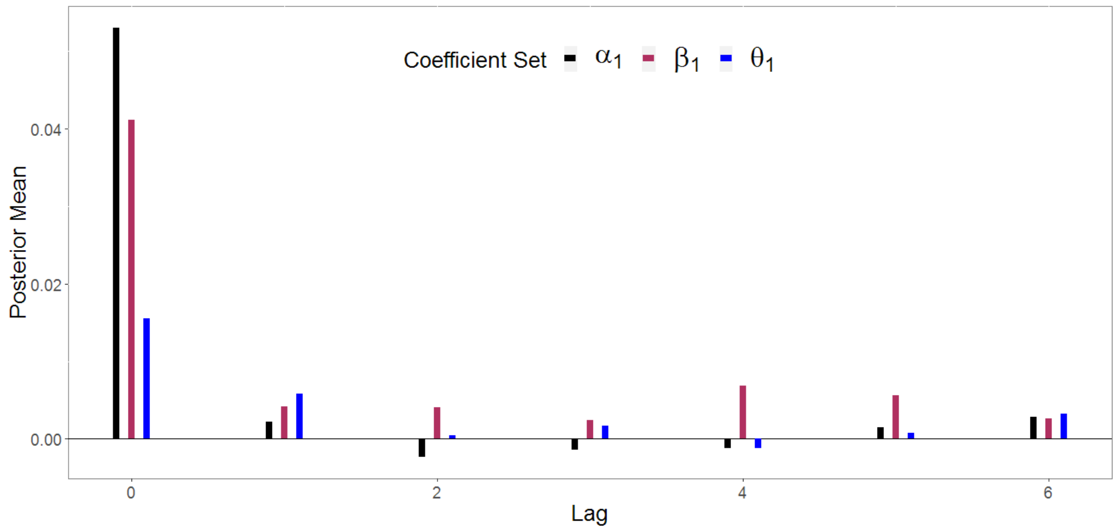
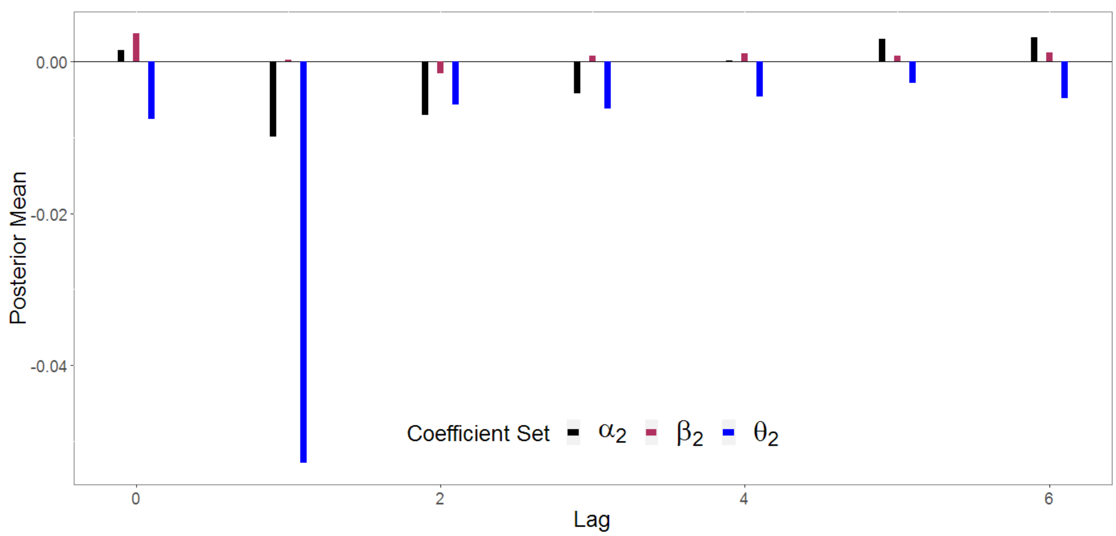


Figure 2.17: High Regime Coefficients for Jarama from $N_1(7)$



resented by Equation 2.1 such as ESTAR and TAR. Future work involves applying and evaluating these methods on multiple regime smooth transition autoregressions (MR-STAR) where the number of unknown parameters may increase dramatically. For MR-STAR, Bayesian shrinkage may be used to circumvent the necessity for nested RJMCMC routines for each regime, or restrictive prior assumptions.

Sample code for this paper is provided in Appendix A. Code provides tutorial for implementing Bayesian regime-specific shrinkage estimation for nonlinear LSTAR models. Examples are provided in terms of a simulation study and an applied situation involving the monthly international sunspot numbers. The simulation study involving the composite threshold variable is used to illustrate the applicability and ease of the *Dirichlet* prior. Code also pertaining to multistep ahead forecasts using the bootstrap method can prove to be useful in countless other situations outside the scope of this paper.

Chapter 3

BAYESIAN ESTIMATION OF SUBSET THRESHOLD AUTOREGRESSIVE MODELS FOR SHORT-TERM FORECASTING OF TRAFFIC OCCUPANCY

3.1 Introduction

Rising populations in major cities add stress to advanced traffic management systems (ATMS) tasked with monitoring real-time traffic variables to proactively reduce congestion. Ever since Ahmed and Cook (1979) used basic ARIMA strategies to model freeway traffic networks in large US cities – Los Angeles, Minneapolis, and Detroit – significant research has accumulated to appropriately utilize the massive amount of data obtained in transportation networks. The global concern has manifested through independent research in major cities in places such as the United Kingdom (Queen and Albers, 2009; Dunne and Ghosh, 2012), Greece (Stathopoulos and Karlaftis, 2003; Kamarianakis *et al.*, 2012; Theofilatos *et al.*, 2017), Italy (Annunziato *et al.*, 2013; Moretti *et al.*, 2015), China (Shang *et al.*, 2006; Jun and Jun, 2007; Min *et al.*, 2010), and Ethiopia (Hellendoorn *et al.*, 2011). Technological advances over this time period have not only improved the gathering of the data but also in the quick distribution of pertinent information to drivers. The speed and accuracy between the detection to the correction rely on efficient modeling and accurate short-term forecasting of important traffic characteristics.

Three traffic variables have been used to quantify traffic congestion per unit of time: flow (volume per time), speed (distance per time), and occupancy (percent of time occupied) Hall (1992). Smith and Demetsky (1997) provide insight into the state of traffic modeling 20 years ago while Vlahogianni *et al.* (2014) do an excellent job summarizing recent advancements in short-term traffic forecasting by posing 10 interesting challenges for future researchers. The scarcity of forecast procedures for traffic occupancy may stem from the instability acknowledged in Levin and Tsao (1980). Traffic occupancy, the percent of time a detection zone is occupied, has been described as "quality assessment measure" as it quantifies how well traffic is moving through a network (Klein and Kelley, 1996). Univariate approaches for modeling traffic occupancy are presented for the terminal goal of evaluating forecasts at multiple horizons. Section 3.2 presents a challenging traffic dataset from a major arterial in Athens, Greece, used for empirical study.

Traffic occupancy has been used to help forecast other traffic characteristics (Hazelton, 2004). In regards to modeling traffic occupancy, most researchers adapt similar methods seen for traffic flow and speed (Kamarianakis *et al.*, 2010). Like other traffic variables, occupancy exhibits abrupt changes in mean, temporal dynamics, and volatility as traffic fluctuates between free flow and congested states. Realizations of recent traffic occupancy can assist in the characterization of these states. In Section 3.3, nonlinear threshold autoregressive (TAR) processes model and forecast traffic occupancy. The parametric TAR structure, first discussed in Tong (1990), is a conditional autoregressive (AR) model dependent on states governed by traffic occupancy . The model is highly interpretable making it appealing to practitioners. For easy application, TAR models for each location are defined to be day-specific and horizon-specific. Also, a periodic linear regression model that adequately captures the seasonality exhibited in traffic data is used to produce baseline forecasts.

Ghosh *et al.* (2007) provide a case for the movement from classical inference to Bayesian inference in traffic models. Joint contributions from Broemeling and Cook (1992); Geweke and Terui (1993); Chen and Lee (1995) formed the foundation of Bayesian TAR modeling. Campbell (2004) applied reversible jump Markov chain

Monte Carlo to select regime-specific AR orders. Subset selection of TAR via stochastic search variable selection (George and McCulloch, 1993) was conducted by So and Chen (2003); Chen and Chan (2011). For all the aforementioned approaches, the number of regimes must be known or assumed. In illustration, simulation, and application, TAR models are often restricted to have at most three regimes.

Chan *et al.* (2015) transformed the nonlinear TAR model into a high dimensional linear regression. Multi-step procedures seek sparse solutions to identify the regimes and perform parameter estimation (Chan *et al.*, 2015, 2017). The fully Bayesian approach of Pan *et al.* (2017) operates similarly by utilizing a sequence of binary inclusion variables to identify change points and select regimes. The fully saturated TAR model defined Section 3.3 follows from Chan *et al.* (2015) with some slight modifications.

Section 3.4 proposes a fully Bayesian three step procedure that automates selecting the regimes and sparse subset AR estimation within regimes. First, a fully saturated TAR model is estimated using Bayesian regularization implemented through a modified horseshoe prior (Carvalho *et al.*, 2009, 2010; Bhadra *et al.*, 2016). Next, the Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951) combined with a forward selection algorithm identifies the regimes through comparing posterior predictive distributions of the linear AR model to multiple regime TAR models. Finally, given a restricted set of regimes, the same procedure is repeated to select the relevant dynamics within each of the regimes. The result is a parsimonious TAR model with a posterior predictive distribution close in KL distance to the posterior predictive distribution from the full model.

Section 3.5 provides empirical results of out-of-sample forecasting results from final TAR models with potentially many regimes. Baseline periodic seasonal regressions are used to produce baseline forecasts. Models are compared using the mean absolute scaled measure of forecast accuracy (MASFE) of Hyndman and Koehler (2006). By scaling forecast errors by the mean absolute error from a horizon-specific naive random walk, models can be simultaneously be compared to each other and the naive method.

3.2 Data

Real-time traffic data are obtained from the major Athen's arterial, Alexandras Avenue, along the westbound direction. Every 90 seconds in April 2000, traffic occupancy is captured from seven loop detectors abbreviated $L \in \{A, B, C, D, E, F, G\}$. The National Technical University of Athens is credited for the gathering of this data. The 2013 Traffic Research Board's (TRB) Annual Meeting Workshop used a larger ecompassing dataset in their TRANSportation Data FORecasting Competition (TRANSFOR). This competition was organized by the Aritificial Intelligence and Advanced Computing Applications Committee. Many of the inherent characteristics in this dataset make short-term forecasting quite challenging. The winning methodology applied adaptive lasso to high-dimensional nonlinear space-time models (Kamarianakis *et al.*, 2012). Figure 3.1 was created using Google Maps to provide a visual depiction of this small network with arrows representing direction of traffic flow. Loop detectors A, B, C, and D measure traffic occupancy in the westbound direction, and detectors E, F, and G, in the eastbound direction.

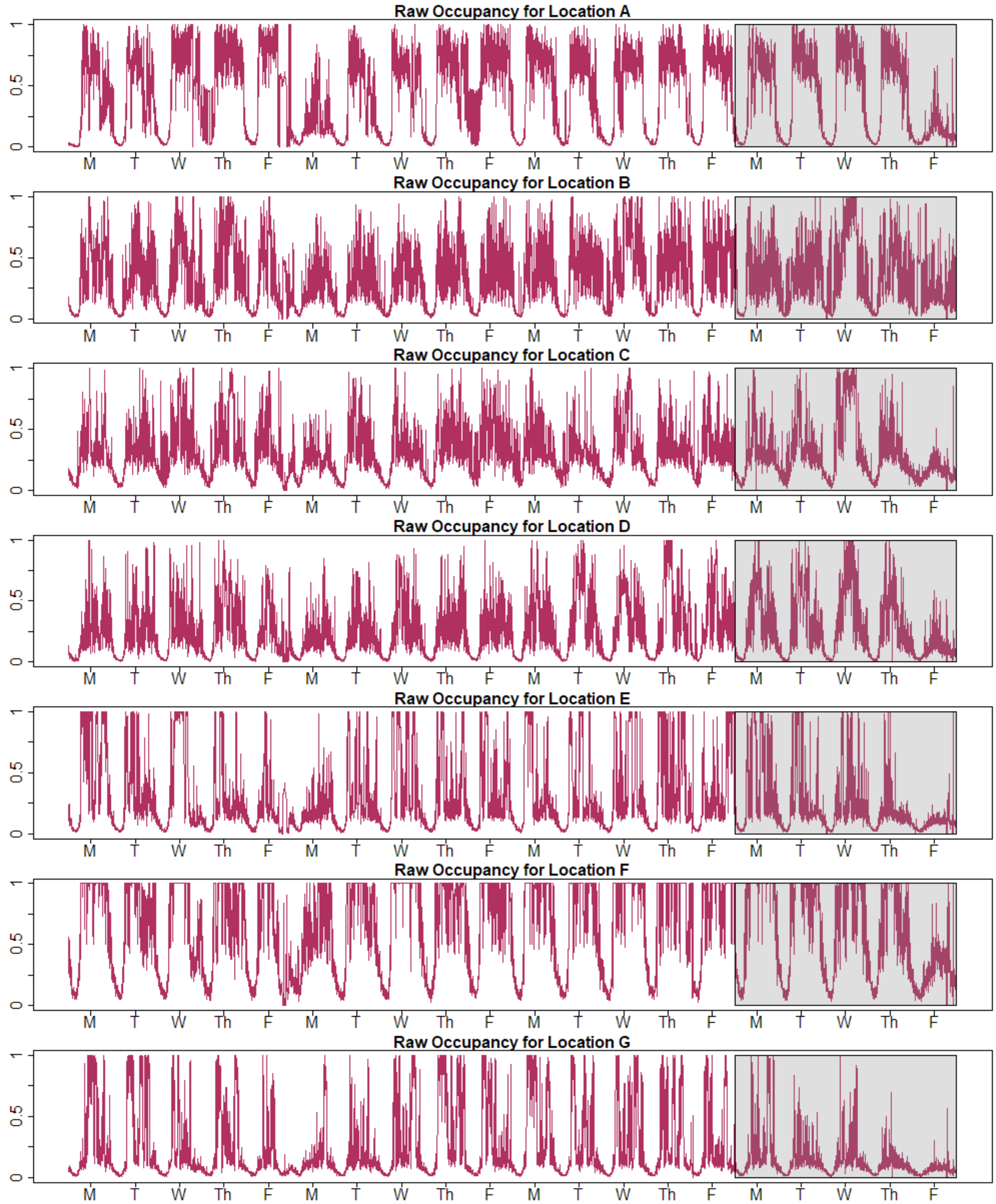
Analyzing the raw traffic occupancy from an urban network measured on the 90s interval becomes problematic due to the large amount of noise seen at high resolutions (Vlahogianni *et al.*, 2014). Temporal aggregation to larger intervals i.e. 15 min has been practiced over the years as a smoothing technique prior to modeling. Not only does this practice make short-term forecasting irrelevant with today's technology but diminishes useful long memory, nonlinear, and heteroskedastic dynamics in the underlying signal Vlahogianni and Karlaftis (2011).Rather than modeling across dif-

Figure 3.1: Locations of Loop Detectors Along Alexandras Ave. in Athens, Greece: Arrows Indicate Direction of Traffic Flow

ferent levels of temporal aggregation, as seen in Shang *et al.* (2006), traffic occupancy is averaged to 3 minute intervals resulting in 480 daily time points per location.

Define random variable $O_{L,t}$ as the traffic occupancy for location L at time t and $o_{L,t}$ represents a known realization. As common practice, weekend data is ignored. Cyclical human behavior patterns throughout the work week lead to weekly seasonal traffic patterns. Data during April 2000 covers four complete weeks. The first three weeks are used to fit TAR models, and the last week is designated for forecasting evaluation. Time series plots of $\{o_{L,t}\}$ are found in Figure 3.2 organized by location.

Figure 3.2: Raw Traffic Occupancy for All Locations Measured on 3-Minute Interval: Shaded Region Indicates the Forecasting Period



3.3 Threshold Autoregressive Model

3.3.1 General Modeling Information

For each location $L \in \{A, B, C, D, E, F, G\}$, day of the week $D \in \{M, T, W, Th, F\}$, and horizon $h \in \{1, 3, 5\}$, (L, D, h)-specific TAR models are built to forecast $\widehat{O}_{L,t} = E[O_{L,t}|\mathcal{I}_t]$ where $\mathcal{I}_t = \{o_{L,k}\}_{k=t-h}^{t-h-P+1}$. Chosen horizons correspond to 3 min, 9 min, and 15 min ahead forecasts. The weekly periodicity of traffic occupancy modeled in Williams and Hoel (1999); Ghosh *et al.* (2007); Kamarianakis *et al.* (2010) and visually seen in Figure 3.2 defends D-specific modeling. The purpose of h-specific models is to ensure multi-step forecasting is user-friendly and computationally efficient for practitioners.

The order parameter $P \in \mathbb{N}$ represents the maximum short-term lag relevant for forecasting and should be chosen large enough to cover relevant temporal dependencies across all traffic states. The order $P = 7$ is fixed equating to the last 21 minutes of known information. To produce short-term forecasts, only short-term dynamics are considered. Long-term or seasonal dynamics can be included but require more periods to adequately estimate. Periodic regression models with Fourier terms adequately capture weekly seasonality and are used to produce baseline forecasts (Kamarianakis *et al.*, 2010). As $h \to \infty$, (L, D)-specific seasonal models are expected to dominate over (L, D, h)-specific TAR models.

3.3.2 Transformed Occupancy

Using function $\text{logit}(x) : (0, 1) \to \mathbb{R}$ such that $\phi(x) = \log[x/(1 - x)]$, define the new transformed variable $Y_{L,t} = \text{logit}(O_{L,t})$. Raw occupancy is bounded on the $[0, 1]$ interval. Recoding 0 with 0.0001 and 1 with 0.9999 is a nonevasive technique to handle extreme occupancies when logit(.) is undefined. All models are defined for the variable $Y_{L,t}$, an approach used for proportional time series since Wallis (1987). Figure 3.3 displays the transformed series $\{y_{L,t}\}$ for each location. Although forecasts are produced for the final week, evaluation of forecasts are considered on the original scale using $\text{logit}^{-1}(x) : \mathbb{R} \to (0,1)$ where $\text{logit}^{-1}(x) = \frac{\exp(x)}{1+\exp(x)}$. Since $\text{logit}(x)$ is a nonlinear transformation, the forecast $\hat{O}_{L,t} \neq \text{logit}^{-1}(\hat{Y}_{L,t})$. Unbiased forecasts and quantiles are produced from the set $\{\text{logit}^{-1}(\hat{Y}_{L,t}^{(s)})\}_{s=1}^{S}$ where $\{\hat{Y}_{L,t}^{(s)}\}_{s=1}^{S}$ are S posterior samples obtained from the posterior predictive distribution $f(\hat{Y}_{L,t}|\mathcal{I}_t^*)$ where $\mathcal{I}_t^* = \{y_{L,k}\}_{k=t-h}^{t-h-P+1}$.

3.3.3 General (L, D, h)-Specific TAR Model

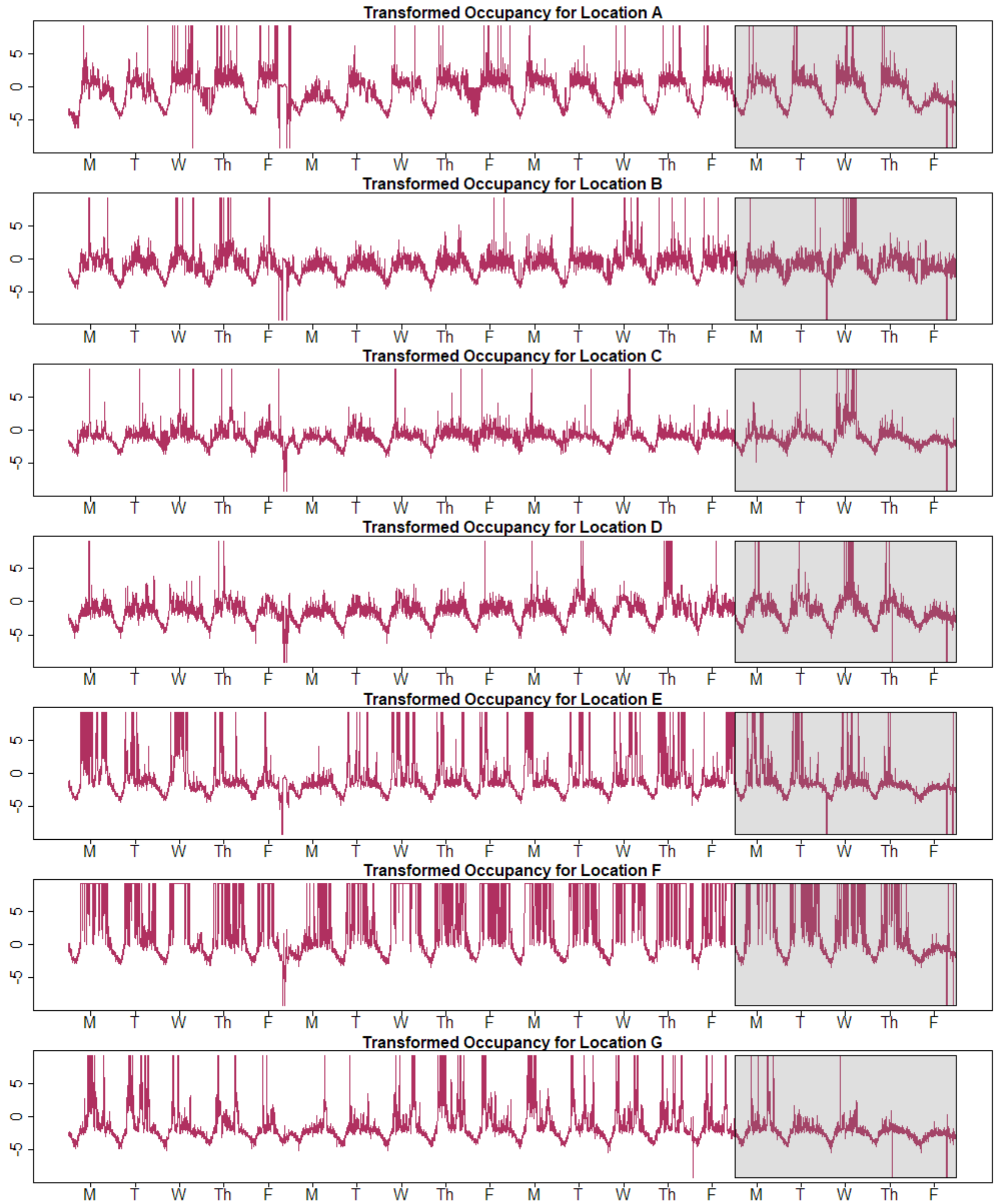
The random process $\{Y_t\}$ follows TAR model of order P with $m+1$ regimes if

$$Y_t = \phi_0^{(j)} + \sum_{i=1}^{P} \phi_i^{(j)} Y_{t-h-i+1} + \sigma\epsilon_t, \text{ for } \delta_{j-1} < Y_{t-h} \leq \delta_j, \tag{3.1}$$

where $\sigma > 0$, $j \in \{1, 2, \cdots, m+1\}$, and $h \in \mathbb{N}$. The vector of thresholds $\boldsymbol{\delta} = [\delta_1, \delta_2, \cdots, \delta_m]'$ divides the process into $m+1$ regimes where $-\infty = \delta_0 < \delta_1 \leq \delta_2 \leq \cdots \leq \delta_m < \delta_{m+1} = \infty$. Since the most recent realization Y_{t-h} determines the current model state, the TAR model is conventionally classified as "self-exciting" (Ghaddar and Tong, 1981). The sequence of errors $\{\epsilon_t\}$ are assumed to be i.i.d. with zero mean and unit variance.

The TAR structure in Equation 3.1 is slightly more rigid than the classic structure in Chen and Lee (1995). Rather than utilizing regime-specific variance parameters σ_j, homoskedasticity is assumed. When $\text{logit}^{-1}(x)$ is used to obtain density forecasts on the original $[0,1]$ scale, heteroskedasticity is naturally captured. This is analogous to *Beta* distributed random variables where the variance is dependent on the mean. Another key difference arises in the selection of the transition variable. Often a delay parameter d is introduced and Y_{t-d} drives regime changes. Following from Chan

Figure 3.3: Logit Transformed Traffic Occupancy for All Locations Measured on 3-Minute Interval: Shaded Region Indicates the Forecasting Period



et al. (2015), d is known and $d = h$ is fixed. Exogenous traffic variables nor time are considered for the transition variable.

3.3.4 High Dimensional Linear Representation

To reformulate Equation 3.1 into a high dimensional linear regression model, a slight deviation from the procedure in Chan *et al.* (2015, 2017) is outlined with similar notation for consistency. Suppose the discrete time series $\{y_t\}_{t=1-h-P+1}^{T}$ is observed. Let $\boldsymbol{y} = [y_1, \cdots, y_T]'$, $\boldsymbol{\epsilon} = [\epsilon_1, \cdots, \epsilon_T]'$, and define matrix $\boldsymbol{X}$ by

$$\boldsymbol{X} = \begin{bmatrix} 1 & y_{1-h} & y_{1-h-1} & \cdots & y_{1-h-P+1} \\ 1 & y_{2-h} & y_{2-h-1} & \cdots & y_{2-h-P+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & y_{T-h} & y_{T-h-1} & \cdots & y_{T-h-P+1} \end{bmatrix}.$$

The $T \times 1$ response vector $\boldsymbol{y}$, $T \times 1$ error vector $\boldsymbol{\epsilon}$, and $T \times (1 + P)$ model matrix $\boldsymbol{X}$ are often seen in matrix representations of h-specific AR(P) models.

The second column in $\boldsymbol{X}$ contains the sequence of h-specific transition variables. Define the sorting function $\pi(i) : \{1, \cdots, T\} \to \{1, \cdots, T\}$ where $\pi(i)$ equates to the time index of the ith smallest element in $[y_{1-h}, y_{2-h}, \cdots, y_{T-h}]'$. The new $\boldsymbol{y}_R = [y_{\pi(1)+h}, \cdots, y_{\pi(T)+h}]'$, $\boldsymbol{\epsilon}_R = [\epsilon_{\pi(1)+h}, \cdots, \epsilon_{\pi(T)+h}]'$ and

$$\boldsymbol{X}_1 = \begin{bmatrix} 1 & y_{\pi(1)} & y_{\pi(1)-1} & \cdots & y_{\pi(1)-P+1} \\ 1 & y_{\pi(2)} & y_{\pi(2)-1} & \cdots & y_{\pi(2)-P+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & y_{\pi(T)} & y_{\pi(T)-1} & \cdots & y_{\pi(T)-P+1} \end{bmatrix} = \begin{bmatrix} \boldsymbol{y}'_{\pi(1)} \\ \boldsymbol{y}'_{\pi(2)} \\ \vdots \\ \boldsymbol{y}'_{\pi(T)} \end{bmatrix}$$

are essentially $\boldsymbol{y}$, $\boldsymbol{\epsilon}$, and $\boldsymbol{X}$ sorted according to the order statistics of the transition variable. The reordered errors in $\boldsymbol{\epsilon}_R$ are assumed to be i.i.d. with mean 0 and variance σ^2.

In practical application, it makes sense to limit the TAR model to $m+1$ regimes requiring the estimation of m thresholds in the range of the transition variable. Let $q(.) : [0,1] \rightarrow [\min\{y_{t-h} : t = 1,2,\cdots,T\}, \max\{y_{t-h} : t = 1,2,\cdots,T\}]$ denote the sample quantile function and consider a sequence $\{p_k\}_{k=1}^{m}$ of m evenly spaced percentiles such that $p_{min} = p_1 < \cdots < p_m = p_{max}$. For a fully saturated TAR model limited to $(m+1)$ regimes, fix *a priori* the vector of thresholds $\boldsymbol{\delta} = [q(p_1), q(p_2), \cdots, q(p_m)]'$. For $j \in \{2, \cdots, m+1\}$, let k_j represent the number of elements in $[y_{1-h}, y_{2-h}, \cdots, y_{T-h}]'$ less than $q(p_{j-1})$ and define

$$\boldsymbol{X}_j = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \\ 1 & y_{\pi(k_j+1)} & y_{\pi(k_j+1)-1} & \cdots & y_{\pi(k_j+1)-P+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & y_{\pi(T)} & y_{\pi(T)-1} & \cdots & y_{\pi(T)-P+1} \end{bmatrix} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ \boldsymbol{y}'_{\pi(k_j+1)} \\ \vdots \\ \boldsymbol{y}'_{\pi(T)} \end{bmatrix}.$$

Finally, a slightly restricted version of the $(m+1)$-regime TAR process seen in Equation 3.1 can be expressed as a linear regression by

$$\boldsymbol{y}_R = \boldsymbol{X}_R \boldsymbol{\theta}_R + \boldsymbol{\epsilon}_R \tag{3.2}$$

where $\boldsymbol{X}_R = [\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_{m+1}]$ is a $T \times (P+1)(m+1)$ model matrix and $\boldsymbol{\theta}_R = [\boldsymbol{\theta}'_1, \boldsymbol{\theta}'_2, \cdots, \boldsymbol{\theta}'_{m+1}]'$ is a $(P+1)(m+1) \times 1$ vector of coefficients grouped by regime. From Equation 3.1, set $\boldsymbol{\phi}_j = [\phi_0^{(j)}, \phi_1^{(j)}, \cdots, \phi_P^{(j)}]'$. Starting with $\boldsymbol{\theta}_1 = \boldsymbol{\phi}_1$, the state dependent coefficient group $\boldsymbol{\theta}_j = \boldsymbol{\phi}_j - \boldsymbol{\phi}_{j-1}$ for $j \in \{2, \cdots, m+1\}$ represents the marginal adjustment in dynamics when y_{t-h} crosses the threshold $\delta_{j-1} = q(p_{j-1})$.

3.3.5 *Baseline Seasonal Model*

Seasonal models allow us to understand the long-run relationship of a variable over time through repetitive cycles. It is common practice in time series analysis to deseasonalize data prior to model buiding through smoothing or seasonal differencing. In many studies, seasonal autoregressive integrated moving average models (SARIMA) have been used to analyze traffic characteristics (Williams and Hoel, 2003; Ghosh *et al.*, 2005; Zhang *et al.*, 2011). As detailed in Kumar and Vanajakshi (2015), these models require large databases of historical data to capture seasonal phenomenon. Similar to Kumar and Vanajakshi (2015), a 3 day period is used for model training. From a similar dataset, Kamarianakis *et al.* (2010) used a smoothing spline with 150 degrees of freedom to magnify weekly seasonal patterns and identify structural changes. Smoothing approaches can lead to simple models that can forecast at all horizons.

Harmonic regressions estimate daily periodic signals from a linear regression over a Fourier basis (Metcalfe and Cowpertwait, 2009). For 3-min data, the seasonal period has a length of 480 discrete measures. Using harmonic regression, (L, D)-specific seasonal profiles of $\{Y_t\}$ are fitted. Expressed in Equation 3.3, the harmonic seasonal model is restricted to the first H terms of the Fourier series. This model is not considered a baseline model for its simplicity in estimation but for its simplicity in multi-step forecasting at all horizons. The error $\{\epsilon_t\}$ are assumed i.i.d. with mean 0 and unit variance.

$$Y_t = \mu + \sum_{j=1}^{H} \left[\alpha_j \sin\left(\frac{2\pi tj}{480}\right) + \beta_j \cos\left(\frac{2\pi tj}{480}\right) \right] + \sigma\epsilon_t \tag{3.3}$$

Fitting the model in Equation 3.3 is not difficult since it can also be represented as a high dimensional linear regression model like

$$\boldsymbol{y}_F = \boldsymbol{X}_F \boldsymbol{\theta}_F + \boldsymbol{\epsilon}_F \tag{3.4}$$

where $\boldsymbol{y}_F = [y_1, y_2, \cdots, y_T]'$, $\boldsymbol{\epsilon}_F = [y_1, y_2, \cdots, \epsilon_T]'$, and $\boldsymbol{\theta}_F = [\mu, \alpha_1, \cdots, \alpha_H, \beta_1, \cdots, \beta_H]'$. The model matrix $\boldsymbol{X}_F = [\mathbf{1}, \boldsymbol{SIN}, \boldsymbol{COS}]$ where $\mathbf{1}$ is a $T \times 1$ vector of 1s, $\boldsymbol{SIN}$ is a $T \times H$ matrix containing the Fourier sine terms, and $\boldsymbol{COS}$ is a $T \times H$ matrix containing the Fourier cosine terms.

3.3.6 Special Considerations for Traffic Modeling

The linear matrix form of TAR in Equation 3.2 arises from restricting the set of possible thresholds $\boldsymbol{\delta}$ to a finite set of quantiles based on the sampled transition variable. The high dimensional regression model in Chan *et al.* (2015, 2017) represents a fully saturated TAR model where every realization in the series $\{y_t\}_{t=1}^T$ resides in a different regime. In classic Bayesian handling of TAR and the related smooth transition autoregressive model (STAR), the number of regimes, $(m + 1)$, is fixed. To restrict estimation of the m thresholds to the range of Y_{t-h}, slightly informative *uniform* priors bounded by empirical quantiles $q(p_{min})$ and $q(p_{max})$ ensure that at least $(1 - \min\{p_{min}, 1 - p_{max}\}) \times 100\%$ of the data is represented in the lowest and highest regime (Chen and Lee, 1995; Chen, 1998; Lubrano, 2000; Lopes and Salazar, 2006). Following from literature, $p_{min} = 0.15$ and $p_{max} = 0.85$ are selected to slightly reduce the dimensionality of $\boldsymbol{X}$.

For (L, D, h)-specific traffic models, the maximum number of thresholds is fixed to $m = 50$ and the maximum autoregressive order to $P = 7$. The model matrix $\boldsymbol{X}_R$ of the fully saturated 51-regime TAR(7) has dimension $T^* \times 408$. The fitting period for each model contains $T = 1440$ discrete time realizations of $\{Y_t\}$ leading to $T^* = 1440 - h - 7 + 1 > 408$. The predetermined threshold vector

$\boldsymbol{\delta} = [q(0.15), q(p_2), \cdots, q(p_{49}), q(0.85)]'$ constructed from m evenly spaced percentiles ensures approximately $\frac{0.85-0.15}{50} = 0.014$ of the full time series is represented in each potentially relevant regime. These modifications to the framework of Chan *et al.* (2015, 2017) are made to ensure the dimensionality of the parameter space is not unnecessarily large for practical application. Based on this approach, it is recommended to select m large enough to ensure the set of quantile-based thresholds is dense to not reduce error in misspecified *a priori* selection of $\boldsymbol{\delta}$.

For (L, D)-specific seasonal models, the number of Fourier sine/cosine pairs H must be less than half the period. Regularized estimation of these models using adaptive LASSO (Zou, 2006) indicated that the largest significant harmonic of the model in Equation 3.3 across locations and days was for $j = 139$. Before estimating these seasonal profiles under the Bayesian framework, a maximum number of harmonics, $H = 150$, is chosen. The vector of coefficients $\boldsymbol{\theta}_F$ contains $2H + 1 = 301 < 1440 = T$ parameters that require estimation. To ensure weekly periodic signals are smooth, sparse estimation of $\boldsymbol{\theta}_F$ is desired.

3.4 Bayesian Estimation, Regime Identification, and Subset Selection

The purpose of representing the $(m+1)$-regime TAR process as a high dimensional linear regression is to make Bayesian posterior estimation and model selection computationally feasible for multiple regime TAR models. More importantly, the fully saturated regression $\boldsymbol{y}_R = \boldsymbol{X}_R\boldsymbol{\theta}_R + \boldsymbol{\epsilon}_R$ nests a finite, but extensive, library of (m^*+1)-regime subset TAR(P) models where $0 \leq m^* \leq m$. This includes all linear subset AR(P) models.

For simplicity, let $\boldsymbol{\Theta} = [\boldsymbol{\theta}_R', \sigma^2]' = [\boldsymbol{\theta}_1', \cdots, \boldsymbol{\theta}_{m+1}', \sigma^2]'$. It is believed that the optimal choice m^* is small implying that only m^*+1 of the vectors in $\{\boldsymbol{\theta}_1', \boldsymbol{\theta}_2', \cdots, \boldsymbol{\theta}_{m+1}'\}$ are nonzero implying that $\boldsymbol{\theta}_R$ is sparse. To simultaneously estimate $\boldsymbol{\theta}_R$, choose the optimal m^*, and identify the thresholds, Chan *et al.* (2015) recommends using the penalized group LASSO estimate $\hat{\boldsymbol{\theta}}_{GL}$ of Yuan and Lin (2006) seen in Equation 3.5. The parameter λ controls regularization, $||\cdot||_2$ is the ℓ_2-norm, and $||\cdot||_1$ is the ℓ_1-norm.

$$\hat{\boldsymbol{\theta}}_{GL} = \underset{\boldsymbol{\theta}_R}{\operatorname{argmin}} = \frac{1}{T}||\boldsymbol{y}_R - \boldsymbol{X}\boldsymbol{\theta}_R||_2^2 + \lambda \sum_{j=1}^{m+1} ||\boldsymbol{\theta}_j||_1 \tag{3.5}$$

When $\boldsymbol{X}_R$ is constructed as seen in Chan *et al.* (2015),the set of thresholds identified from $\hat{\boldsymbol{\theta}}_{GL}$ consistently estimates the true thresholds if the true m^* is known *a priori*. In practice, m^* is unknown, and $\hat{\boldsymbol{\theta}}_{GL}$ overestimates the number of regimes. Second stage selection of the best subset of the group LASSO identified thresholds via penalized information criteria (IC), i.e. AIC (Li and Ling, 2012), BIC (Yao, 1988), or MDL (Davis *et al.*, 2006), leads to consistent estimation of the true set of thresholds (Chan *et al.*, 2015). The three-step procedure of Chan *et al.* (2017), primarily based on a group orthogonal greedy algorithm (GOGA) and high dimensional information criteria (HDIC), significantly outperforms two-step group LASSO approach in Chan *et al.* (2015).

The estimation procedures of Chan *et al.* (2015, 2017) focus on estimation and selection of $\boldsymbol{\delta}$ assuming P is known and the same for each regime. The consistency and convergent rate maintain when these assumptions are dropped. Using the Bayesian framework, a three step procedure, outlined in Sections 3.4.1, 3.4.2, and 3.4.3, identifies the important regimes with potentially subset AR(P) dynamics. The order parameter P should be chosen large enough to cover all temporal dynamics across all regimes, and, as previously mentioned, $P = 7$ for all (L, D, h)-specific traffic occupancy subset TAR(P) models.

3.4.1 Bayesian Penalized Estimation

Conditional Likelihood

All Bayesian inference extends from the full posterior distribution $p(\boldsymbol{\Theta}|\boldsymbol{y}_R, \boldsymbol{X}_R)$. As Bayes' rule suggests, the full posterior distribution is expressed as

$$p(\boldsymbol{\Theta}|\boldsymbol{y}_R, \boldsymbol{X}_R) \propto p(\boldsymbol{y}_R|\boldsymbol{X}_R, \boldsymbol{\Theta})p(\boldsymbol{\Theta}) \tag{3.6}$$

where $p(\boldsymbol{y}_R|\boldsymbol{X}_R, \boldsymbol{\Theta})$ is the model likelihood and $p(\boldsymbol{\Theta})$ is the prior. Options for $p(\boldsymbol{\Theta})$ are discussed in the subsequent section, but all immediate attention is on the model likelihood $p(\boldsymbol{y}_R|\boldsymbol{X}_R, \boldsymbol{\Theta})$. Given the linear model $\boldsymbol{y}_R = \boldsymbol{X}_R\boldsymbol{\theta}_R + \boldsymbol{\epsilon}_R$, the likelihood $p(\boldsymbol{y}_R|\boldsymbol{X}_R, \boldsymbol{\Theta})$ stems from a distributional assumption about the errors $\{\epsilon_{\pi(t)+h}\}$ in $\boldsymbol{\epsilon}_R$. So far, we have assumed $\{\epsilon_{\pi(t)+h}\}$ are i.i.d. with mean 0 and variance σ^2. For modeling traffic occupancy, we consider and compare two distribution options.

Assume $\{\epsilon_{\pi(t)+h}\} \sim$ i.i.d. $\mathcal{N}(0, \sigma^2)$ where $\mathcal{N}$ denotes the *normal* distribution. Throughout statistics, this is the most commonly used distribution for the errors. The *normal* regression model is recognized by

$$\boldsymbol{y}_R|\boldsymbol{X}_R, \boldsymbol{\Theta} \sim \mathcal{N}_T(\boldsymbol{X}_R\boldsymbol{\theta}_R, \sigma^2\boldsymbol{I}) \tag{3.7}$$

where $\mathcal{N}_T$ is a T-dimensional *Multivariate normal* distribution and $\boldsymbol{I}$ is a $T \times T$ identity matrix. The *Guassian* error specification for all (L, D, h)-specific TAR models and (L, D)-specific seasonal harmonic profiles is used with caution. Even after aggregating to a 3-min interval, influential spikes toward 0 and 1 are seen. For future reference, we let TAR and SEAS denote the *normal* regression models from Equations 3.2 and 3.4. In both TAR and SEAS, Jefferys' prior is used for σ^2. Derivation of the *inverse-gamma* full conditional distribution for σ^2 under both linear models can be found in Schmidt and Makalic (2016).

Horseshoe+ Priors for Penalized Regression

Mallick and Yi (2013) examines the historical significance of Bayesian model selection approaches in high dimensional linear models. Bayesian penalized regression methods using continuous scale-mixture priors (O'Hara and Sillanpaa, 2009; Polson and Scott, 2010) have been proposed to approximate the spike-and-slab shape of discrete mixture priors (Mitchell and Beauchamp, 1988; George and McCulloch, 1993; Madigan and Raftery, 1994; Carlin and Chib, 1995; Kuo and Mallick, 1998; Ishwaran and Rao, 2005, 2011).

Specifically, the Bayesian horseshoe (BHS) estimator of (Carvalho *et al.*, 2009, 2010) has been extensively researched and shown to have excellent theoretical properties in achieving sparsity (Polson and Scott, 2012; Datta and Ghosh, 2013; Van Der Pas *et al.*, 2014). The BHS prior falls in the extensive class of shrinkage priors with global-local hierarchical representations (Polson and Scott, 2010). As common to regularization techniques, a global tuning parameter is used to enforce variable selection by shrinking coefficients toward 0. However, BHS utilizes additional coefficient-specific tuning parameters to ensure relevant effects are not overshrunk.

The horseshoe+ estimator (BHS^+) of Bhadra *et al.* (2016) results from a slightly modified hierarchy with additional tuning on the local level. The BHS^+ hierarchical prior for each parameter θ_i in the full parameter vector $\boldsymbol{\theta}_R$ is represented as

$$\begin{aligned}
\theta_i|\lambda_i, \tau, \sigma^2 &\sim \mathcal{N}(0, \lambda_i^2\tau^2\sigma^2) \\
\lambda_i &\sim \mathcal{C}^+(0, \eta_i) \\
\eta_i &\sim \mathcal{C}^+(0, 1) \\
\tau &\sim \mathcal{C}^+(0, 1)
\end{aligned} \tag{3.8}$$

where $\mathcal{C}^+$ is the *half-Cauchy* distribution. The BHS^+ prior provides better detection of ultra-sparse signals than the original BHS; therefore, BHS^+ shrinkage priors are preferred in all TAR and SEAS models. Additional theoretical and empirical defense of BHS$^+$ priors in the high dimensional regression setting, see Bhadra *et al.* (2016) and Appendix B.

The original hierarchy seen in Equation 3.8 makes posterior sampling difficult since full conditional distributions are not obtainable. By exploiting the scale-mixture decomposition of the *half-Cauchy* distribution using *inverse gamma* distributions abbreviated $\mathcal{IG}$ (Wand *et al.*, 2011), Makalic and Schmidt (2016) derived full conditional distributions for all parameters in $\boldsymbol{\Theta}$ so Gibbs sampling (Geman and Geman, 1987; Gelfand and Smith, 1990) can be utilized to sample from the full posterior distribution $p(\boldsymbol{\Theta}|\boldsymbol{y}_R, \boldsymbol{X}_R)$. Equation 3.9 reflects the changes to the BHS$^+$ hierarchy for each parameter θ_i in $\boldsymbol{\theta}_R$.

$$\begin{aligned}
\theta_i|\lambda_i^2, \tau^2, \sigma^2 &\sim \mathcal{N}(0, \lambda_i^2\tau^2\sigma^2) \\
\lambda_i^2|\nu_i &\sim \mathcal{IG}(1/2, 1/\nu_i) \\
\tau^2|\xi &\sim \mathcal{IG}(1/2, 1/\xi) \\
\nu_i &\sim \mathcal{IG}(1/2, 1) \\
\xi &\sim \mathcal{IG}(1/2, 1)
\end{aligned} \tag{3.9}$$

Posterior Sampling

The high dimensional TAR and SEAS models, with large $\boldsymbol{X}_R$ and $\boldsymbol{X}_F$ design matrices, causes issues in Gibbs sampling. Specifically, the full conditional distributions of $\boldsymbol{\theta}_R$ and $\boldsymbol{\theta}_S$ require large 408×480 and 301×301 matrices, respectively. To obtain S posterior samples $\{\boldsymbol{\theta}_R^{(s)}\}_{s=1}^S$ and $\{\boldsymbol{\theta}_F^{(s)}\}_{s=1}^S$, inversion of $\boldsymbol{X}_R'\boldsymbol{X}_R$ and $\boldsymbol{X}_S'\boldsymbol{X}_S$ is required in the derived *multivariate normal* full conditional distributions. In all cases, the algorithm of Rue (2001) provides fast Gibbs sampling, but for larger choices of m, P, and H, the algorithm of Bhattacharya *et al.* (2016) is a helpful alternative. For both TAR and SEAS models, $S = 2000$ posterior samples after a burn-in period of

5000 and with a thinning of 10 was large enough to ensure the minimum effective sample size of all parameters was larger than 150. Posterior means and quantiles capture the uncertainty of the models given the data. Even though BHS$^+$ shrinkage priors are applied for both TAR and SEAS, posterior means of irrelevant effects in $\boldsymbol{\theta}_R$ and $\boldsymbol{\theta}_S$ will never equal 0. The profiles obtained from all SEAS models satisfactorily captured the weekly periodic signal from the first three weeks of April. For the TAR models, the three step procedure continues to identify which autoregressive groups are irrelevant, and then perform variable selection ignoring the natural grouping. Let $\mathcal{M}_R$ represent the fully saturated TAR model fitted using BHS*. The primary goal of the next two steps is to search for the best submodel $\mathcal{M}_*$ from the $2^{(m+1)(P+1)}$ different possible submodels $\mathcal{M}_\perp$. When $m = 50$ and $P = 7$, there are 6.61×10^{122} subset TAR(P) models. Naive exploration of this model space is not recommended.

3.4.2 Regime Identification

The samples $\{\boldsymbol{\theta}_R^{(s)}\}_{s=1}^S$ and $\{\sigma^{(s)}\}_{s=1}^S$ from the joint posterior distribution

$$p(\boldsymbol{\theta}_R, \sigma^2 | \mathcal{M}_R, \boldsymbol{y}_R, \boldsymbol{X}_R)$$

is a good starting point for forecasting since BHS$^+$ priors were used to enforce sparsity. Under model $\mathcal{M}_R$, density forecasts at time $T+1$ can be obtained from the posterior predictive distribution represented by

$$p(y_{T+1} | \mathcal{M}_R, \boldsymbol{y}_R, \boldsymbol{X}_R).$$

Assuming the model $\mathcal{M}_R$ produces reasonable forecasts, it can serve as a valid reference model. Given a simpler submodel $\mathcal{M}_\perp$, the Kullback-Leibler (KL) divergence (Kullback and Leibler, 1951) measures the distance between the posterior predictive distributions (Goutis and Robert, 1998; Dupuis and Robert, 2003). If minor discrepancy is detected between $\mathcal{M}_R$ and $\mathcal{M}_\perp$, the more parsimonious model is favored.

Samples $\{\boldsymbol{\theta}_{\perp}^{(s)}\}_{s=1}^{S}$ and $\{\sigma_{\perp}^{(s)}\}_{s=1}^{S}$ from the joint posterior distribution

$$p(\boldsymbol{\theta}_{\perp}, \sigma_{\perp}^{2} | \mathcal{M}_{\perp}, \boldsymbol{y}_{\perp}, \boldsymbol{X}_{\perp})$$

are obtained via projection removing the need for repeated Gibbs sampling. In this representation, $\boldsymbol{y}_{\perp} = \boldsymbol{y}_{R}$ and $\boldsymbol{X}_{\perp}$ contains the columns of $\boldsymbol{X}_{R}$ associated with the submodel $\mathcal{M}_{\perp}$. Piironen and Vehtari (2017) derived analytical solutions to acquire the projected samples and measure the KL divergence from forecasts under $[\boldsymbol{\theta}_{R}^{\prime(s)}, \sigma_{R}^{(s)}]'$ versus $[\boldsymbol{\theta}_{\perp}^{\prime(s)}, \sigma_{\perp}^{(s)}]'$ for linear Gaussian models. The sth posterior sample $[\boldsymbol{\theta}_{\perp}^{\prime(s)}, \sigma_{\perp}^{(s)}]'$ is given in Equation 3.10. For each posterior draw from $p(\boldsymbol{\theta}_{R}, \sigma_{R}^{2} | \mathcal{M}_{R}, \boldsymbol{y}_{R}, \boldsymbol{X}_{R})$, the associated KL divergence $d_{\perp}$ is given Equation 3.11.

$$\begin{aligned} \boldsymbol{\theta}_{\perp}^{(s)} &= (\boldsymbol{X}_{\perp}'\boldsymbol{X}_{\perp})^{-1}\boldsymbol{X}_{\perp}'\boldsymbol{X}_{R}\boldsymbol{\theta}_{R}^{(s)} \\ \sigma_{\perp}^{(s)} &= \sqrt{(\sigma_{R}^{(s)})^{2} + \frac{(\boldsymbol{X}_{R}\boldsymbol{\theta}_{R}^{(s)} - \boldsymbol{X}_{\perp}\boldsymbol{\theta}_{\perp}^{(s)})'(\boldsymbol{X}_{R}\boldsymbol{\theta}_{R}^{(s)} - \boldsymbol{X}_{\perp}\boldsymbol{\theta}_{\perp}^{(s)})}{T}} \end{aligned} \tag{3.10}$$

$$d_{\perp}^{(s)}(\boldsymbol{\theta}_{R}^{(s)}, \sigma_{R}^{(s)}) = \frac{1}{2}\log\left(\frac{\sigma_{\perp}^{(s)}}{\sigma_{R}^{(s)}}\right)^{2} \tag{3.11}$$

Finally, averaging the KL divergences across all posterior samples estimates the overall discrepancy between posterior predictive distributions of $\mathcal{M}_{R}$ and $\mathcal{M}_{\perp}$. This discrepancy, denoted $D(\mathcal{M}_{R}||\mathcal{M}_{\perp})$, is expressed in Equation 3.12.

$$D(\mathcal{M}_{R}||\mathcal{M}_{\perp}) = \frac{1}{S}\sum_{s=1}^{S} d_{\perp}^{(s)}(\boldsymbol{\theta}_{R}^{(s)}, \sigma_{R}^{(s)}) \tag{3.12}$$

Using these concepts, surveying the entire model space is avoided by employing a forward stepwise selection algorithm similar to Piironen and Vehtari (2015b). Starting with the linear AR(P) model, denoted $\mathcal{M}_{\perp}^{(1)}$ where $\boldsymbol{\theta}_{j} = 0$ if $j > 1$ and $\boldsymbol{\theta}_{\perp}^{(1)} = [\boldsymbol{\theta}_{1}', \mathbf{0}', \mathbf{0}', \cdots, \mathbf{0}']'$, the initial discrepancy $D(\mathcal{M}_{R}||\mathcal{M}_{\perp}^{(1)})$ represents the maximum divergence between the fully saturated $\mathcal{M}_{R}$ and all nested TAR(P) models with less than $(m+1)$ regimes. For each $j \in \{2, \cdots, m+1\}$, $\boldsymbol{\theta}_j$ is added to $\boldsymbol{\theta}_{\perp}^{(1)}$ and the best 2-regime TAR(P) model $\mathcal{M}_{\perp}^{(2)}$ that minimizes the discrepancy in Equation 3.12 is selected. Likewise, this procedure is continued to identify the best 3-regime TAR(P), 4-regime TAR(P), and j-regime models, denoted $\boldsymbol{\theta}_{\perp}^{(3)}$, $\boldsymbol{\theta}_{\perp}^{(4)}$, and $\boldsymbol{\theta}_{\perp}^{(j)}$, respectively. Although this process can be continued up to $j = m+1$, where $D(\mathcal{M}_R || \mathcal{M}_{\perp}^{(m+1)}) = 0$, a stopping rule is enforced based on relative explanatory power ($RelE$) given in Equation 3.13 (Dupuis and Robert, 2003). Based on the additive properties of KL, $RelE$ strictly increases from 0 to 1. In the traffic application, regime-specific AR(P) parameter groups are added until $RelE$ exceeds 0.95.

$$RelE(\mathcal{M}_{\perp}) = 1 - \frac{D(\mathcal{M} || \mathcal{M}_{\perp})}{D(\mathcal{M} || \mathcal{M}_{\perp}^{(1)})} \tag{3.13}$$

3.4.3 Subset Variable Selection

Let $\mathcal{J} = \{j : \boldsymbol{\theta}_j \neq 0\}$ indicate the AR(P) parameter groups in $\boldsymbol{\theta}_R$ selected via the algorithm outlined in Section 3.4.2. The set complement $\bar{\mathcal{J}} = \{j : \boldsymbol{\theta}_j = 0\}$ indicates the AR(P) parameter groups believed to be irrelevant. By design, this approach is greedy, and the final $|\mathcal{J}|$-regime TAR(P), recognized as $\mathcal{M}_{\perp}^{(|\mathcal{J}|)}$, is likely to include many irrelevant parameter groups ($|\cdot|$ measures the cardinality of a set). When some subset of the linear AR(P) model is optimal, the initial discrepancy $D(\mathcal{M}_R || \mathcal{M}_{\perp}^{(1)})$ is small. As higher regime models are considered, the reduction in the discrepancy may decrease at a slow rate. This method is recommended when there exists prior understanding that a nonlinear model is advantageous.

Let $\theta_{i,j}$ represent the ith parameter in the jth vector $\boldsymbol{\theta}_j$ for $i \in \{1, 2, \cdots, P+1\}$ and $j \in \{1, 2, \cdots, m+1\}$. Following regime identification, the set $\mathcal{I} = \{\theta_{i,j} : i = 1, \cdots, P+1 \text{ and } j \in \mathcal{J}\}$ contains potentially relevant parameters in $\boldsymbol{\theta}_R$. Because $\mathcal{J}$ may still contain irrelevant AR(P) parameter groups, the regime identification stage can be considered a filtering step leading to a restricted model space with

$2^{|\mathcal{I}|} = 2^{(P+1)|\mathcal{J}|}$ different j^*-regime subset TAR(P) where $j^* \leq |\mathcal{J}|$.

After fixing all $\theta_{i,j} \notin \mathcal{I}$ to 0, the forward selection algorithm is repeated to search for the best and final subset TAR(P) model $\mathcal{M}_*$ resulting in sparse estimation of Equation 3.2. The intercept-only model, where $\theta_{i,j} = 0$ unless $i = j = 1$, is the starting point and identified as $\mathcal{M}_{\perp}^{(1)}$. One-at-a-time parameters from $\mathcal{I}$ are added to the intercept-only model to obtain a a chain of optimal subset models $\boldsymbol{\theta}_{\perp}^{(2)}, \boldsymbol{\theta}_{\perp}^{(3)}, \boldsymbol{\theta}_{\perp}^{(4)}, \cdots$. The superscript of these models does not indicate the number of regimes, but rather the number of nonzero parameters in $\boldsymbol{\theta}_R$. The final model $\mathcal{M}_*$ is identified using the same stopping rule seen in Section 3.4.2. Based on the ordering of $\boldsymbol{\theta}_R$ in Equation 3.2, the subset of parameters in $\mathcal{I}$ selected in $\mathcal{M}_*$ imply the optimal number of regimes $(m^* + 1) \leq (m + 1)$, the optimal choice of the m^* thresholds in δ, and the relevant parameters within each of the regimes.

3.5 Results

On the original $[0, 1]$-interval, traffic occupancy forecasts are evaluated over the final week of April using a rolling-window and without re-estimation. For each (L, D, h)-specific TAR model, there are $T_h = 480 - P - h + 1$ time points requiring forecasts, and for SEAS models, $T_h = 480$. Denote the traffic occupancy forecast at time t for a specific detector location L as $\hat{O}_{L,t}$. Using the (L, D)-specific seasonal models estimated from the first three weeks of April (1440 discrete time points), $\hat{O}_{L,t}$ is quickly obtained for all future horizons. Figure 3.4 displays the SEAS models fitted to the last week of April. The 3 min, 9 min, and 15 min horizon forecasts from all final TAR models are displayed in Figures 3.5, 3.6, and 3.7. In the TAR plots, $(1 - 2\alpha) \times 100\%$ credible regions are displayed instead of point forecasts for $\alpha \in \{0.4, 0.25, 0.2, 0.15, 0.1, 0.05\}$. The true traffic occupancies are plotted in gray for all figures. As illustrated by the SEAS models, the last Friday of April had unusually low congestion relative to the first three Fridays of April. The fact that this Friday also falls on a Grecian holiday weekend (Labor Day) provides a reasonable explanation since vacation time is often used around holidays. Although traffic forecasting is less important for low congested states, this day illustrates the deficiency of ignoring short term temporal dependencies in modeling traffic variables.

Figure 3.4: Forecasts Based on SEAS Models

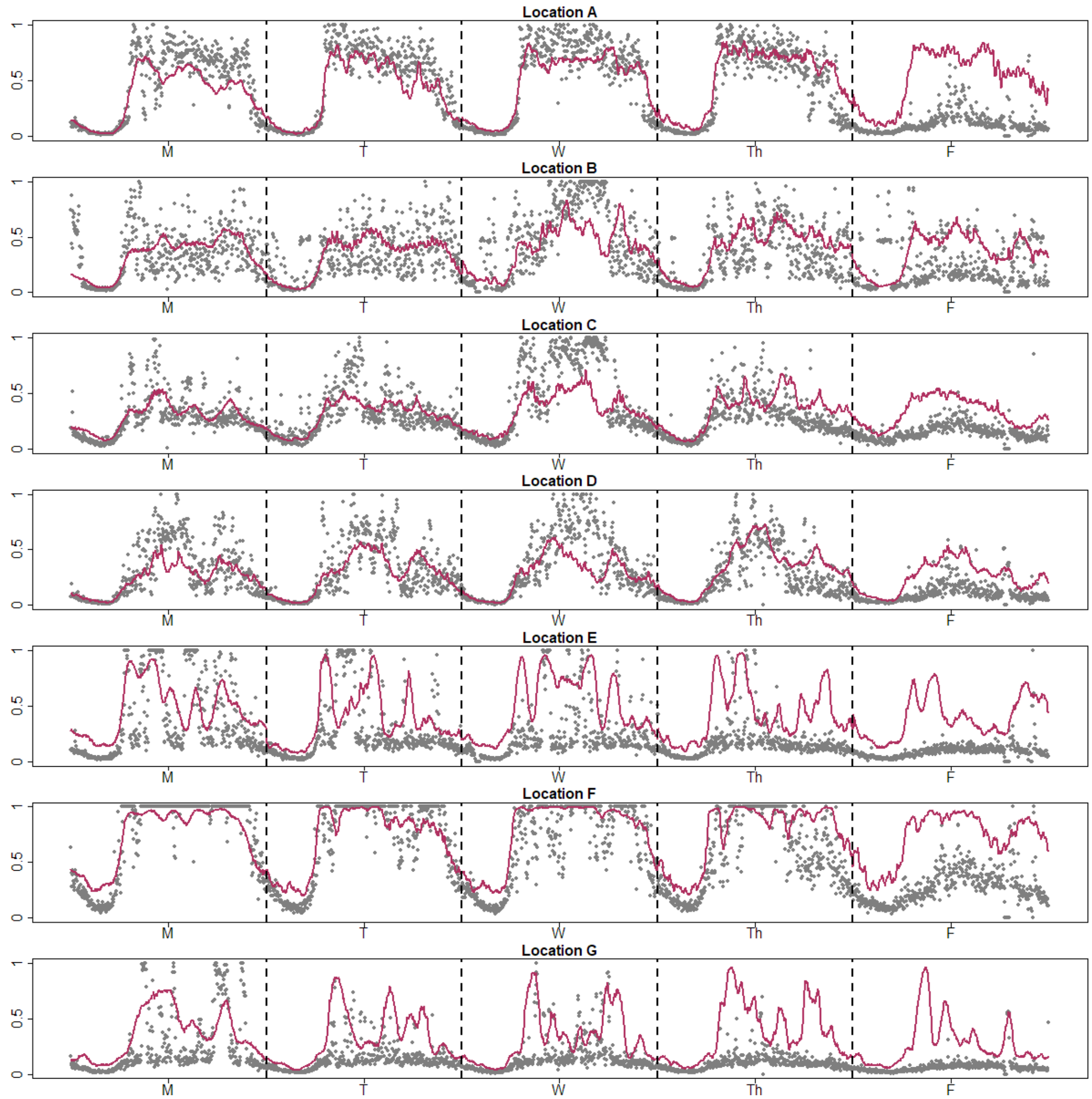


Figure 3.5: 1-Step Ahead Density Forecasts for TAR Models

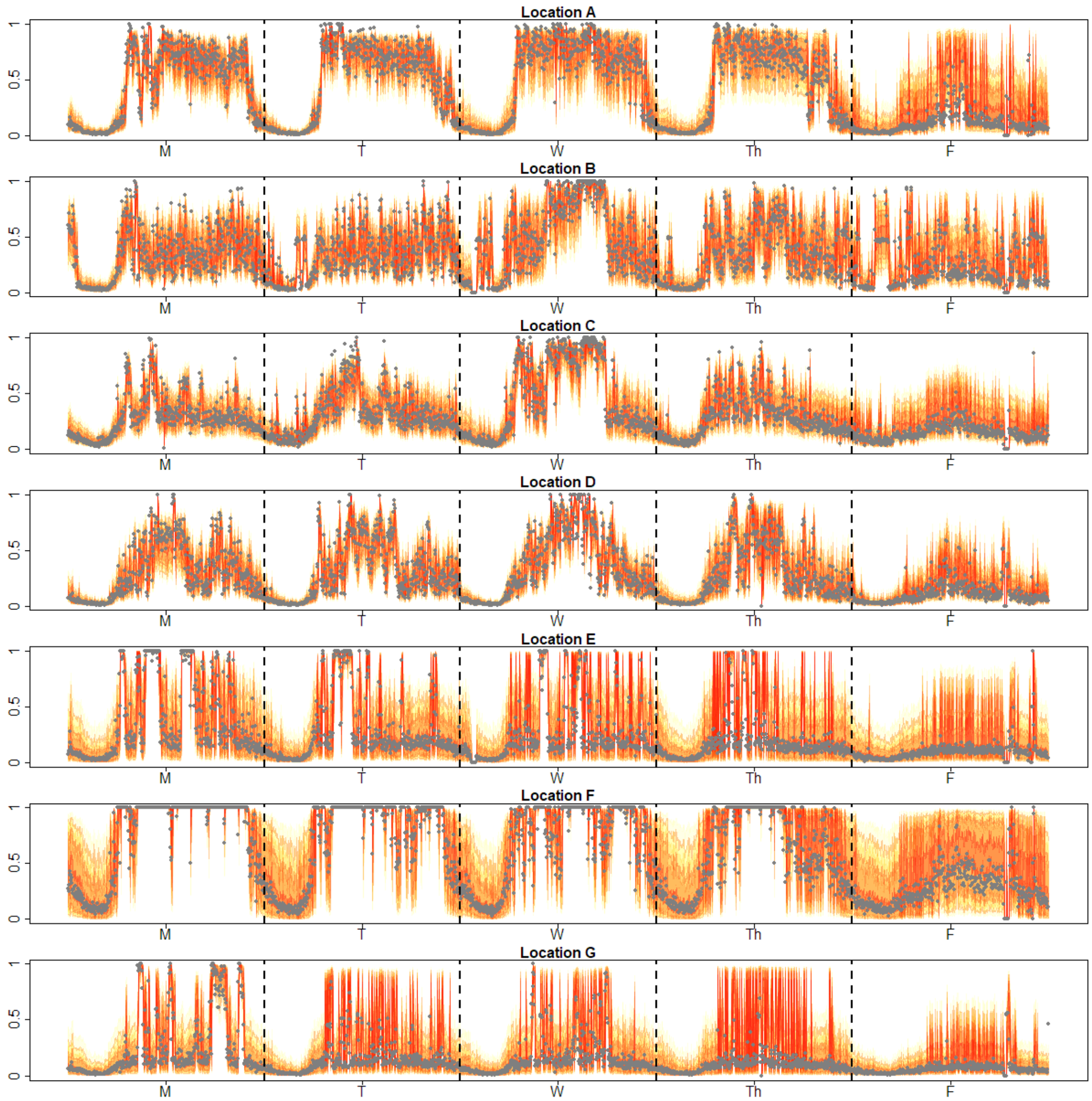


Figure 3.6: 3-Step Ahead Density Forecasts for TAR Models

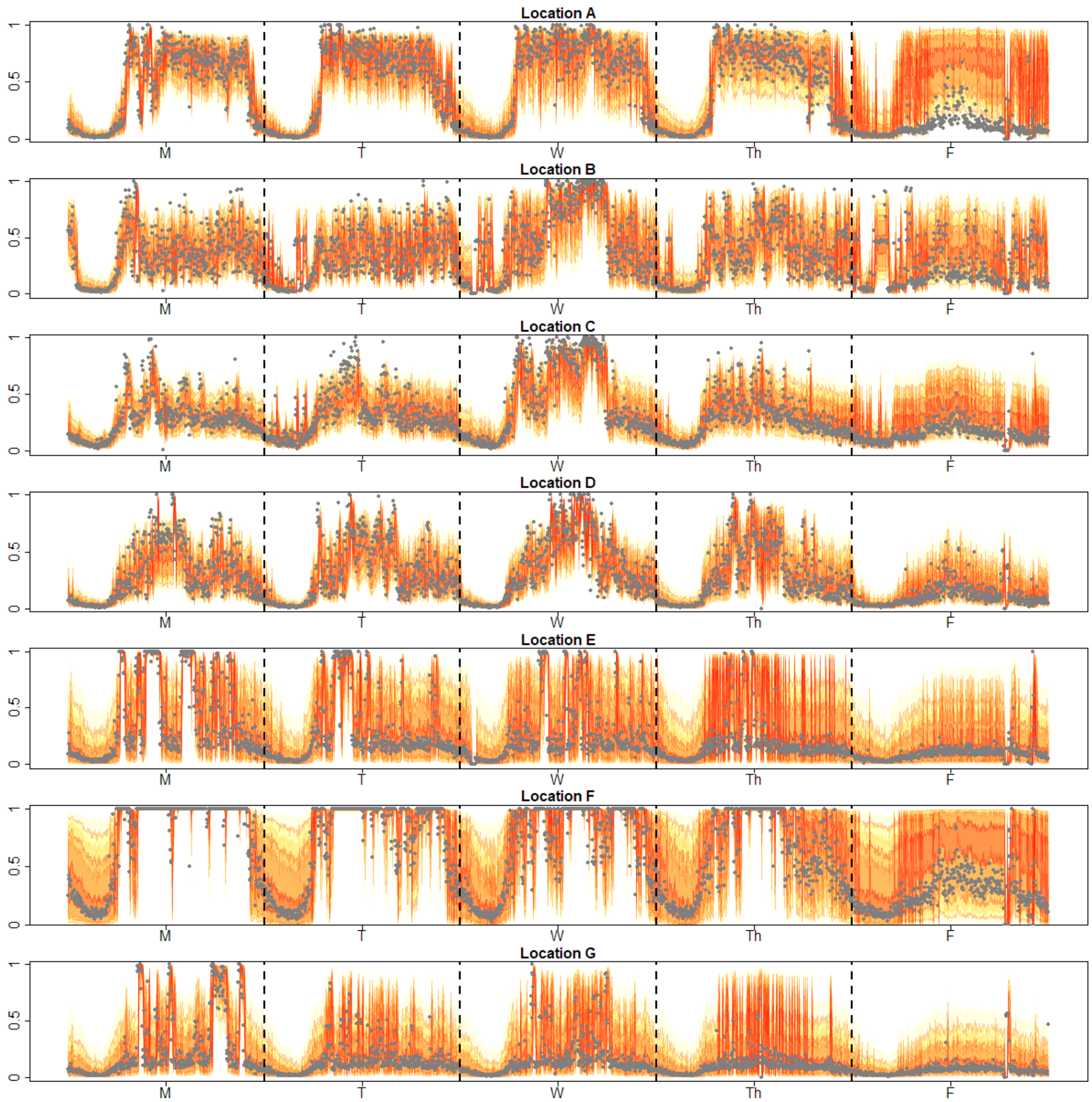
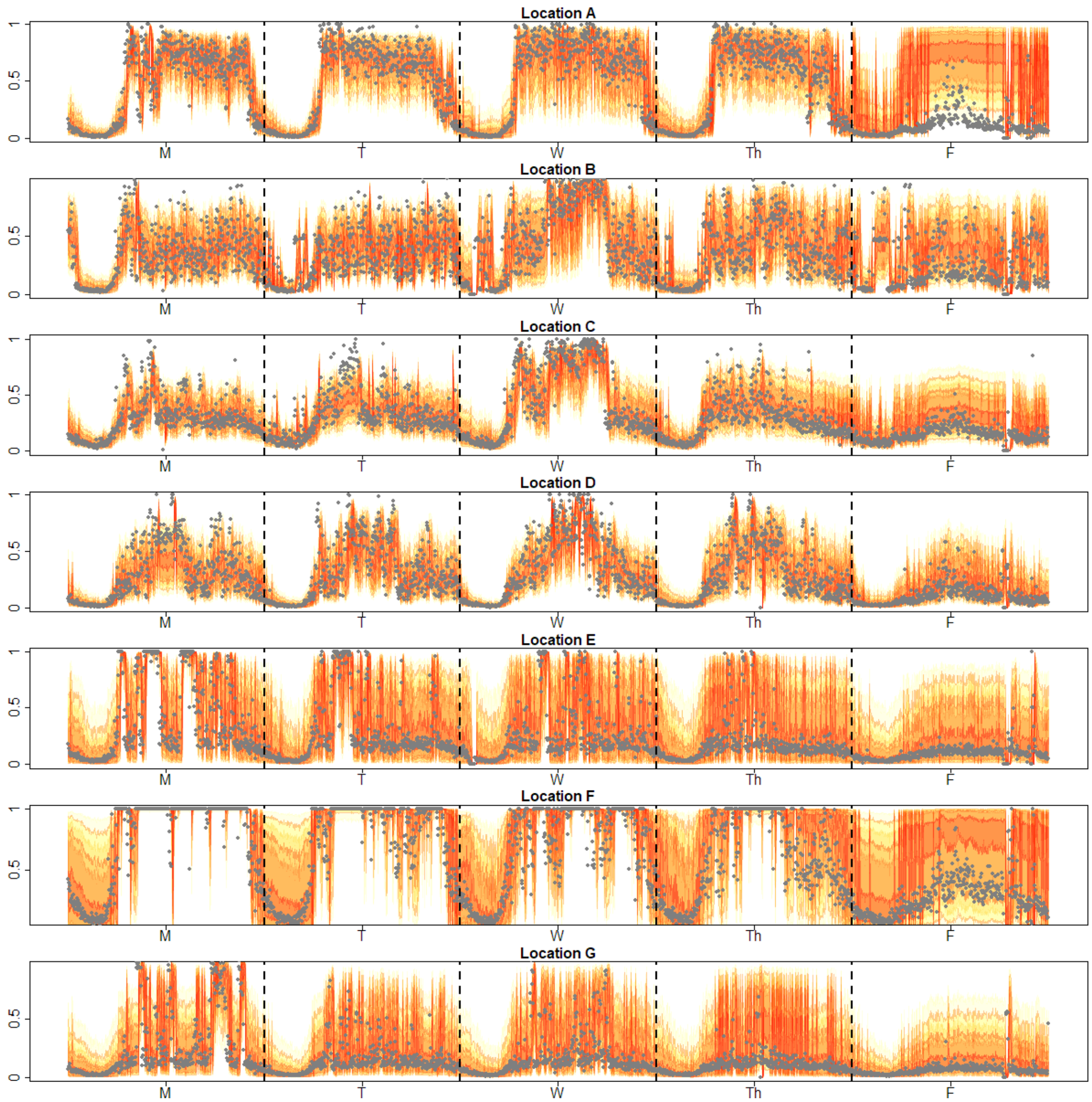


Figure 3.7: 5-Step Ahead Density Forecasts for TAR Models



Forecasts from both models over this period are evaluated using the mean absolute scaled forecast error (MASFE) metric from Hyndman and Koehler (2006). The formulation of MASFE is found in Equation 3.14 where $\text{MAE}_{RW}(h)$ represents the mean absolute error from a h-specific naive random walk (RW) model over the fitting period where $\widehat{O}_{L,t}$ is the observed $O_{L,t-h}$. By scaling errors using $\text{MAE}_{RW}(h)$, TAR can be compared to SEAS and both can be compared to simple naive approaches. Whenever $\text{MASFE}(h) < 1$, the model produces absolute forecast errors that are, on average, less than the forecast errors from a simple random walk model with no parameters.

$$\text{MASFE}(h) = \frac{1}{T_h} \sum_{t=P+h}^{480} \left| \frac{O_{L,t} - \widehat{O}_{L,t}}{\text{MAE}_{RW}(h)} \right| \tag{3.14}$$

Tables 3.1, 3.2, and 3.3 compares final TAR and SEAS models for all days and locations at horizons $h \in \{1, 3, 5\}$, respectively. Multiple-regime TAR models consistently outperform SEAS profiles at all locations and days when forecasting 1-step ahead. For 3-step ahead forecasts, the SEAS model for Tuesday at location C has a smaller MASFE, by a negligible amount. For 5-step ahead forecasts, more occurrences of SEAS producing forecasts, as good or better, than TAR are observed. The opposite pattern is exhibited for h-specific RW models. For 1-step ahead forecasts, MASFE>1 for many of the models. As the horizon h increases, a clear advantage of using more complicated models (TAR and SEAS) to capture nonlinear and/or seasonal dynamics is notices. This is generally true except for locations E, F, and G where traffic occupancies are considerably more difficult to model which is visually indicated by the wide credible regions in Figures 3.5, 3.6, and 3.7.

3.6 Conclusion

Short-term forecasting of traffic occupancy is useful in real-time monitoring of a network. The nonlinearities present in the data make it difficult to get precise pre-

Table 3.1: 1-Step Ahead MASFE Forecast Comparison

				Location				
Day	Model	A	B	C	D	E	F	G
M	TAR	1.02	1.10	0.88	1.07	1.87	0.81	1.48
	SEAS	1.80	1.47	1.15	1.43	4.02	1.57	3.66
T	TAR	0.90	1.05	1.04	0.98	1.36	1.03	1.95
	SEAS	1.35	1.36	1.22	1.46	3.36	1.65	3.29
W	TAR	1.04	1.11	0.91	0.97	2.27	1.86	1.55
	SEAS	1.39	2.01	2.18	1.61	4.65	2.90	2.80
Th	TAR	0.93	0.89	0.82	0.92	1.48	1.52	1.83
	SEAS	1.44	1.43	1.51	1.42	3.98	2.74	4.07
F	TAR	1.80	1.08	1.01	0.85	1.45	2.40	1.16
	SEAS	4.77	2.23	1.98	1.83	4.37	6.24	3.78

Table 3.2: 3-Step Ahead MASFE Forecast Comparison

				Location				
Day	Model	A	B	C	D	E	F	G
M	TAR	0.94	1.06	0.88	1.13	1.85	0.88	1.50
	SEAS	1.36	1.17	0.93	1.14	2.91	1.19	2.57
T	TAR	0.87	1.04	0.96	1.03	1.46	1.15	1.21
	SEAS	1.04	1.06	0.91	1.10	2.24	1.22	2.15
W	TAR	1.09	1.15	1.10	0.99	1.75	2.00	1.26
	SEAS	1.15	1.61	1.69	1.21	2.94	2.24	1.71
Th	TAR	0.90	0.96	0.82	0.93	1.88	1.47	1.14
	SEAS	1.15	1.09	1.14	1.03	2.69	1.96	2.68
F	TAR	3.04	1.09	1.00	0.66	1.14	2.47	0.92
	SEAS	3.53	1.57	1.42	1.33	3.12	4.35	2.42

dictions. Daily traffic occupancy cycles between periods of free flow to periods of congestion. Threshold autoregressions capture many of these nonlinearities by using separate autoregressive processes to model dynamics for different states. Since occupancy is quality measure of traffic flow, this endogenous characteristic can characterize the regimes.

The general estimation difficulties of TAR models lead users to fix the number of regimes prior to fitting the model. In this application to traffic occupancy, the number of thresholds m is fixed and a high dimensional linear model matrix that nests many

Table 3.3: 5-Step Ahead MASFE Forecast Comparison

		Location						
Day	Model	A	B	C	D	E	F	G
M	TAR	0.94	0.99	0.89	1.12	1.97	0.86	1.68
	SEAS	1.24	1.06	0.88	1.06	2.46	1.08	2.17
T	TAR	0.81	1.05	0.95	1.00	1.47	1.13	1.15
	SEAS	0.95	0.95	0.85	0.99	1.85	1.07	1.77
W	TAR	1.02	1.12	1.04	0.99	1.67	1.94	1.26
	SEAS	1.01	1.44	1.56	1.12	2.30	1.95	1.44
Th	TAR	0.84	0.98	0.82	0.87	1.51	1.48	1.22
	SEAS	1.05	1.03	1.07	0.94	2.13	1.73	2.24
F	TAR	2.85	1.20	0.88	0.70	1.26	2.52	1.00
	SEAS	3.07	1.48	1.33	1.19	2.59	3.82	2.01

TAR models with regimes less than $m+1$ is constructed. Sparse estimation of the coefficients not only identifies the optimal number of regimes, but also selects the thresholds. After fixing the maximum autoregressive order P and choosing the m thresholds, we present a three step model building procedure that automates subset TAR(P) selection.

Using a metric scaled by the MAE of a naive random walk evaluated from the training period, advantages of TAR models over sparse periodic seasonal signals are discovered for forecasting 3, 9, and 15 minutes ahead. Nevertheless, there is room for improvement. The outlined posterior prediction projective method for subset selection of TAR requires the assumption that errors follow a *normal* distribution with zero mean and constant variance. Although we estimate linear models using logit transformed data and capture some of the heteroskedasticity when we convert back to the original scale, it may be advantageous to modify the approaches with robust *Student t* errors, within-regime homoskedasticity, or stochastic volatility models for the variance.

Chapter 4

REGULARIZATION METHODS FOR SUBSET ARMA SELECTION

4.1 Introduction

Let $\{y_t : t = 1, 2, \cdots, T\}$ be a sequentially observed discrete and equally-spaced sample from a weakly stationary, homoskedastic process $\{Y_t : t = \cdots, -1, 0, 1, \cdots\}$. For the purpose of forecasting future realizations i.e. $\hat{y}_{T+h}$ where $h \in \mathbb{N}$, a model of the general form

$$y_t = f(y_{t-1}, y_{t-2}, \cdots, y_{t-p}, \epsilon_{t-1}, \epsilon_{t-2}, \cdots, \epsilon_{t-q}) + \epsilon_t$$

is used. Under homoskedasticity, $\{\epsilon_t\}$ is assumed to be white noise with mean 0 and variance σ^2. Finite order parameters $p, q \in \mathbb{N}$ quantify the strength that past information has on prediction. Define $m = \max\{p, q\}$. In most cases, m is small relative to T; however, when cyclical phenomenon is detected, $m \geq s$ where s is the seasonal periodicity. The latter scenario leads to long gaps in relevant information for forecasting.

The seasonal autoregressive moving average (SARMA) process, popularized by Box and Jenkins (1976), jointly models the temporal short-term and seasonal dynamics of $\{y_t\}$ to forecast future unknown realizations. Let B represent the backshift operator where $B^k y_t = y_{t-k}$ and define polynomial functions $\Phi(B^s) = 1 - \sum_{J=1}^{P} \Phi_J B^{sJ}$, $\phi(B) = 1 - \sum_{j=1}^{p} \phi_j B^j$, $\Theta(B^s) = 1 + \sum_{K=1}^{Q} \Theta_K B^{sK}$, and $\theta(B) = 1 + \sum_{k=1}^{q} \theta_k B^K$. If the seasonal periodicity $s > 1$ is known, the SARMA$(p, q) \times (P, Q)_s$ process in Equation 4.1 represents a viable family of models for forecasting.

$$\Phi(B^s)\phi(B)y_t = \Theta(B^s)\theta(B)\epsilon_t \tag{4.1}$$

The seasonal periodicity s is typically unknown *a priori*. Any SARMA model from Equation 4.1 algebraically reduces to an ARMA(p^*, q^*) process $\phi^*(B)y_t = \theta^*(B)\epsilon_t$ where $\max\{p^*, q^*\} = \max\{Ps + p, Qs + q\}$ where $[p, P, q, Q, s]' \in \mathbb{N}^5$. For example, consider a quarterly SARMA$(1,0) \times (1,0)_4$ process $\{x_t\}$ where $\phi_1 = 0.6$ and $\Phi_1 = 0.3$. The temporal dynamics of $\{x_t\}$ are equivalently modeled using an ARMA$(5,0)$ process such that $\boldsymbol{\phi} = [\phi_1, \phi_2, \phi_3, \phi_4, \phi_5]' = [0.6, 0, 0, 0.3, -0.18]'$ (see Equation 4.2).

$$\begin{aligned}
\Phi(B^4)\phi(B)x_t &= \epsilon_t \\
(1 - 0.3B^4)(1 - 0.6B)x_t &= \epsilon_t \qquad (4.2) \\
(1 - 0.6B - 0.3B^4 + 0.18B^5)x_t &= \epsilon_t
\end{aligned}$$

Fitting an ARMA(p^*, q^*) model to an arbitrary series $\{y_t\}$ requires estimation of AR coefficients $\boldsymbol{\phi} = [\phi_1, \cdots, \phi_{p*}]'$ and MA coefficients $\boldsymbol{\theta} = [\theta_1, \cdots, \theta_{q^*}]'$. Estimates $\hat{\boldsymbol{\phi}}$ and $\hat{\boldsymbol{\theta}}$ that validate stationary and invertible regulatory assumptions are desired. Stationarity and invertibility require all roots of both characteristic equations, $1 - \phi_1 z - \phi_2 z^2 - \cdots - \phi_{p^*} z^{p^*} = 0$ and $1 + \theta_1 z + \theta_2 z^2 + \cdots + \theta_{q^*} z^{q^*} = 0$, to be outside the unit circle. Classically, parameter estimation is conducted via method of moments, least squares, or maximum likelihood (Hamilton, 1994; Cryer and Chan, 2008). When $q^* = 0$, these approaches are simple extensions of linear regression where the set of predictor variables are lagged realizations of the time series. If $q^* > 0$, a linear model representation exists, but the presence of MA terms poses an estimation problem since the innovations $\{\epsilon_t\}$ are unobservable and dependent on $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$. Popular least squares and maximum likelihood estimation methods become far less efficient and require nonlinear optimization techniques.

Any ARMA(p^*, q^*) model that satisfies the invertibility condition has an $AR(\infty)$ representation, i.e. $(1 - \sum_{j=1}^{\infty} \phi'_j B^j)y_t = \epsilon_t$. If $\boldsymbol{\phi}'$ is known, the full set of $\{\epsilon_t\}$ can be obtained. The residuals $\{\hat{\epsilon}_t : t = p' + 1, \cdots, T\}$ of a long AR(p') process fitted to $\{y_t\}$ can approximate the unobserved $\{\epsilon_t\}$. This approach was initially proposed by

Hannan and Rissanen (1982) to obtain quick estimation of ARMA(p^*, q^*) as it avoids previously mentioned estimation issues. For further information, see Brockwell and Davis (2016, pp. 156-158).

The model orders p^* and q^* can be heuristically selected through inspection of sample autocorrelation and partial autocorrelation functions (abbreviated ACF and PACF, respectively). This non-scientific approach could lead to misspecified models and possibly poor forecasting performance. Suppose p and q are safe upper bounds such that $p \geq p^*$ and $q \geq q^*$. For the pq different ARMA models, final order selection can be based off minimization of some measure of prediction error (PE). Information criteria such as AIC (Akaike, 1974) or BIC (Schwarz, 1978) are popular metrics that penalize for model complexity. Stepwise selection algorithms are usually instituted to accelerate this process.

These approaches are best suited for estimating ARMA processes where $\phi_j \neq 0$ and $\theta_k \neq 0$ for $j \in \{1, \cdots, p^*\}$ and $k \in \{1, \cdots, q^*\}$. For the scenario in Equation 4.2, correct identification of $p^* = 5$ and $q^* = 0$ still leads to overfitting since truly zero parameters, ϕ_2 and ϕ_3, are included in estimation. The true process in Equation 4.2 is a a subset ARMA(5, 0) model where $\boldsymbol{\phi} = [\phi_1, \phi_4, \phi_5]' = [0.6, 0.3, -0.18]'$. Common approaches for ARMA(p, q) model selection become less efficient and reliable when searching through the $2^{(p+q)}$ unique subset ARMA(p, q) models.

Let $\boldsymbol{y} = [y_m, \cdots, y_T]'$, $\boldsymbol{\epsilon} = [\epsilon_m, \cdots, \epsilon_T]'$, $\boldsymbol{\beta} = [\boldsymbol{\phi}', \boldsymbol{\theta}']' = [\phi_1, \cdots, \phi_p, \theta_1, \cdots, \theta_q]'$, and

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}'_m \\ \boldsymbol{x}'_{m+1} \\ \vdots \\ \boldsymbol{x}'_T \end{bmatrix} = \begin{bmatrix} y_{m-1} & \cdots & y_{m-p} & \hat{\epsilon}_{m-1} & \cdots & \hat{\epsilon}_{m-q} \\ y_m & \cdots & y_{m-p+1} & \hat{\epsilon}_m & \cdots & \hat{\epsilon}_{m-q+1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ y_{T-1} & \cdots & y_{T-p} & \hat{\epsilon}_{T-1} & \cdots & \hat{\epsilon}_{T-q} \end{bmatrix}.$$

The ARMA(p, q) model is equivalently represented by $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$. Recall that $\hat{\epsilon}_t$

are residuals from fitted AR(p') models used to estimate unknown innovations. Similar to estimation via conditional least squares and conditional maximum likelihood (Hamilton, 1994), the first $m-1$ observations are lost in parameter estimation where $m = p' + max\{p, q\} + 1$. For reduction of m, selection of p' can be based off AIC or BIC (Hannan and Kavalieris, 1984; Chen and Chan, 2011). Also, it is important to note $\{y_t\}$ is assumed to be mean-centered. An additional mean parameter μ can be included in $\boldsymbol{\beta}$ via binding a column of 1s to $\boldsymbol{X}$.

Presenting the ARMA(p, q) model as a linear Gaussian model is quite advantageous. For both linear and generalized linear models, the least absolute shrinkage and selection operator (LASSO) of Tibshirani (1996) efficiently combines model selection and estimation. The LASSO estimator in Equation 4.3 achieves sparsity through ℓ_1 penalization of the least squares criterion. The tuning parameter $\lambda > 0$ controls overall shrinkage of $\boldsymbol{\beta}$ towards 0. Consequentially, the LASSO estimate is a function of λ, but full solution paths are quickly obtained via well-developed algorithms (Efron *et al.*, 2004). The optimal λ is often based off minimization of AIC, BIC, or some generalization of prediction error. The effectiveness of LASSO motivated analogous Bayesian approaches using Laplace priors (Park and Casella, 2008; Yuan and Lin, 2005). Similarly, hyperpriors placed on λ encourage data-driven shrinkage of posterior estimates.

$$\hat{\boldsymbol{\beta}}_L(\lambda) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \; ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||^2 + \lambda \sum_{i=1}^{p+q} |\beta_i| \tag{4.3}$$

Applying LASSO in time series analysis is potentially problematic since the ARMA model matrix $\boldsymbol{X}$ contains correlated predictors. Nardi and Rinaldo (2011) explored the consistency properties of $\hat{\boldsymbol{\beta}}_L$ for AR(p) processes to approximate realizations from ARMA data generating processes (DGPs). However, high correlation between nonzero and irrelevant ARMA predictors may violate the "irrepresentable condition" required for sign and model selection consistency (Zhao and Yu, 2006). Hebiri and Lederer (2013) demonstrate that highly collinear designs yield underestimation of λ and poor prediction. Modified LASSO and other methods with better asymptotic properties mitigate the consequences of correlated predictors.

In this context, p and q should be safely overestimated, resulting in a sparse parameter vector $\boldsymbol{\beta}$. In this article, the application of regularization methods to automate subset ARMA(p, q) selection and estimation of $\boldsymbol{\beta}$ is explored. Section 4.2 presents three different methods that incorporate subset selection through regularization estimation. The first two methods extend off work from Chen and Chan (2011). A discussion of cross-validation techniques explores alternative ways to select regularization tuning parameters. The final regularization method is developed under the Bayesian framework for a contrast to the preceding classical approaches. Section 4.3 contains simulation studies evaluating and comparing the different methods. Section 4.4 applies the methods to monthly carbon dioxide time series collected from two atmospheric observatories.

4.2 Methods

Assume y_t follows a subset ARMA(p, q) process. Recall the matrix ARMA representation $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$ where $\boldsymbol{\beta} = [\boldsymbol{\phi}', \boldsymbol{\theta}'] = [\beta_1, \cdots, \beta_{p+q}]'$. The set $\mathcal{V} = \{i : \beta_i \neq 0\}$ indicates the AR and MA terms relevant to the true process. If the cardinality $|\mathcal{V}| < p + q$, irrelevant predictors are included in the ARMA model matrix $\boldsymbol{X}$. Given observed data $\{y_t : t = m, m+1, \cdots, T\}$, $\hat{\boldsymbol{\beta}}$ is an estimator for $\boldsymbol{\beta}$ and $\hat{\mathcal{V}} = \{i : \hat{\beta}_i \neq 0\}$ for $\mathcal{V}$. Multiple researchers have theoretically explored the asymptotic behavior of penalized estimators including the popular oracle property (Fan and Li, 2001; Fan and Peng, 2004; Fan and Lv, 2011). A method for estimating $\boldsymbol{\beta}$ is described as oracle if the estimator $\hat{\boldsymbol{\beta}}$ asymptotically behaves as an estimator developed under prior knowledge of $\mathcal{V}$. Under these considerations, outlined approaches estimate ARMA coefficients while simultaneously identifying $\mathcal{V}$ through shrinking irrelevant effects to 0.

4.2.1 Adaptive LASSO

Zou (2006) highlighted the conditional consistency of LASSO and introduced adaptive LASSO (ADLASSO) which enjoys the oracle properties. For a chosen $\eta > 0$, define the vector of weights $\hat{\boldsymbol{w}} = |\hat{\boldsymbol{\beta}} + 1/T|^{-\eta}$ where $\hat{\boldsymbol{\beta}}$ represents an initial estimate of $\boldsymbol{\beta}$ derived using ordinary least squares (OLS), ridge, or LASSO regression. The additional $1/T$ exists so division by 0 is prevented. The ADLASSO estimator $\hat{\boldsymbol{\beta}}_{AL}$ is described in Equation 4.4. The tuning parameter $\lambda > 0$ controls the degree of penalization across all ARMA terms while coefficient-specific weights fine tune shrinkage.

$$\hat{\boldsymbol{\beta}}_{AL}(\lambda) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\ ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||^2 + \lambda \sum_{i=1}^{p+q} \hat{w}_i |\beta_i| \tag{4.4}$$

For subset ARMA model selection, Chen and Chan (2011) showed ADLASSO is an oracle procedure under 3 regulatory assumptions when $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{y}$. Proof of this result followed from using a long AR(p') process to estimate unknown innovations. Simulation results indicated best empirical performance when the initial estimate $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}_L$. Following from Zou (2006) and Chen and Chan (2011), $\eta = 2$ and $\hat{\boldsymbol{\beta}}_L$ is used for the formulation of $\hat{\boldsymbol{w}}$.

4.2.2 Adaptive Elastic Net

The ADLASSO procedure has become popular in time series analysis since parsimonious models typically improve forecasting. Incorporating lags of exogenous time series in $\boldsymbol{X}$ adds complexity that ADLASSO can discriminate against. Assuming information becomes less relevant for forecasting as time passes has encouraged modifications for more complicated full models. For example, lag lengths can be included in the functional representation of $\hat{\boldsymbol{w}}$ to further encourage penalization for long-lagged terms (Park and Sakaori, 2013; Konzen and Ziegelmann, 2016). When seasonal effects are prevalent, these ADLASSO modifications may completely eliminate important terms at long lags.

The elastic net (ENET) of Zou and Hastie (2005) has applicability in this context where $\boldsymbol{X}$ contains two groups of predictors with potentially high pairwise collinearity. Although variable selection benefits of LASSO would be lost, the ridge estimator of Hoerl and Kennard (1970) could lead to better forecasting. The ENET estimator in Equation 4.5 introduces another tuning parameter $\alpha \in [0, 1]$ to influence the trade-off between ℓ_1 and ℓ_2 penalties (De Mol *et al.*, 2009). The original motivation of ENET was to overcome model selection limitations of LASSO when the number of parameters is larger than the sample size, a common problem in bioinformatic data (Zou and Hastie, 2005). This problem is not prevalent in time series analysis; however, seasonal dynamics, which require multiple cycles to estimate, are difficult to identify when data is limited and/or the period is large. Hypothetically, it makes sense to evaluate empirical performance of ENET in this context.

$$\hat{\boldsymbol{\beta}}_E(\lambda, \alpha) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \; ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||^2 + \lambda \left[(1-\alpha) \sum_{i=1}^{p+q} \beta_i^2 + \alpha \sum_{i=1}^{p+q} |\beta_i| \right] \tag{4.5}$$

As previously seen, ADLASSO satisfies the oracle properties (Zou, 2006) and ENET (Zou and Hastie, 2005) manages collinearity. Zou and Zhang (2009) exploit both advantages by modifying the ℓ_1 penalty Equation 4.5 to match the weighted form in Equation 4.4. This adaptive ENET (ADENET) estimator is formally presented in Equation 4.6. Zou and Zhang (2009) recommend selecting $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}_E$. Since $\hat{\boldsymbol{\beta}}_E$ depends on the choice of two tuning parameters, λ and α, optimal selection requires a grid search. Upon empirical evaluation, setting $\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}_L$ is sufficient for obtaining the initial weights.

$$\hat{\boldsymbol{\beta}}_{AE}(\lambda,\alpha) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\ ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}||^2 + \lambda\left[(1-\alpha)\sum_{i=1}^{p+q}\beta_i^2 + \alpha\sum_{i=1}^{p+q}\hat{w}_i|\beta_i|\right] \tag{4.6}$$

4.2.3 Options for Selecting Tuning Parameters

The adaptive estimators $\hat{\boldsymbol{\beta}}_{AL}(\lambda)$ and $\hat{\boldsymbol{\beta}}_{AE}(\lambda,\alpha)$ depend on choices for λ and α. Given finite sets $\mathcal{L} = \{\lambda_j > 0 : j = 1, \cdots, J\}$ and $\mathcal{A} = \{0 < \alpha_k < 1 : k = 1, \cdots, K\}$, full solution paths for both estimators can be produced via LARS algorithm (Efron *et al.*, 2004) or coordinate descent (Friedman *et al.*, 2010). Essentially, each $\lambda \in \mathcal{L}$ and $\alpha \in \mathcal{A}$ corresponds to a different subset ARMA(p,q) model, equating to $|\mathcal{L}|$ different ADLASSO models and $|\mathcal{L}| \times |\mathcal{A}|$ different ADENET models. The optimal λ^* and α^* should be empirically chosen based off some estimate of forecasting performance. In this section, different algorithms to select final subset ARMA models, $\hat{\boldsymbol{\beta}}_{AL} = \hat{\boldsymbol{\beta}}_{AL}(\lambda^*)$ and $\hat{\boldsymbol{\beta}}_{AE} = \hat{\boldsymbol{\beta}}_{AE}(\lambda^*, \alpha^*)$, are explained. See Hastie *et al.* (2009, pp. 241-254) for classic approaches to select tuning parameters.

SELECTION BASED ON AIC OR BIC

Popular information criteria AIC and BIC can be used to select tuning parameters λ and α. These penalized measures of error effect model selection for the initial LASSO-based weights $\hat{\boldsymbol{w}}$ and final models. To quantify model complexity, consider the approximate degrees of freedom $\hat{v}(\lambda)$=$|\widehat{\mathcal{V}}(\lambda)|$ where $\widehat{\mathcal{V}}(\lambda) = \{i : \hat{\beta}_i \neq 0\}$ (Zou *et al.*, 2007). The AIC and BIC formulas for LASSO and ADLASSO are given in Equation 4.7. For ENET and ADENET, $\hat{\boldsymbol{\beta}}(\lambda)$ and $\hat{v}(\lambda)$ must be replaced with $\hat{\boldsymbol{\beta}}(\lambda,\alpha)$ and $\hat{v}(\lambda,\alpha)$.

$$\begin{aligned} \mathrm{AIC}(\hat{\boldsymbol{\beta}}(\lambda)) &= 2\hat{v}(\lambda) + (T-m+1)\log\left(\frac{||\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}(\lambda)||^2}{T-m+1}\right) \\ \mathrm{BIC}(\hat{\boldsymbol{\beta}}(\lambda)) &= \log(T-m+1)\hat{v}(\lambda) + (T-m+1)\log\left(\frac{||\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}(\lambda)||^2}{T-m+1}\right) \end{aligned} \tag{4.7}$$

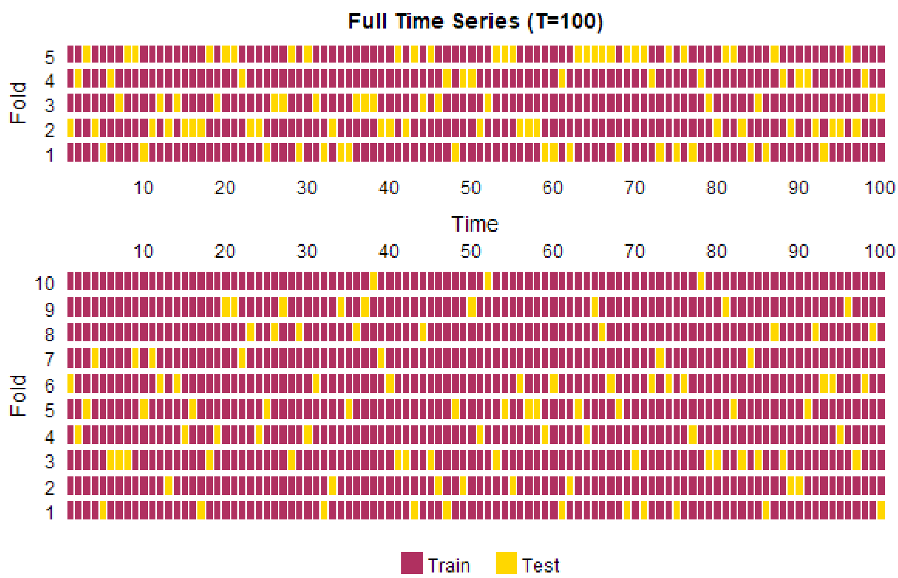
Choice of information criteria (BIC or AIC) is not a Bayesian versus non-Bayesian argument, but an argument about whether true models exist and can be discovered (Burnham and Anderson, 2003). Empirical analysis indicates AIC more frequently outperforms in prediction, but BIC's stronger penalty notoriously leads to better model selection (Burnham and Anderson, 2004). The true complexity of the unknown DGP and the path of AIC/BIC influence this decision(Shao, 1997; Burnham and Anderson, 2003). Chen and Chan (2011) consider AIC and BIC in both stages of ADLASSO and acknowledge this phenomenon in simulation of subset ARMA models. Averaged models weighted based off AIC and BIC are often superior in prediction to individual models, but this is out of the scope of this paper (Burnham and Anderson, 2004).

Philosophical differences aside, both measures are utilized in model selection and forecasting. ADLASSO and ADENET are two stage procedures. In Chen and Chan (2011), only BIC is used for LASSO estimated weights. These weights are crucial in the overall effectiveness of both estimation algorithm. If the BIC penalty overshrinks estimates toward 0, relevant parameters can be unrecognized in the second stage regardless of whether AIC or BIC are used. AIC is favored in the first stage providing safer protection against losing too many key variables. To provide comparison to Chen and Chan (2011), consider the three of four possible combinations: AIC in both stages, AIC then BIC, and BIC in both stages. The third option was evaluated in Chen and Chan (2011).

Selection Based on Cross-Validation

Optimal tuning parameters for regularization are typically chosen via cross-validation (CV) (Hastie *et al.*, 2009). This approach has been popular for model selection in classic linear regression since Stone (1974). For K-Fold CV (CV-K), begin by splitting the usable $T-m+1$ portion of the time series into K separate folds. Each fold acts as a testing period for models fitted to remaining data. Figure 4.1 illustrates this partitioning for CV-5 and CV-10 assuming $T-m+1=100$. Random assignment of data to K folds leads to approximately $100/K$ prediction points in each data split.

Figure 4.1: General K-fold Cross-Validation for Model Selection for $K=5$ (top) and $K=10$ (bottom)



Following similar notation from Hastie *et al.* (2009), $\kappa : \{m, m+1, \cdots, T\} \rightarrow \{1, \cdots, K\}$ is the indexing function mapping data to specific testing groups. An estimate of PE is obtained for each $\lambda \in \mathcal{L}$ and $\alpha \in \mathcal{A}$, and optimal tuning parameters are chosen based on minimization of this estimate. Specifically for the LASSO cases, $\widehat{\text{PE}}(\lambda)$ is expressed in Equation 4.8. Use $\hat{\boldsymbol{\beta}}_{\kappa(t)}(\lambda)$ to represent the estimated ARMA parameters from models fitted to data not in the $\kappa(t)$ group. The most exhaustive case is leave-one-out CV (LOOCV) where $K=(T-m+1)$ and $\kappa(t)=t-m+1$. Obtaining $\widehat{\text{PE}}(\lambda)$ for LOOCV would be time consuming if not for the generalized CV

(GCV) of Wahba and Craven (1978).

$$\widehat{\mathrm{PE}}(\lambda) = \frac{1}{T-m+1}\sum_{t=m}^{T}\left(y_t - \boldsymbol{x}_t'\hat{\boldsymbol{\beta}}_{\kappa(t)}(\lambda)\right)^2 \tag{4.8}$$

Selection Based on Out-of-Sample Evaluation

Applied statisticians prefer CV-K or LOOCV when data is cross-sectional. This approach is not intuitive for time series data where prediction on a randomly selected subset of the full data does not seem like forecasting. For a particular $\tau \in (0,1)$, the out-of-sample (OOS) method estimates $\hat{\boldsymbol{\beta}}(\lambda)$ from the first $(1-\tau)\times 100\%$ of the data (TRAIN) and forecasts on the final $\tau \times 100\%$ (TEST). Equation 4.9 equates to mean squared forecast error (MSFE) and is used to optimally select tuning parameters.

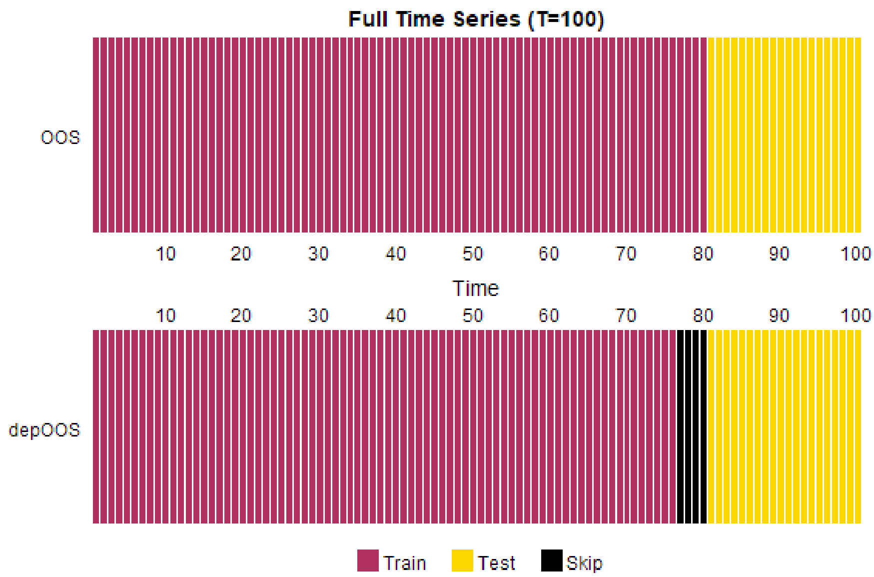
$$\widehat{\mathrm{PE}}(\lambda) = \frac{1}{\tau T}\sum_{t\in\mathrm{TEST}}\left(y_t - \boldsymbol{x}_t'\hat{\boldsymbol{\beta}}(\lambda)\right)^2 \tag{4.9}$$

For subset ARMA selection, order parameters p and q are fixed and quantify the memory required to forecast. In this naive description of OOS, the fact, that some of the forecasts in the TEST period are obtained using data in the TRAIN period, is ignored. Given p and q, the final $d = \max\{p,q\}$ points in the TRAIN period are neglected in model fitting. Now, models are strictly evaluated on future data independent of the TRAIN period. In some literature, this is default OOS (Bergmeir *et al.*, 2018); however, this modified version, abbreviated depOOS, highlights the additional considerations being made. Figure 4.2 displays the difference data division between OOS and depOOS.

Selection Based on Blocked CV

Classic CV estimates the expected PE constructed from predictions on unfitted data. This may lead to a poor estimate for time series data where the popular "independent observation" assumption is violated (Arlot *et al.*, 2010). Burman *et al.* (1994)

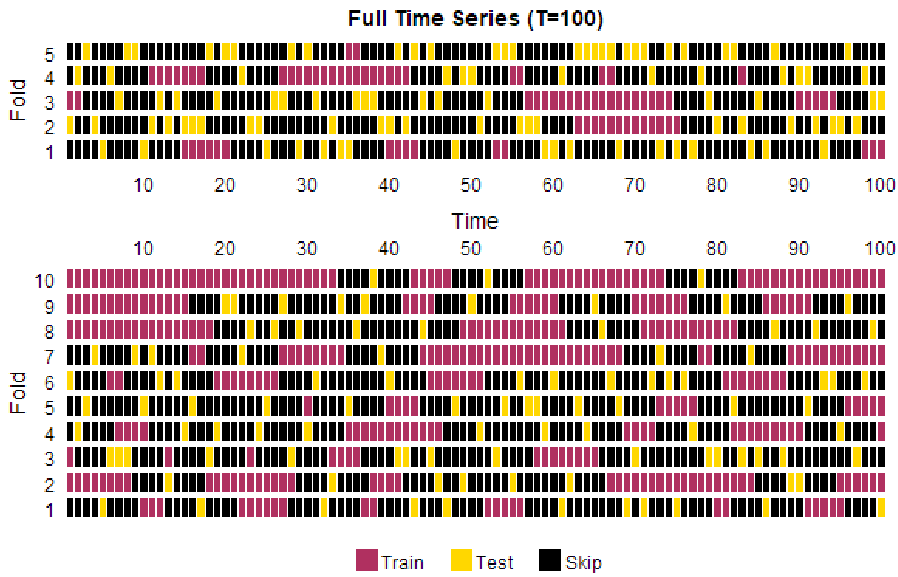
Figure 4.2: Variations of Out-of-Sample Procedures for Model Selection



modified LOOCV by ignoring the d observations before and after each time point in fitting. Similarly, Bergmeir *et al.* (2018) describe and evaluate a non-dependent version CV-K. For $T-m+1 = 100$ and $d = 4$, this modification, illustrated in Figure 4.3, is based off the same random assignment in Figure 4.2. Controlling the number of points available for fitting models is difficult for this modified CV-K even for a low order d. This along with the poor empirical results in Bergmeir *et al.* (2018) removes this approach from consideration.

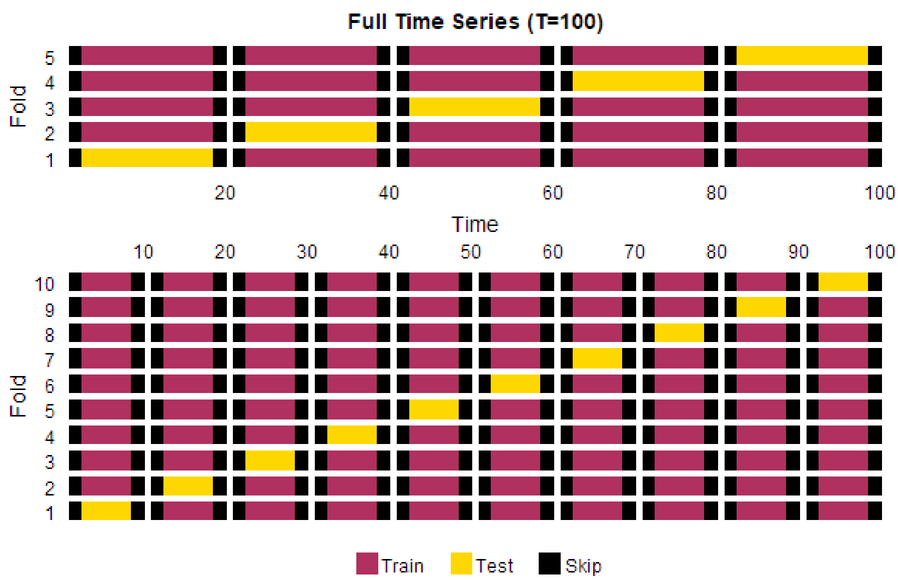
In this paper, blocked variants of CV retain the ordered structure and ensure reasonable sample sizes for model fitting. Racine (2000) alters the CV method of Burman *et al.* (1994) to measure prediction error on blocks of data around each data point for each fold. Bergmeir and Benítez (2012) proposes K-fold blocked CV (BCV-K) where naturally ordered data is evenly split into K sets. For order d, the first and last $\lceil \frac{d}{2} \rceil$ data points are removed from each block to remove dependence, and ordinary CV is performed using the blocks. See Figure 4.4 for for BCV-5 and BCV-10 when

Figure 4.3: Non-Dependent K-fold Cross-Validation for Model Selection for $K = 5$ (top) and $K = 10$ (bottom)



$d = 4$.

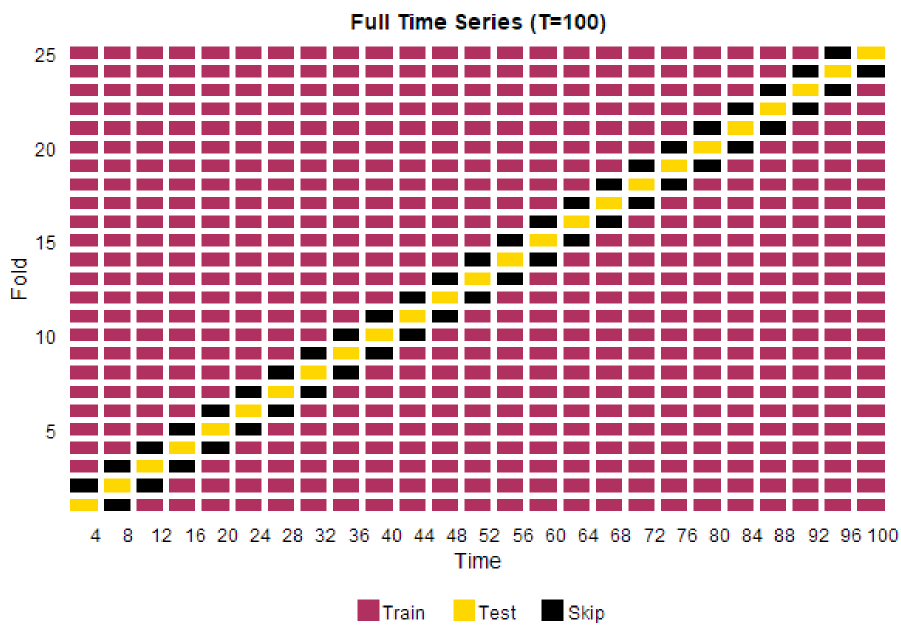
Figure 4.4: Non-Dependent K-Block Cross-Validation for Model Selection for $K = 5$ (top) and $K = 10$ (bottom)



Analogous to LOOCV, leave-one-block-out design (LOBOCV) is a natural extension of BCV-K. This approach is similar to BCV-K* when $K^* = \lfloor \frac{T-m+1}{d} \rfloor$ since the time series is sequentially divided into K^* blocks. Block specific estimates $\widehat{\text{PE}}_K(\lambda)$ or $\widehat{\text{PE}}_K(\lambda, \alpha)$ are evaluated after models are fitted to data in non-adjacent blocks. In LASSO cases, overall BCV prediction error is based on expression in Equation 4.10. Similar expressions are seen for BCV-5 and BCV-10 since all prediction periods are of the same length. This is a key difference to the initial proposed non-dependent CV-K.

$$\widehat{\text{PE}}(\lambda) = \frac{1}{K} \sum_{k=1}^{K} \widehat{\text{PE}}_K(\lambda) \tag{4.10}$$

Figure 4.5: Leave-One-Block-Out Cross-Validation for Model Selection



Literature evaluates these methods on the error between estimated $\widehat{\text{PE}}$ using CV and OOS and true PE from data completely ignored (Bergmeir *et al.*, 2014, 2018). Typical experiments examine this error when the fitted models are known to be mis-specified (Burman *et al.*, 1994; Racine, 2000; Bergmeir *et al.*, 2018). These discussions are not the focus of this paper. Only the performance of these methods in selection of λ and α for ADLASSO and ADENET is evaluated.

4.2.4 *Bayesian Predictive Posterior Projection Method*

Traditional Bayesian Model Selection

Classic Bayesian model selection starts by reparamaterizing $\beta_i^* = \xi_i \beta_i$ where $\xi_i \in \{0,1\}$. For the new vector of parameters $\boldsymbol{\beta}^* = [\beta_1^*, \beta_2^*, \cdots, \beta_{p+q}^*]'$, the set of relevant parameters $\mathcal{V} = \{i : \beta_i^* \neq 0\} = \{i : \xi_i \neq 0\}$. Let $\mathcal{N}_p(\boldsymbol{\mu}, \sigma^2 \boldsymbol{\Sigma}_p)$ represent the p-dimensional multivariate normal distribution and $BERN(\pi)$ represent the *Bernouilli* distribution. If dimension p is not given, assume $p = 1$. The scenario $\boldsymbol{\Sigma}_p = \boldsymbol{I}_p$, where $\boldsymbol{I}_p$ is $p \times p$ identity matrix, implies that $\beta_j \perp \beta_k$ for all $j \neq k$. For the new linear model $\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta}^*$, Kuo and Mallick (1998) suggested the prior $p(\beta_i, \xi_i) = p(\beta_i)p(\xi_i)$ indicating $\beta_i \perp \xi_i$. Later authors suggested $p(\beta_i, \xi_i) = p(\beta_i|\xi_i)p(\xi_i)$ where $p(\beta_i|\xi_i) = (1 - \xi_i)p(\beta_i|\xi_i = 1) + \xi_i p(\beta_i|\xi_i = 0)$ is a mixture prior (Carlin and Chib, 1995). The popular "spike and slab" prior is of this type where the slab $p(\beta_i|\xi_i = 1)$ is uninformative and the spike $p(\beta_i|\xi_i = 0)$ is concentrated around 0 (Mitchell and Beauchamp, 1988; George and McCulloch, 1993; Carlin and Chib, 1995). Common to all methods, $\xi_i \sim BERN(\pi_{i,0})$ where $\pi_{i,0}$ is the prior probability that variable $\beta_i \neq 0$. All subset models are equally likely *a priori* when $\pi_{i,0} = 0.5$. Sampling from $p(\boldsymbol{\beta}, \boldsymbol{\xi}, \sigma^2|\boldsymbol{y}, \boldsymbol{X})$ requires a combination of approaches (Dellaportas *et al.*, 2002), and the posterior mode of $\boldsymbol{\xi}$ indicates the best model. Posterior model probabilities and Bayes factors are used to discriminate between possible sub models. Implementation of Bayesian model averaging is within this umbrella (Raftery *et al.*, 1997; Hoeting *et al.*, 1998, 1999).

Bayesian Regularization

Posterior samplers based on 2-component mixture priors are often slow in exploring high-dimensional spaces. Priors developed from continuous mixing densities achieve similar results without requiring tuning. For example, Andrews and Mallows (1974) presented a hierarchy for the *Laplace* (i.e. $Double - Exponential$) distribution from scale-mixture of $Normals$ using *Exponential* mixing density. The Bayesian LASSO (Park and Casella, 2008; Hans, 2009) uses this hierarchy for $p(\boldsymbol{\beta}|\sigma^2)$ understanding the link between ℓ_1-regularization and posterior modes from *Laplace* priors (Tibshirani, 1996). See O'Hara and Sillanpaa (2009) for a historical look and comparison of adaptive Laplacian priors to discontinuous mixture priors.

Since the introduction of Bayesian LASSO, research in Bayesian regularization methods has exploded over the last ten years. Bayesian methods analogous to ADLASSO (Leng *et al.*, 2014), ENET (Li and Lin, 2010), and ADENET (Stankiewicz, 2015) have been introduced and applied. The prior hierarchies of the aforementioned methods are in a class of "global-local" shrinkage priors (Polson and Scott, 2010). The recently popular Bayesian horseshoe (BHS) prior falls in this class where $half - Cauchy$ priors are used to enforce global sparsity while preventing overshrinking of relevant parameters (Carvalho *et al.*, 2009, 2010). The BHS enjoys the important oracle properties established for ADLASSO and ADENET (Datta and Ghosh, 2015). Bhadra *et al.* (2016) introduced horseshoe+ (BHS^+) which includes an additional layer of local shrinkage improving estimation when $\boldsymbol{\beta}$ is "ultra-sparse". In subset ARMA selection, safely choose p and q large enough to ensure any long lag seasonal effects may be discovered. Overestimation of p and q may introduce many non-seasonal ARMA terms that are equal to 0. For these reasons, BHS and BHS^+ type priors are applied to β_i .

The hierarchical representations of BHS and BHS^+ displayed in Equations 4.11 & 4.12 allow posterior sampling via Gibbs (Makalic and Schmidt, 2016). These hierarchies developed from understanding that $\tau^2|\xi \sim \mathcal{IG}(1/2, 1/\xi)$ and $\xi \sim \mathcal{IG}(1/2, 1/a)$ imply $\tau \sim \mathcal{C}^+(0, a)$ (Wand *et al.*, 2011). Expressions $\mathcal{IG}(a, b)$ and $\mathcal{C}^+(0, a)$ represent

$Inverse-Gamma$ and $half-Cauchy$ distributions, respectively. The latent parameter τ controls overall regularization. Global shrinkage parameter τ can be fixed (Van Der Pas *et al.*, 2014), updated via empirical Bayes (Johnstone *et al.*, 2004), or given a hyperprior (Carvalho *et al.*, 2009, 2010). Prior beliefs on the degree of sparsity in $\boldsymbol{\beta}$ should drive the handling of τ improving regularization (Van Der Pas *et al.*, 2014; Piironen and Vehtari, 2016). The additional latent parameters λ_i fine tune the regularization induced by τ for individual β_i. Heavy-tails of $\mathcal{C}^+(0,1)$ prevent relevant ARMA parameters from being overshrunk to 0.

$$
\begin{aligned}
\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\beta},\sigma^2 &\sim \mathcal{N}_{p+q}(\boldsymbol{X}\boldsymbol{\beta},\sigma^2\boldsymbol{I}_{p+q})\\
\beta_i|\lambda_i^2,\tau^2,\sigma^2 &\sim \mathcal{N}(0,\lambda_i^2\tau^2\sigma^2)\\
\sigma^2 &\sim \sigma^{-2}\mathrm{d}\sigma^2\\
\lambda_i^2|\nu_i &\sim \mathcal{IG}(1/2,1/\nu_i)\\
\tau^2|\xi &\sim \mathcal{IG}(1/2,1/\xi)\\
\nu_1,\cdots,\nu_{p+q} &\sim \mathcal{IG}(1/2,1)\\
\xi &\sim \mathcal{IG}(1/2,1)
\end{aligned}
\tag{4.11}
$$

$$
\begin{aligned}
\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\beta},\sigma^2 &\sim \mathcal{N}_{p+q}(\boldsymbol{X}\boldsymbol{\beta},\sigma^2\boldsymbol{I}_{p+q})\\
\beta_i|\lambda_{1,i}^2,\lambda_{2,i}^2,\tau^2,\sigma^2 &\sim \mathcal{N}(0,\lambda_{1,i}^2\lambda_{2,i}^2\tau^2\sigma^2)\\
\sigma^2 &\sim \sigma^{-2}\mathrm{d}\sigma^2\\
\lambda_{1,i}^2|\nu_{1,i} &\sim \mathcal{IG}(1/2,1/\nu_{1,i})\\
\lambda_{2,i}^2|\nu_{2,i} &\sim \mathcal{IG}(1/2,1/\nu_{2,i})\\
\tau^2|\xi &\sim \mathcal{IG}(1/2,1/\xi)\\
\nu_{1,i},\cdots,\nu_{1,p+q} &\sim \mathcal{IG}(1/2,1)\\
\nu_{2,i},\cdots,\nu_{2,p+q} &\sim \mathcal{IG}(1/2,1)\\
\xi &\sim \mathcal{IG}(1/2,1)
\end{aligned}
\tag{4.12}
$$

Predictive Posterior Projection

Define $\mathcal{V}_F = \{1, 2, \cdots, p+q\}$. For the fully saturated ARMA(p, q) model, let $\hat{\boldsymbol{\beta}}_{HS}(\mathcal{V}_F)$ and $\hat{\boldsymbol{\beta}}_{HS^+}(\mathcal{V}_F)$ correspond to the posterior means of $\boldsymbol{\beta}$ under BHS and BHS$^+$, respectively. Both $\hat{\boldsymbol{\beta}}_{HS}(\mathcal{V}_F)$ and $\hat{\boldsymbol{\beta}}_{HS^+}(\mathcal{V}_F)$ are quality initial estimates of $\boldsymbol{\beta}$ but not sparse since $\hat{\beta}_i \neq 0$ for all i. Obtaining these estimates is analogous to the first stages of AD-LASSO and ADENET. Any $\mathcal{V}_\perp \subset \mathcal{V}_F$ characterizes a particular subset ARMA(p, q) model via indicating the parameters of $\boldsymbol{\beta}$ included. Although the best model $\mathcal{V}^* \subset \mathcal{V}_F$ may differ under BHS and BHS$^+$, the corresponding final subset ARMA(p, q) models are defined $\hat{\boldsymbol{\beta}}_{HS} = \hat{\boldsymbol{\beta}}_{HS}(\mathcal{V}^*)$ and $\hat{\boldsymbol{\beta}}_{HS^+} = \hat{\boldsymbol{\beta}}_{HS^+}(\mathcal{V}^*)$. In this section, a Bayesian inspired algorithm to select $\mathcal{V}^*$ after Bayesian regularization is presented. For simplicity, the outline of this approach is generalized for both BHS and BHS$^+$.

After Bayesian estimation, the full model $\mathcal{V}_F$ represents a viable reference model. The sets $\{\boldsymbol{\beta}^{(s)}(\mathcal{V}_F)\}_{s=1}^S$ and $\{\sigma^{(s)}(\mathcal{V}_F)\}_{s=1}^S$ are the S posterior samples under the reference model. Given a proposed nested model $\mathcal{V}_\perp \subset \mathcal{V}_F$, Goutis and Robert (1998) suggested the Kullback-Leibler (K-L) distance (Kullback and Leibler, 1951) to evaluate discrepancy between $\mathcal{V}_F$ and $\mathcal{V}_\perp$. Classic model selection via AIC is based on K-L information and derivable from a Bayesian perspective (Akaike, 1974, 1985; Burnham and Anderson, 2003, 2004). For a future value $\tilde{y} = y_{T+1}$, the loss in explanatory power from using $\mathcal{V}_\perp$ instead of $\mathcal{V}_F$ is assessed by the K-L distance between posterior predictive distributions listed in Equation 4.13. If the discrepancy between $p(\tilde{y}|\boldsymbol{y}, \boldsymbol{X}, V_F)$ and $p(\tilde{y}|\boldsymbol{y}, \boldsymbol{X}, V_\perp)$ is small, the more parsimonious $\mathcal{V}_\perp$ is favored. The foundation of this concept is provided in Dupuis and Robert (2003); Nott and Leng (2010); Vehtari and Ojanen (2012); Piironen and Vehtari (2015a, 2017).

$$\begin{aligned} p(\tilde{y}|\boldsymbol{y}, \boldsymbol{X}, V_F) &= \int\int p(\tilde{y}|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{\beta}_F, \sigma_\perp, V_F) p(\boldsymbol{\beta}_F, \sigma_F|\boldsymbol{y}, \boldsymbol{X}, V_F)\ \mathrm{d}\boldsymbol{\beta}_F\ \mathrm{d}\sigma_F \\ p(\tilde{y}|\boldsymbol{y}, \boldsymbol{X}, V_\perp) &= \int\int p(\tilde{y}|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{\beta}_\perp, \sigma_\perp, V_\perp) p(\boldsymbol{\beta}_\perp, \sigma_\perp|\boldsymbol{y}, \boldsymbol{X}, V_\perp)\ \mathrm{d}\boldsymbol{\beta}_\perp\ \mathrm{d}\sigma_\perp \end{aligned} \tag{4.13}$$

For Gaussian linear models, S posterior samples $\{\boldsymbol{\beta}^{(s)}(\mathcal{V}_\perp), \sigma^{(s)}(\mathcal{V}_\perp)\}_{s=1}^{S}$ for a nested submodel $\mathcal{V}_\perp$ are quickly obtained via Equation 4.14 (Piironen and Vehtari, 2015a). The matrix $\boldsymbol{X}_\perp$ contains the columns of the reference model matrix $\boldsymbol{X}$ corresponding to $\mathcal{V}_\perp$. Essentially, S samples from the posterior distribution of a submodel are obtained through projecting the fitted values from the full model onto a smaller parameter space.

$$\begin{aligned}
\boldsymbol{\beta}^{(s)}(\mathcal{V}_\perp) &= (\boldsymbol{X}_\perp'\boldsymbol{X}_\perp)^{-1}\boldsymbol{X}_\perp'\boldsymbol{X}\boldsymbol{\beta}^{(s)}(\mathcal{V}_F) \\
\sigma^{(s)}(\mathcal{V}_\perp) &= \sqrt{[\sigma^{(s)}(\mathcal{V}_\perp)]^2 + \frac{||\boldsymbol{X}\boldsymbol{\beta}^{(s)}(\mathcal{V}_F) - \boldsymbol{X}_\perp\boldsymbol{\beta}^{(s)}(\mathcal{V}_\perp)||^2}{T-m+1}}
\end{aligned} \tag{4.14}$$

The overall discrepancy between the full ARMA(p,q) model and a subset ARMA(p,q) model is measured in Equation 4.15. The expected KL divergence is estimated between the predictive distribution of the $\mathcal{V}_F$ and $\mathcal{V}_\perp$.

$$D(\mathcal{V}_F||\mathcal{V}_\perp) = \frac{1}{S}\sum_{s=1}^{S}\log\left(\frac{\sigma^{(s)}(\mathcal{V}_\perp)}{\sigma^{(s)}(\mathcal{V}_F)}\right) \tag{4.15}$$

Measuring the discrepancy in Equation 4.15 for all $2^{p+q}-1$ subset ARMA models is impractical; therefore, the forward stepwise algorithm of Peltola *et al.* (2014); Piironen and Vehtari (2015a). If $\mathcal{V}_0$ represents the intercept-only model (empty model when intercept is unnecessary), $D(\mathcal{V}_F||\mathcal{V}_0)$ is the maximum discrepancy for all possible $\mathcal{V}_\perp$ (Dupuis and Robert, 2003). Next, the best subset ARMA model with one additional parameter by Equation 4.16 is selected.

$$\mathcal{V}_1 = \underset{\{\mathcal{V}_0\subset\mathcal{V}_\perp\subset\mathcal{V}_F:|\mathcal{V}_\perp|=1\}}{\text{argmin}} D(\mathcal{V}_F||\mathcal{V}_\perp) \tag{4.16}$$

Moving forward to models with two additional parameters, the best subset ARMA model with 2 ARMA parameters is identified by Equation 4.17.

$$\mathcal{V}_2 = \underset{\{\mathcal{V}_1\subset\mathcal{V}_\perp\subset\mathcal{V}_F:|\mathcal{V}_\perp|=2\}}{\text{argmin}} D(\mathcal{V}_F||\mathcal{V}_\perp) \tag{4.17}$$

In general, the best subset ARMA model with m coefficients, represented by $\mathcal{V}_m$ where $\mathcal{V}_{m-1} \subset \mathcal{V}_m \subset \mathcal{V}_{m+1}$, is based on Equation 4.18. Piironen and Vehtari (2015b) present a helpful tutorial of this approach with **R** code and application.

$$\mathcal{V}_m = \underset{\{\mathcal{V}_{m-1} \subset \mathcal{V}_\perp \subset \mathcal{V}_F : |\mathcal{V}_\perp| = i\}}{\operatorname{argmin}} D(\mathcal{V}_F || \mathcal{V}_\perp) \tag{4.18}$$

The forward stepwise algorithm leads to the following sequence of $p+q$ nested models: $\mathcal{V}_1 \subset \cdots \subset \mathcal{V}_F$. Because of the additive property of $D(\cdot||\cdot)$, Dupuis and Robert (2003) recommend selecting $\mathcal{V}^* \in \{\mathcal{V}_1, \cdots, \mathcal{V}_F\}$ based on the relative explanatory power (e) defined in Equation 4.19.

$$e(\mathcal{V}_m) = 1 - \frac{D(\mathcal{V}||\mathcal{V}_m)}{D(\mathcal{V}||\mathcal{V}_0)} \tag{4.19}$$

This additive property ensures $0 = e(\mathcal{V}_0) < e(\mathcal{V}_m) < e(\mathcal{V}_F) = 1$ for any $m \in \{1, \cdots, p+q-1\}$. For an acceptable explanatory power e^*, $\mathcal{V}^* = \mathcal{V}_{m^*}$ is selected based on m^* defined in Equation 4.20. Piironen and Vehtari (2015b) suggest $e^* \geq 0.90$. In the following empirical studies, model selection sensitivity for $e^* \in \{0.9, 0.95, 0.98\}$ is examined.

$$m^* = \min\{m : e(\mathcal{V}_m) > e^*\} \tag{4.20}$$

In an application to biomarker identification for cardiovascular event risk, Peltola *et al.* (2014) based model selection from estimating predictive performance via 10-fold CV. Combining Bayesian techniques with multi-fold CV is time consuming, and the validity of general CV in time series analysis is questionable. Similar to the OOS scheme illustrated in Figure 4.2, a final portion of the data is intentionally withheld for forecast evaluation. The PE for each nested model is estimated using MSFE according to Equation 4.21. Although $\tau \times 100\%$ of the data is lost in estimation, the final model $\mathcal{V}^*$ must demonstrate superior OOS forecasting performance to the other

$p+q$ candidates.

$$\widehat{\text{PE}}(\mathcal{V}) = \frac{1}{\tau T} \sum_{t \in \text{TEST}} \left(y_t - \boldsymbol{x}_t' \hat{\boldsymbol{\beta}}_{HS}(\mathcal{V}) \right)^2 \tag{4.21}$$

4.2.5 Summary of Methods

In this section, OOS and CV techniques are included for tuning parameter selection. Table 4.1 lists the gamut of options discussed and tested in Monte Carlo simulations. For future reference, the 2-stage ADLASSO and ADENET variants are denoted AL_m and AE_m where $m \in \{1, 2, \cdots, 11\}$ identifies the method.

Table 4.1: Summary of ADLASSO and ADENET Variants

Method (m)	Initial Weights (Stage 1)	Final Model (Stage 2)
1	AIC	AIC
2	AIC	BIC
3	BIC	BIC
4	OOS	
5	depOOS	
6	CV-5	
7	CV-10	
8	LOOCV	
9	BCV-5	
10	BCV-10	
11	LOBOCV	

Additional methods considered are from a Bayesian perspective. Initial posterior sampling is based off either BHS or BHS^+ priors. Table 4.2 lists different options for final model selection in the predictive posterior projection method. For future reference, all Bayesian options are abbreviated BHS_m or BHS^+_m where $m \in \{1, 2, \cdots, 4\}$.

Table 4.2: Summary of BHS and BHS$^+$ Variants

Method (m)	Final Model Selection
1	$e(\cdot) > 0.90$
2	$e(\cdot) > 0.95$
3	$e(\cdot) > 0.98$
4	OOS

4.3 Monte Carlo Simulations

4.3.1 General Simulation Specifications

Multiple Monte Carlo studies are performed to evaluate and compare ADLASSO, ADENET, BHS, and BHS$^+$ on subset ARMA selection. Consider the three time series $\{y_{1,t}\}$, $\{y_{2,t}\}$, and $\{y_{3,t}\}$ generated by the Gaussian ARMA processes expressed in Equations 4.22, 4.23, and 4.24 and abbreviated Models I, II, and III, respectively.

$$y_{1,t} = 0.8y_{1,t-1} + 0.7y_{1,t-6} - 0.56y_{1,t-7} + \epsilon_{1,t} \tag{4.22}$$

$$\begin{aligned} y_{2,t} &= 0.8y_{2,t-1} + 0.7y_{2,t-6} - 0.56y_{2,t-7} \\ &\quad + 0.8\epsilon_{2,t-1} + 0.7\epsilon_{2,t-6} + 0.56\epsilon_{2,t-7} + \epsilon_{2,t} \end{aligned} \tag{4.23}$$

$$y_{3,t} = 0.8\epsilon_{3,t-1} + 0.7\epsilon_{3,t-6} + 0.56\epsilon_{3,t-7} + \epsilon_{3,t} \tag{4.24}$$

The errors $\{\epsilon_{1,t}\}$, $\{\epsilon_{2,t}\}$, and $\{\epsilon_{3,t}\}$ are i.i.d. Gaussian processes with $\mu = 0$ and $\sigma = 1$. Models I-III are algebraically equivalent to the first three SARMA$(p, q) \times (P, Q)_6$ models found in Chen and Chan (2011), and similarly, samples of length $T \in \{120, 240, 360\}$ are generated.

All three DGPs are subset ARMA$(7, 7)$ models. Assuming the maximum ARMA orders are $p = q = 14$, all variants of ADLASSO, ADENET, BHS and BHS$^+$ listed in Tables 4.1 and 4.2 are used to fit subset ARMA(p, q) models. Methods are evaluated using 4 model selection accuracy statistics (C, I, $-$, $+$) across 200 replications.

Statistics C and I are relative frequencies of final models that contain all relevant variables and identify the true model, respectively. The statistic $-$ represents the average false negative rate (probability of missing a relevant ARMA parameter), and the statistic $+$ represents the average false positive rate (probability of selecting an irrelevant ARMA parameter).

All experiments are conducted in **R** (R Core Team, 2017) on an Intel Xeon CPU E5-2697 v3 @ 2.60 GHz server with 132GB of RAM and 56 cores maintained at Arizona State University. Popular **R** packages **doParallel** and **foreach** are used for parallelization of replications. Replications of Models I-III are simulated according to their SARMA$(p, q) \times (P, Q)_s$ equivalents using the **forecast** package (Hyndman and Khandakar, 2008). Additional **R** packages required are introduced and cited when necessary.

4.3.2 Sensitivity: Order Selection of Long AR(p') Process

The proxy innovations $\{\hat{\epsilon}_{k,t}\}$ are obtained from long AR(p') models where $p' = 10\log_{10}(T)$. An initial AR(p') model is estimated using Yule-Walker equations for ADLASSO and ADENET with the **ar()** function in base **R**. For BHS and BHS$^+$, Bayesian linear regression (Gelman *et al.*, 2014, pg.354) can be conducted with **MCMCregress()** (Martin *et al.*, 2011). Using basic Gibbs sampling, the posterior mean from 2000 posterior samples after a burn-in of 10000 and a thinning of 10 is used to estimate the initial AR(p').

Chen and Chan (2011) consider $10\log_{10}(T)$ as a maximum and select p' based on AIC. The advantage here is in the reduction of m when a shorter AR(p') process is selected; therefore, more data is retained for the ADLASSO or ADENET stages. In simulations, a deterioration in overall subset selection is noticed under this approach compared to fixing p'. Due to this disagreement, these two ideologies are compared in simulation. Only AL_1, AL_2, and AL_3 methods are considered since these were introduced in Chen and Chan (2011). Tables 4.3, 4.4, and 4.5 compare the model selection results for Models I-III. The full sensitivity analysis is based on 500 replications.

Table 4.3: Effect of Using AIC to Select p' on ADLASSO Subset ARMA$(14, 14)$ Estimation of Model I Based on 500 Replications

		Long AR(p')				Short AR(p')			
	T	C	I	$-$	$+$	C	I	$-$	$+$
	120	0.19	0.01	0.36	0.28	0.02	0.00	0.49	0.28
AL_1	240	0.40	0.05	0.24	0.27	0.02	0.00	0.43	0.34
	360	0.46	0.07	0.21	0.26	0.03	0.00	0.38	0.36
	120	0.16	0.04	0.40	0.17	0.02	0.00	0.53	0.18
AL_2	240	0.36	0.13	0.27	0.18	0.01	0.00	0.47	0.24
	360	0.45	0.17	0.22	0.18	0.03	0.01	0.41	0.27
	120	0.05	0.01	0.44	0.12	0.01	0.00	0.48	0.14
AL_3	240	0.15	0.05	0.35	0.17	0.01	0.00	0.44	0.20
	360	0.21	0.09	0.31	0.20	0.01	0.01	0.39	0.24

Contrary to Chen and Chan (2011), better performance was observed when p' is fixed versus selection of p' through minimization of AIC. This is especially apparent for Model I where selection of p' systematically results in missing relevant parameters. All future results using both ADLASSO and ADENET begin with fixing p' to estimate innovations $\{\hat{\epsilon}_t\}$ for $\boldsymbol{X}$. Likewise, model selection at this initial step is not considered for Bayesian-based methods either.

4.3.3 Model Selection Results for All Methods

Now that a guideline for estimating the innovations has been established, all ADLASSO, ADENET, BHS, and BHS^+ methods are evaluated in simulation. Due

Table 4.4: Effect of Using AIC to Select p' on ADLASSO Subset ARMA$(14, 14)$ Estimation of Model II Based on 500 Replications

		Long AR(p')				Short AR(p')			
	T	C	I	$-$	$+$	C	I	$-$	$+$
	120	0.09	0.00	0.24	0.39	0.00	0.00	0.31	0.34
AL_1	240	0.23	0.00	0.18	0.39	0.09	0.00	0.22	0.37
	360	0.30	0.01	0.14	0.37	0.20	0.00	0.17	0.37
	120	0.06	0.00	0.27	0.30	0.00	0.00	0.33	0.28
AL_2	240	0.17	0.01	0.20	0.32	0.07	0.01	0.23	0.33
	360	0.26	0.02	0.16	0.31	0.19	0.01	0.18	0.33
	120	0.03	0.00	0.30	0.25	0.00	0.00	0.33	0.22
AL_3	240	0.10	0.00	0.23	0.30	0.07	0.01	0.24	0.29
	360	0.20	0.01	0.17	0.31	0.15	0.02	0.19	0.31

Table 4.5: Effect of Using AIC to Select p' on ADLASSO Subset ARMA$(14, 14)$ Estimation of Model III Based on 500 Replications

		Long AR(p')				Short AR(p')			
	T	C	I	$-$	$+$	C	I	$-$	$+$
	120	0.34	0.00	0.32	0.28	0.20	0.00	0.35	0.23
AL_1	240	0.42	0.01	0.26	0.32	0.39	0.02	0.26	0.28
	360	0.44	0.03	0.25	0.34	0.47	0.04	0.23	0.31
	120	0.24	0.03	0.41	0.15	0.14	0.02	0.41	0.13
AL_2	240	0.37	0.08	0.32	0.17	0.33	0.05	0.34	0.16
	360	0.40	0.08	0.31	0.19	0.43	0.10	0.29	0.17
	120	0.27	0.04	0.36	0.10	0.17	0.02	0.39	0.10
AL_3	240	0.64	0.14	0.15	0.11	0.59	0.10	0.18	0.11
	360	0.76	0.20	0.11	0.11	0.77	0.23	0.10	0.09

to the large variety of methods considered, experiments are based on 200 replications for Models I-III. For brevity, results are not reported for $T = 240$.

The **glmnet** package handles LASSO and ENET estimation, performing CV-K for optimal selection of $\lambda^* \in \mathcal{L}$ (Friedman *et al.*, 2010). Set $\mathcal{L}$ is automatically determined in **glmnet**, and set $\mathcal{A} = \{0, 0.1, \cdots, 0.9, 1\}$ is considered for α. For ADENET, a grid search identifies the optimal λ^*_α for each $\alpha \in \mathcal{A}$. Final selection of the tuning parameter pair (α^*, λ^*) is based on $\min\{\widehat{\text{PE}}(\alpha, \lambda^*_\alpha) : \alpha \in \mathcal{A}\}$. For methods AL_4, AL_5, AE_4, and AE_5, the percent of data removed for OOS forecasting $\tau = 0.20$. Methods AL_m for $m \in \{5, 9, 10, 11\}$ are based on the maximum ARMA dependence $d = \max\{14, 14\} = 14$ and are manually programmed.

Fast BHS and BHS^+ estimation is a product of hierarchies presented in Equations 4.11 and 4.12. The **bayesreg** package samples from the full posterior distributions for both $\boldsymbol{\beta}$ and σ^2 via Gibbs (Schmidt and Makalic, 2016). Through visual inspection of MCMC chains, a burn-in period of 10000 is adequate for convergence. Only retaining every tenth sample, $S = 2000$ posterior samples are obtained from the fully saturated ARMA(14, 14). Likewise, estimation of subset ARMA(14, 14) models is based on $S = 2000$ posterior samples obtained through projection. Consistent with ADLASSO and ADENET, $\tau = 0.20$ for BHS_4 and BHS^+_4.

Tables 4.6, 4.7, and 4.8 display the model selection results applying all AL_m and AE_m to Models I-III, respectively. The different algorithms for selecting tuning parameters are grouped according to the division in Table 4.1. ADLASSO and ADENET are paired to evaluate the effectiveness of the additional mixing parameter α. Across all m, ADENET consistently outperforms ADLASSO in discovering relevant parameters. Combining OOS or depOOS with ADENET (AE_4 & AE_5) further increases C; however, none of the replications identified the true model ($I = 0.00$). This demonstrates the cost to decrease $-$ at the expense of increasing $+$. An oracle procedure implies $I \to 1$ as $T \to \infty$. None of these methods perform adequately for $T = 120$. When T increases, there is a natural increase in C and I and decrease in $-$ and $+$. In Model II, this effect is witnessed, yet all methods rarely identify the true model as indicated by $I \approx 0$. In Model I and Model III, many of the ADLASSO methods lead to similar C and I. Using AIC/BIC or CV-K for ADENET drastically improves both C and I when compared to BCV-K or OOS. Statistic I is typically higher for ADLASSO, but combining ADENET with CV-K (AE_6, AE_7, & AE_8) is competitive.

Chen and Chan (2011) explore the efficacy of AIC/BIC-based ADLASSO methods when BIC is always used for LASSO stage 1 followed by AIC or BIC in the adaptive stage 2. Method AL_3 is the only common method whereas AL_1 and AL_2 begin with AIC in the weight estimation. The full AIC method AL_1 rarely identifies the true model. Under Model I and $T = 360$, AL_2 slightly outperforms AL_3 based on I but selects all significant parameters more than double of the time. Under Model III and $T = 360$, the full BIC method AL_3 not only outperforms AL_2 but also every other ADLASSO method based on the combination of low $-$ and $+$ error rates. This result is mimicked for ADENET methods where AE_3 sees similar performance.

Modifications for temporal dependence d are introduced in methods AL_4 versus AL_5 and AE_4 versus AE_5. OOS ($m = 4$) and depOOS ($m = 5$) produce similar results, which is expected considering the minor difference in training periods. Accounting for the assumed dependence $d = 14$ in ADLASSO does not impact performance. As previously implied, neither CV-K nor BCV-K methods worked well on Model II; however, false positive rates $+$ are much lower for BCV-K. The ADENET methods are more sensitive to the way the tuning parameters are selected. ADENET CV methods AE_6, AE_7, and AE_8 select the true model far more frequently than AE_9, AE_{10}, and AE_{11}. To be sure final subset selection contains all relevant parameters, BCV methods indicate a larger C but negatively impact the false positive rate $+$.

Table 4.6: ADLASSO and ADENET Subset ARMA(14, 14) Results from 200 Replications of Model I

		AL$_m$				AE$_m$			
m	T	C	I	$-$	$+$	C	I	$-$	$+$
1	120	0.19	0.01	0.35	0.29	0.19	0.01	0.36	0.28
1	360	0.50	0.08	0.20	0.24	0.54	0.08	0.17	0.25
2	120	0.10	0.04	0.42	0.18	0.15	0.04	0.40	0.17
2	360	0.42	0.16	0.23	0.19	0.50	0.20	0.19	0.15
3	120	0.04	0.00	0.44	0.12	0.06	0.01	0.42	0.13
3	360	0.20	0.10	0.32	0.19	0.20	0.10	0.30	0.19
4	120	0.13	0.00	0.42	0.16	0.36	0.00	0.24	0.55
4	360	0.28	0.12	0.30	0.14	0.70	0.00	0.10	0.67
5	120	0.14	0.02	0.42	0.20	0.41	0.00	0.22	0.57
5	360	0.24	0.12	0.32	0.15	0.66	0.00	0.12	0.66
6	120	0.08	0.02	0.44	0.11	0.12	0.01	0.40	0.22
6	360	0.36	0.16	0.27	0.16	0.52	0.18	0.19	0.17
7	120	0.09	0.02	0.44	0.13	0.15	0.04	0.41	0.17
7	360	0.44	0.16	0.23	0.15	0.54	0.16	0.18	0.17
8	120	0.08	0.02	0.46	0.12	0.12	0.02	0.40	0.19
8	360	0.42	0.21	0.24	0.15	0.53	0.17	0.19	0.16
9	120	0.06	0.00	0.54	0.06	0.28	0.00	0.36	0.33
9	360	0.44	0.20	0.23	0.12	0.60	0.04	0.15	0.27
10	120	0.12	0.02	0.41	0.18	0.23	0.00	0.32	0.34
10	360	0.36	0.16	0.26	0.13	0.54	0.04	0.17	0.26
11	120	0.14	0.02	0.40	0.13	0.22	0.00	0.33	0.31
11	360	0.46	0.24	0.23	0.11	0.62	0.05	0.14	0.29

Table 4.7: ADLASSO and ADENET Subset ARMA$(14, 14)$ Results from 200 Replications of Model II

		AL_m				AE_m			
m	T	C	I	$-$	$+$	C	I	$-$	$+$
1	120	0.13	0.00	0.23	0.40	0.13	0.00	0.23	0.40
1	360	0.26	0.01	0.15	0.38	0.42	0.00	0.13	0.38
2	120	0.06	0.00	0.28	0.29	0.06	0.00	0.28	0.33
2	360	0.26	0.02	0.16	0.32	0.36	0.02	0.14	0.33
3	120	0.02	0.00	0.30	0.25	0.02	0.00	0.30	0.24
3	360	0.20	0.01	0.18	0.30	0.23	0.02	0.17	0.31
4	120	0.02	0.00	0.39	0.14	0.30	0.00	0.15	0.61
4	360	0.06	0.03	0.28	0.08	0.70	0.00	0.05	0.78
5	120	0.01	0.00	0.38	0.17	0.32	0.00	0.17	0.60
5	360	0.06	0.04	0.27	0.08	0.67	0.00	0.06	0.78
6	120	0.02	0.00	0.29	0.25	0.04	0.00	0.27	0.27
6	360	0.18	0.05	0.18	0.25	0.30	0.00	0.14	0.32
7	120	0.04	0.00	0.28	0.25	0.07	0.00	0.27	0.27
7	360	0.16	0.04	0.19	0.24	0.26	0.01	0.16	0.30
8	120	0.03	0.00	0.30	0.24	0.04	0.00	0.27	0.27
8	360	0.16	0.02	0.20	0.26	0.26	0.01	0.16	0.31
9	120	0.02	0.00	0.46	0.06	0.08	0.00	0.34	0.19
9	360	0.06	0.04	0.29	0.06	0.34	0.00	0.14	0.36
10	120	0.00	0.00	0.37	0.13	0.10	0.00	0.24	0.38
10	360	0.08	0.06	0.26	0.07	0.26	0.00	0.16	0.37
11	120	0.00	0.00	0.37	0.13	0.06	0.00	0.27	0.32
11	360	0.04	0.03	0.27	0.06	0.31	0.00	0.14	0.38

Table 4.8: ADLASSO and ADENET Subset ARMA$(14, 14)$ Results from 200 Replications of Model III

		AL$_m$				AE$_m$			
m	T	C	I	$-$	$+$	C	I	$-$	$+$
1	120	0.32	0.00	0.33	0.27	0.28	0.00	0.34	0.29
1	360	0.45	0.03	0.26	0.33	0.47	0.02	0.21	0.34
2	120	0.24	0.02	0.40	0.16	0.20	0.04	0.42	0.16
2	360	0.36	0.04	0.33	0.21	0.40	0.07	0.27	0.18
3	120	0.26	0.05	0.37	0.10	0.24	0.04	0.38	0.11
3	360	0.78	0.19	0.10	0.11	0.78	0.18	0.10	0.11
4	120	0.26	0.02	0.37	0.25	0.64	0.00	0.14	0.52
4	360	0.49	0.18	0.24	0.21	0.86	0.00	0.05	0.61
5	120	0.21	0.03	0.43	0.21	0.64	0.00	0.15	0.51
5	360	0.52	0.20	0.23	0.20	0.90	0.00	0.03	0.61
6	120	0.19	0.06	0.40	0.11	0.32	0.06	0.32	0.18
6	360	0.46	0.22	0.25	0.12	0.48	0.12	0.22	0.18
7	120	0.16	0.06	0.42	0.10	0.28	0.07	0.34	0.17
7	360	0.44	0.23	0.28	0.12	0.50	0.14	0.23	0.16
8	120	0.18	0.04	0.43	0.10	0.30	0.06	0.34	0.19
8	360	0.48	0.22	0.24	0.10	0.48	0.12	0.23	0.17
9	120	0.12	0.03	0.64	0.06	0.52	0.01	0.27	0.46
9	360	0.55	0.14	0.20	0.15	0.60	0.04	0.16	0.29
10	120	0.28	0.03	0.36	0.18	0.47	0.01	0.23	0.36
10	360	0.56	0.21	0.18	0.14	0.64	0.02	0.14	0.25
11	120	0.26	0.03	0.35	0.14	0.46	0.02	0.23	0.35
11	360	0.42	0.08	0.27	0.19	0.48	0.01	0.22	0.34

Results for BHS_m and BHS_m^+ for $m \in \{1, 2, \cdots, 4\}$ are displayed in Tables 4.9, 4.10, and 4.11. These tables are subdivided according to Table 4.2. As it pertains to Models I-III, results for BHS_m and BHS_m^+ are close to identical; therefore, performance only regarding BHS_m is discussed. Subset selection from methods BHS_1, BHS_2, and BHS_3 based off relative efficiency $e(\cdot)$ is effected by the threshold e^*. Increasing e^* increases C and decreases $-$ due to the nested nature of models obtained via forward stepwise algorithm. Jointly considering $(I, +)$, $e^* = 0.95$ (BHS_2) yields the best results. Specifically for Models II and III, the results for $e^* = 0.95$ are not superb. Setting $e^* = 0.99$ Selection of e^* is more a preference-based decision than scientific decision. The advantage of BHS_4 is that final model selection is based on actual OOS forecasting rather than an arbitrary threshold. For shorter time series ($T = 120$), BHS_4 does not outperform BHS_2, but when $T = 360$, BHS_4 starts to be competitive. Modifications can be made to τ to ensure enough data remains for model fitting, but for right now, the recommendation is to reserve BHS_4 for longer series.

Table 4.9: BHS and BHS^+ Subset ARMA$(14, 14)$ Results from 200 Replications of Model I

		BHS_m				BHS_m^+			
m	T	C	I	$-$	$+$	C	I	$-$	$+$
1	120	0.66	0.34	0.18	0.06	0.64	0.38	0.20	0.06
1	360	0.70	0.60	0.18	0.04	0.66	0.57	0.21	0.04
2	120	0.88	0.05	0.06	0.15	0.86	0.14	0.06	0.11
2	360	0.88	0.62	0.08	0.04	0.88	0.60	0.07	0.04
3	120	0.95	0.00	0.02	0.36	0.96	0.00	0.02	0.30
3	360	0.92	0.42	0.05	0.06	0.90	0.49	0.06	0.05
4	120	0.63	0.15	0.20	0.13	0.65	0.14	0.20	0.14
4	360	0.92	0.30	0.04	0.09	0.92	0.32	0.05	0.08

The four statistics $(C, I, -, +)$ quantify subset selection differently, and identifying a best method is difficult. For Models I and II, all Bayesian methods universally

Table 4.10: BHS and BHS^+ Subset ARMA$(14, 14)$ Results from 200 Replications of Model II

		BHS_m				BHS_m^+			
m	T	C	I	$-$	$+$	C	I	$-$	$+$
1	120	0.08	0.01	0.32	0.11	0.06	0.02	0.32	0.11
1	360	0.13	0.02	0.24	0.09	0.12	0.03	0.25	0.09
2	120	0.33	0.02	0.18	0.17	0.31	0.01	0.19	0.16
2	360	0.57	0.18	0.10	0.09	0.51	0.16	0.12	0.10
3	120	0.56	0.00	0.09	0.33	0.55	0.00	0.10	0.28
3	360	0.89	0.05	0.03	0.15	0.88	0.08	0.04	0.14
4	120	0.32	0.00	0.24	0.25	0.28	0.00	0.26	0.21
4	360	0.84	0.09	0.05	0.17	0.87	0.10	0.04	0.15

Table 4.11: BHS and BHS^+ Subset ARMA$(14, 14)$ Results from 200 Replications of Model III

		BHS_m				BHS_m^+			
m	T	C	I	$-$	$+$	C	I	$-$	$+$
1	120	0.25	0.00	0.36	0.22	0.21	0.00	0.39	0.20
1	360	0.26	0.03	0.46	0.14	0.26	0.04	0.46	0.14
2	120	0.38	0.00	0.26	0.38	0.36	0.00	0.27	0.33
2	360	0.40	0.00	0.34	0.22	0.38	0.00	0.36	0.21
3	120	0.53	0.00	0.19	0.56	0.43	0.00	0.22	0.51
3	360	0.62	0.00	0.18	0.39	0.59	0.00	0.20	0.34
4	120	0.16	0.00	0.54	0.26	0.12	0.00	0.54	0.22
4	360	0.40	0.02	0.35	0.25	0.35	0.02	0.36	0.24

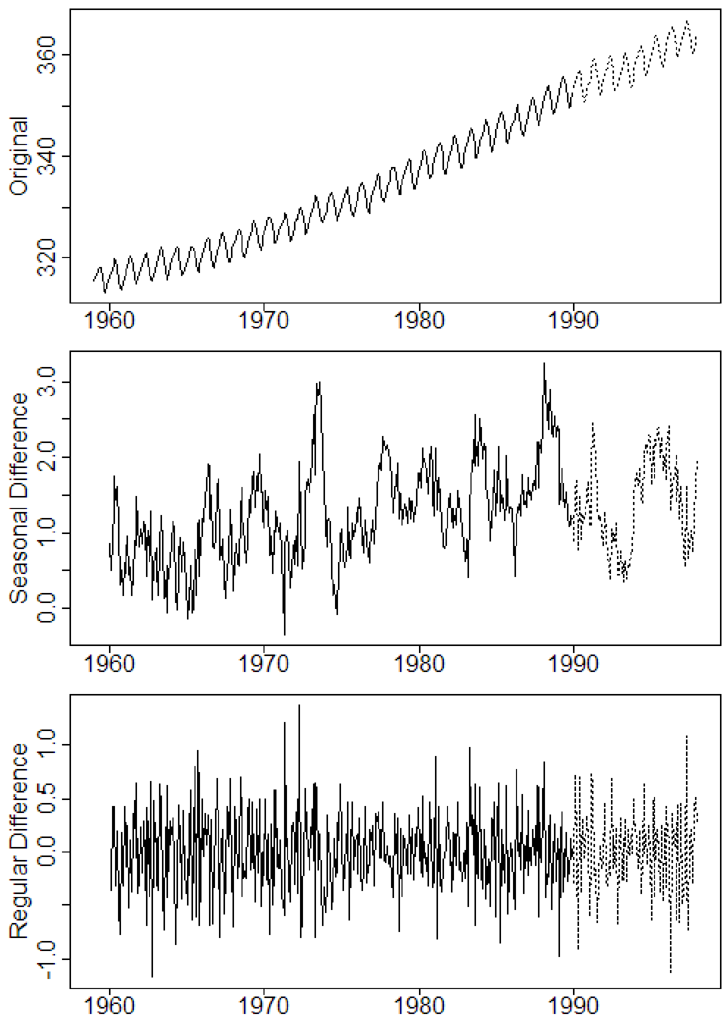
outperform ADLASSO and ADENET regarding C and I. ADLASSO and ADENET performed best under Model III where BHS_m and BHS_m^+ rarely identified the true model. Model I contains only AR terms and Model III contains only MA terms. Constricting estimation of subset ARMA$(14, 0)$ for Model I and subset ARMA$(0, 14)$ for Model III dramatically improves all selection statistics $(C, I, -, +)$. From all experiments, the best results are seen when BHS_m and BHS_m^+ are used for Model I. Because AR(p) models can approximate MA(q) processes and are easier to handle computationally, Bayesian projection approaches using relative efficiency threshold $e^* \in [0.9, 0.95]$ are recommended, and if T is large enough, consider OOS.

4.4 Application

Carbon dioxide (CO_2) levels are constantly measured at atmospheric monitoring observatories around the world to track climate change. The **datasets** package in **R** (R Core Team, 2017) contains a monthly time series of CO_2 levels for January 1959 to December 1997 measured in Mauna Loa, Hawaii, United States. The **TSA** package in **R** (Chan and Ripley, 2012) contains a similar but shorter series from Alert, Nunavut, Canada, from January 1994 to December 2004. Let $\{x_{1,t}\}$ represent the Mauna Loa data, and $\{x_{2,t}\}$ represent the Alert data. Both $\{x_{1,t}\}$ and $\{x_{2,t}\}$ are nonstationary in mean and cyclical with seasonal periodicity $s = 12$. The latter series $\{x_{2,t}\}$ serves as a primary textbook example to demonstrate the selection, fitting, and forecasting of multiplicative seasonal SARMA$(p, q) \times (P, Q)_{12}$ (Cryer and Chan, 2008, pp. 227-245). Following the examples provided in Cryer and Chan (2008); Chen and Chan (2011), subset ARMA(p, q) procedures are applied after seasonal and regular differencing for both locations.

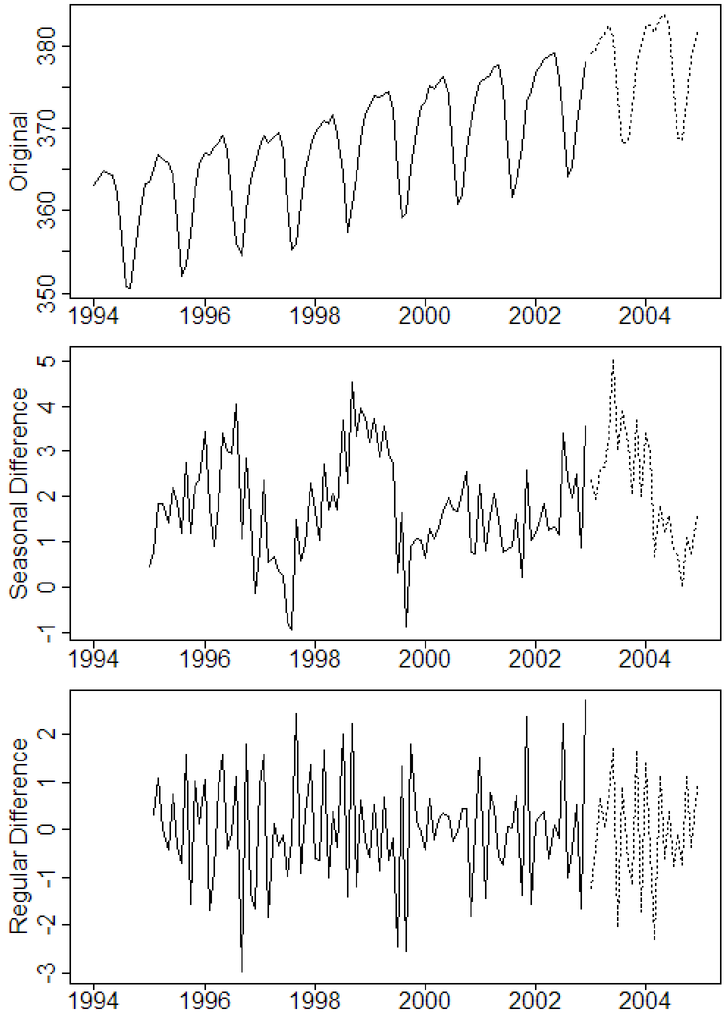
Define $y_{k,t} = \Delta_1 \Delta_{12} x_{k,t}$ for $k \in \{1, 2\}$ where Δ_s is the difference operator such that $\Delta_s y_t = y_t - y_{t-s}$. Using variations of adaptive lasso, adaptive elastic net, and projection model selection, subset ARMA(14, 14) models are fitted to $\{y_{1,t} : t = 1, 2, \cdots, 372\}$ corresponding to data prior to 1990 and $\{y_{2,t} : t = 1, 2, \cdots, 108\}$ corresponding to data prior to 2003. Remaining portions $\{y_{1,t} : t = 373, 374, \cdots, 468\}$ and $\{y_{2,t} : t = 109, 110, \cdots, 132\}$ are intentionally preserved for one-step ahead forecasting. Figures 4.6 and 4.7 illustrate the division of the data into fitting and forecasting periods, as well as, the progression of seasonal and regular differencing for Mauna Loa and Alert, respectively.

Figure 4.6: Plots of $x_{1,t}$ (Top),$\Delta_{12}x_{1,t}$ (Middle), and $\Delta_1\Delta_{12}x_{1,t}$ (Bottom) Partitioned Into Fitting (solid) and Forecasting (dotted) Periods



The DGPs of $\{y_{1,t}\}$ and $\{y_{2,t}\}$ are hidden to the observer; therefore, evaluating

Figure 4.7: Plots of $x_{2,t}$ (Top),$\Delta_{12}x_{2,t}$ (Middle), and $\Delta_1\Delta_{12}x_{2,t}$ (Bottom) Partitioned Into Fitting (solid) and Forecasting (dotted) Periods



the ability of a subset selection method to uncover the truth is an impossible task. The exploration into various cross-validation methods was motivated by the terminal desire to produce forecasts. Frequentist and Bayesian methods in Tables 4.1 and 4.2 are applied to estimate subset ARMA$(14, 14)$ models on the data provided in the fitting period. Methods AL_5, AL_7, AL_{10} and elastic net counterparts are removed from the consideration because of their similarity to other methods. From the final subset ARMA$(14, 14)$ models, rolling one-step ahead predictions $\hat{y}_{k,t}$ are obtained over the full forecasting period of length n_k where $k \in \{1, 2\}$. As previously determined,

$n_1 = 96$ and $n_2 = 24$.

Additionally, three classic methods are explored for baseline forecast performance. First, the naive random walk (RW) model, which does not require estimation, is considered. RW forecasts are obtained via $\hat{y}_{k,t} = y_{k,t-1}$. Next, a saturated ARMA$(\tilde{p}, \tilde{q})$ where $\tilde{p} < 14$ and $\tilde{q} < 14$ is estimated. Finally, a saturated SARMA$(\tilde{p}, \tilde{q}) \times (\tilde{P}, \tilde{Q})_s$, under prior assumptions $s = 12$, $\tilde{p} < 14$, $\tilde{q} < 14$, $\tilde{P} < 14$, and $\tilde{Q} < 14$, is also used. Best ARMA and SARMA models are selected using **auto.arima()** in the **forecast** package. By default, a stepwise algorithm searches for $\tilde{p}$, $\tilde{q}$, $\tilde{P}$, and $\tilde{Q}$ based on minimization of AIC.

Methods are evaluated based on root mean squared error ($RMSE$), mean absolute scaled error ($MASE$), mean bias (MB), and mean directional bias (MDB). The formulas for these metrics are expressed in Equation 4.25. $MASE$ is based on errors scaled by the mean absolute error under the naive RW model for the training period ($MAE_{k,\text{RW}}$). $MASE < 1$ occurs when a method outperforms the naive RW, on average (Hyndman and Koehler, 2006). In the expression for MDB, $\text{sgn}(x_t) = 1$ if $x_t > 0$ and $\text{sgn}(x_t) = -1$ if $x_t < 0$. Large values of $RMSE$ and $MASE$ indicate poor predictive accuracy. The bias metrics, MB and MDB, are negative when models overestimate future values and positive when models underestimate future values.

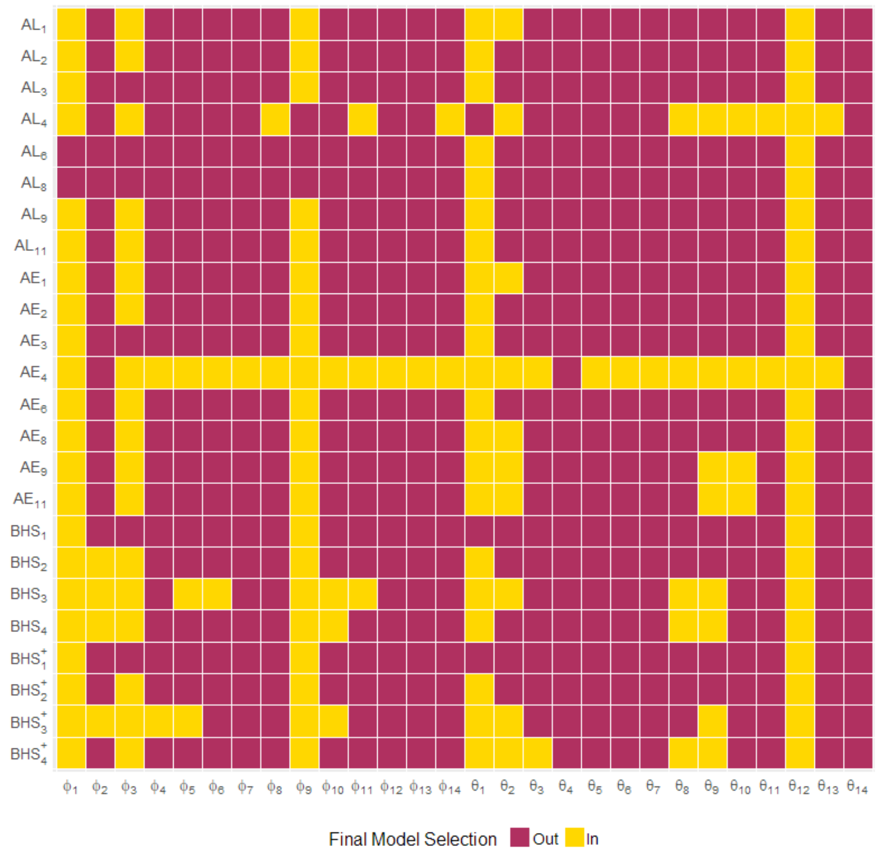
$$\begin{aligned}
RMSE_k &= \frac{1}{n_k} \sum_{j=1}^{n_k} (y_{k,j} - \hat{y}_{k,j})^2 \\
MASE_k &= \frac{1}{n_k} \sum_{j=1}^{n_k} \left| \frac{y_{k,j} - \hat{y}_{k,j}}{MAE_{k,\text{RW}}} \right| \\
MB_k &= \frac{1}{n_k} \sum_{j=1}^{n_k} (y_{k,j} - \hat{y}_{k,j}) \\
MDB_k &= \frac{1}{n_k} \sum_{j=1}^{n_k} \text{sgn}(y_{k,j} - \hat{y}_{k,j})
\end{aligned} \tag{4.25}$$

Starting with the Mauna Loa series $\{y_{1,t}\}$, final subset ARMA$(14, 14)$ model selection is summarized in Figure 4.8. AR parameters $\{\phi_1, \phi_3, \phi_9\}$ and MA parameters $\{\theta_1, \theta_{12}\}$ are consistently selected. Stationary and invertibility characteristics of ARMA are tested according to the characteristic polynomials. Final models under AL_4 and AE_4 fail the invertibility assumption, leading to unbounded forecasts. Biasing final model selection on a single OOS period seems to be less protective against non-stationary and non-invertible estimates. One-step ahead forecast evaluation for the remaining models is displayed in Table 4.12. Bayesian methods outperform ADLASSO and ADENET according to all metrics. Forecasts from BHS_m and BHS_m^+ for $m \in \{1, 2, 3, 4\}$ are slightly more accurate ($RMSE$ & $MASE$) and significantly less biased (MB & MDB).

Forecasting performance from all subset ARMA models is superior to results from the naive RW. Although the bias is relatively low for RW, the error associated with point forecasts is at least double the error for BHS and BHS^+. Based on AIC, saturated ARMA$(0, 1)$ and SARMA$(0, 0, 1)$ are selected. The Bayesian subset ARMA models outperform ARMA$(0, 1)$. When compared to ADLASSO and ADENET, forecasting accuracy is similar, but MB and MDB show ARMA$(0, 1)$ forecasts are less biased. Furthermore, the sign difference in MDB indicates ARMA$(0, 1)$ forecasts are occasionally underestimated while AL and AE often overestimate. The SARMA model is extremely competitive to BHS and BHS^+. Recall that estimation of SARMA requires knowing the seasonal periodicity $s = 12$; and although this is a reasonable assumption, subset ARMA methods do not require this prior belief.

In regards to the Alert series $\{y_{2,t}\}$, final subset ARMA$(14, 14)$ model selection is summarized in Figure 4.9. Cryer and Chan (2008) and Chen and Chan (2011) build models for $\{y_{2,t}\}$ but neither evaluate forecasting; therefore, they use the full series for model selection and estimation. Despite this difference, the best subset ARMA model from ADLASSO, containing $\{\phi_1, \phi_{12}, \theta_9, \theta_{11}, \theta_{12}\}$, overlaps with many of the

Figure 4.8: Final Model Selection for Mauna Loa CO_2



final subset ARMA models (Chen and Chan, 2011). Post ADLASSO, Chen and Chan (2011) refit subset SARMA$(1, 9) \times (1, 1)_{12}$, where both ϕ_1 and ϕ_{12} are deselected. Parameters ϕ_1 and θ_{12} are consistently relevant. Recall the similar pattern exhibited for Mauna Loa (Figure 4.8). Specifically for Bayesian methods, the seasonal AR parameter ϕ_{12} is selected in all final models. For the shorter Alert data, the final AL/AD are more parsimonious than final BHS/BHS$^+$. This effect is not as pronounced for Mauna Loa. This implies that the forecasting advantages from BHS/BHS$^+$ (see Table 4.12) are solely based on the improved estimation of relevant ARMA parameters under Bayesian horseshoe regularization.

Table 4.13 summarizes one-step ahead forecasting for $\{y_{2,t}\}$. Again, stationarity

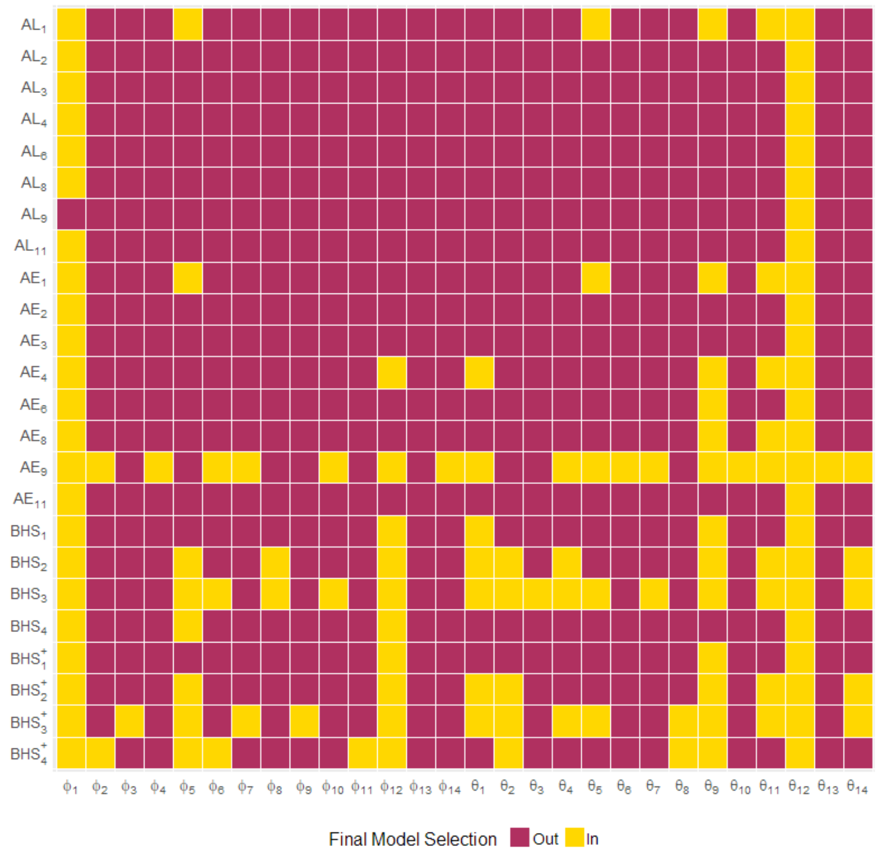
Table 4.12: One-Step Ahead Forecasting Results for Mauna Loa CO_2

	$RMSE$		$MASE$		MB		MDB	
m	AL_m	AE_m	AL_m	AE_m	AL_m	AE_m	AL_m	AE_m
1	0.34	0.34	0.53	0.53	-0.13	-0.13	-0.31	-0.29
2	0.33	0.33	0.52	0.52	-0.10	-0.10	-0.21	-0.21
3	0.34	0.34	0.52	0.52	-0.10	-0.10	-0.21	-0.21
4	Not Invertible (NI)							
6	0.34	0.33	0.53	0.52	-0.10	-0.09	-0.17	-0.23
8	0.34	0.34	0.53	0.54	-0.10	-0.14	-0.19	-0.31
9	0.34	0.36	0.52	0.57	-0.10	-0.18	-0.23	-0.44
11	0.33	0.36	0.52	0.56	-0.10	-0.17	-0.21	-0.44
	$RMSE$		$MASE$		MB		MDB	
m	BHS_m	BHS_m^+	BHS_m	BHS_m^+	BHS_m	BHS_m^+	BHS_m	BHS_m^+
1	0.31	0.31	0.49	0.49	-0.01	-0.01	0.04	0.06
2	0.32	0.32	0.50	0.50	-0.02	-0.02	0.00	-0.02
3	0.32	0.32	0.51	0.51	-0.02	-0.02	0.00	0.00
4	0.32	0.32	0.50	0.50	-0.01	-0.01	0.02	0.02

m	$RMSE$	$MASE$	MB	MDB
RW	0.64	1.03	0.00	-0.02
ARMA	0.37	0.60	0.01	0.13
SARMA	0.30	0.49	-0.04	-0.02

and invertiblity are checked. Results for AL_1, AE_1, and AE_9 are unreported for failing the invertibility condition. Again, the naive RW produces the worst forecasts. For Alert, saturated ARMA$(1,1)$ and SARMA$(0,1)\times(0,1)_{12}$ are selected. The latter is identical to the one selected in Cryer and Chan (2008). All subset ARMA methods perform as well or better than baseline ARMA, but outperforming the best SARMA is difficult. In this scenario, there is not a clear divide between the frequentist and Bayesian methods. All results are based on a short time period ($n_2 = 24$) and no subset ARMA procedure is definitively superior.

Figure 4.9: Final Model Selection for Alert CO_2



Although a "best" procedure does not emerge for the Alert series, this example is an opportunity to reemphasize the importance of the stationarity and invertibility conditions. Methods AL_1 and AE_1 use AIC in the selection of optimal tuning parameters λ^* and α^*. Each resulting estimate, $\hat{\boldsymbol{\beta}}_{AL}(\lambda^*)$ and $\hat{\boldsymbol{\beta}}_{AE}(\lambda^*, \alpha^*)$, produce a set of MA coefficients $\hat{\boldsymbol{\theta}}$ that represent a non-invertible ARMA process. Naturally, out-of-sample forecasting from AL_1 and AE_1 is poor and unreported in Table 4.13. Both of these approaches for subset ARMA estimation were introduced and demonstrated in Chen and Chan (2011); however, this issue is also present in AL_4 and AE_4 for Mauna Loa and in AE_9 for Alert. Specifically for AL_1 and AE_1, the grid search across $\mathcal{L}$ and $\mathcal{A}$ is adjusted to only produce estimates of stationary and invertible

Table 4.13: One-Step Ahead Forecasting Results for Alert CO_2

	RMSE		*MASE*		*MB*		*MDB*	
m	AL_m	AE_m	AL_m	AE_m	AL_m	AE_m	AL_m	AE_m
1	Not Invertible (NI)							
2	0.84	0.84	0.43	0.43	0.03	0.03	0.08	0.08
3	0.84	0.84	0.43	0.43	0.05	0.05	0.08	0.08
4	0.85	0.81	0.42	0.39	0.03	0.06	0.08	0.17
6	0.86	0.81	0.41	0.42	-0.07	0.12	0.08	0.33
8	0.86	0.82	0.41	0.40	-0.07	0.08	0.08	0.25
9	1.14	NI	0.62	NI	-0.12	NI	-0.17	NI
11	0.86	0.84	0.41	0.41	-0.07	-0.02	0.08	0.08
	RMSE		*MASE*		*MB*		*MDB*	
m	BHS_m	BHS^+_m	BHS_m	BHS^+_m	BHS_m	BHS^+_m	BHS_m	BHS^+_m
1	0.82	0.83	0.39	0.40	-0.10	-0.10	0.00	0.00
2	0.82	0.83	0.39	0.39	-0.11	-0.11	-0.08	-0.08
3	0.81	0.82	0.38	0.39	-0.11	-0.11	-0.08	-0.08
4	0.88	0.89	0.43	0.43	-0.12	-0.12	-0.08	0.00
m	*RMSE*		*MASE*		*MB*		*MDB*	
RW	2.11		1.15		-0.08		0.00	
ARMA	0.91		0.44		-0.11		0.08	
SARMA	0.79		0.41		0.01		0.08	

ARMA process. The final models under AL_1 and AE_1 identify a new set of relevant parameters $\{\phi_1, \theta_9, \theta_{11}, \theta_{12}\}$ which is even closer to the best model identified in Chen and Chan (2011). The adjusted forecasting results from AL_1 and AE_1 are displayed in Table 4.14. These issues stem from the treatment of ARMA as an unconstrained linear regression, and this approach is inappropriate for nonstationary data. For data that seems to be stationary, i.e. $\{y_{1,t}\}$ and $\{y_{2,t}\}$, ADLASSO and ADENET methods are easily modified to ignore parts of the solution path that violate the important regulatory assumptions. Similarly, BHS and BHS^+ can be modified to ignore posterior samples $\boldsymbol{\beta}^{(s)}$ and $\sigma^{(s)}$ if $\boldsymbol{\beta}^{(s)}$ is not ARMA.

Table 4.14: Adjusted One-Step Ahead Forecasting Results for Alert CO_2

	RMSE		*MASE*		*MB*		*MDB*	
m	AL_m	AE_m	AL_m	AE_m	AL_m	AE_m	AL_m	AE_m
1	0.81	0.81	0.40	0.40	0.08	0.08	0.25	0.25

4.5 Conclusion

Subset ARMA(p, q) models are widely applicable for modeling temporal dynamics and forecasting of weakly stationary time series. ARMA modeling via linear regression has advantages and disadvantages. If p and q are intentionally overestimated, frequentist and Bayesian regularization techniques, that shrink irrelevant parameters to 0 without overshrinking relevant parameters, are easily applied. All regularization methods presented were chosen based on their theoretical oracle properties and capability of handling correlated predictors. However, it is not guaranteed the final $\hat{\boldsymbol{\beta}}$ represents a stationary and invertible ARMA process. Simple adjustments to all methods are discussed to remove this problem. Another issue is the sensitivity to the selection of proxy innovations $\{\hat{\epsilon}_t\}$ using a long AR(p') model. It is strongly suggested that initial model selection is not performed for selection of p' at this step.

When ADLASSO or ADENET methods are used to automate estimation and model selection, the approach taken to search for tuning parameters is important. Empirical analysis demonstrates that the true DGP limits the effectiveness of these approaches. Modified BCV-K based on the maximum temporal dependency does not improve model selection or forecasting over CV-K in ADLASSO. Based on simulation results, CV-K is recommended for ADENET regularization. Bergmeir *et al.* (2018) shows that regular CV-K always outperforms OOS and adequately estimates PE if the considered model is not far from the truth. Although the saturated model is grossly overfitted, estimation via regularization shrinks $\hat{\boldsymbol{\beta}}$ to the "truth," preventing this from being an issue.

Bayesian regularization via horseshoe priors reduces irrelevant effects but does not perform model selection. Posterior distributions of subset ARMA(p, q) models can be obtained through projection removing the need for repeated Gibbs sampling. Whether BHS or BHS^{+} is chosen, a general improvement in model selection and forecasting is observed compared to ADLASSO and ADENET. Also, posterior means $\hat{\boldsymbol{\beta}}$ across replications and practicle examples consistently validated stationary and invertible conditions. However, when the unknown DGP was a subset MA(q) process, BHS and BHS^{+} rarely selected the corrected model. Combining CV algorithms with BHS and BHS^{+} may produce better results but are computationally expensive (Peltola *et al.*, 2014); therefore, this is left for future research.

All discussed methods are quick and easy to employ. In Appendix C, detailed **R** code is provided to encourage reproducibility. Furthermore, the application of these methods to the Mauna Loa CO_2 time series is included to demonstrate usage and illustrate forecasting.

Chapter 5

CONCLUSION

The overall focus of this dissertation has been on the application and evaluation of Bayesian regularization and model selection methods to obtain sparse estimation of linear and nonlinear time series models. A different model and application is provided in each chapter to illustrate the overall efficacy of considering Bayesian approaches for discovering the relevant temporal dynamics for the purpose of forecasting at multi-step horizons. Each chapter provides a novelty that contributes to the growing field of Bayesian time series analysis.

In Chapter 2, different shrinkage priors are utilized to estimate a 2-regime smooth transition autoregressive model with a more flexible parametric representation than previously used. The use of the *dirichlet* prior to select the delay parameter allows for composite transition variables to be estimated. Regime-specific tuning parameters in hierachical representations of global-local shrinkage priors ensure that regularization is regime-specific. The corresponding Appendix A contains detailed **R** code making these methods reproducible for future applications. Using deviations from daily maximum water temperature profiles, the ease of these methods in estimating smooth transition autoregressions with endogenous and exogenous lag effects is illustrated. Often smooth transition autoregressive models are applied to univariate time series, but the Bayesian methods discussed are able to perform selection on lag effects for input time series, such as deviations from maximum air temperature profiles.

The threshold autoregressive process is the limit of its smooth counterpart where the slope in the transition function approaches infinity. Often times in practice, the difficulty in estimation restricts consideration to threshold autoregressive models with at most 3 regimes. In Chapter 3, the nonlinear threshold autoregressive process is restructured to a high dimensional linear regression model through limiting the thresholds to a finite set. This re-framed approach is only found in a handful of works but should become industry standard since the linear form nests all threshold autoregressive models with regimes less than the sample size. In this context, a fully Bayesian three step model building procedure is outlined to not only select the number of regimes but also perform within regime variable selection. Empirical results from a high dimensional simulation study in Appendix B are referenced to defend the choice of the horseshoe+ shrinkage prior. Using traffic occupancy data, the best subset TAR model outperforms seasonal profiles for 3 minute, 9 minute, and 15 minute forecasting horizons. Final TAR models are also used to produce density forecasts for the entire out-of-sample period.

Chapter 4 focuses on subset selection of the classic autoregressive moving average model which has proven to be most popular in modeling and forecasting stationary time series. By considering subset selection methods, the more complicated multiplicative seasonal model can be estimated without knowing the period. From a frequentist viewpoint, penalized adaptive LASSO estimation has been used to yield consistent subset selection of these models. The adaptive elastic net is a natural extension from adaptive LASSO with a more flexible penalty. Previous works have used information criteria to select tuning parameters in these circumstances. Various cross-validation techniques are also appropriate alternatives to information criteria, even for time series data. For comparison, a Bayesian approach, that uses the Kullback-Leibler distance, searches for the best submodel with a posterior predictive distribution relatively close to the predictions from the full model. Within this method, there are multiple ways to identify the final model that do not require cross-validation. In simulation, potential pitfalls of adaptive lasso and adaptive elastic net are shown, highlighting the advantages of the Bayesian-based posterior predictive projection algorithm. Subset ARMA methods are applied to CO_2 data from two locations. Multiple measures of forecasting accuracy and bias assess the techniques for one-step ahead forecasts. Code provided in Appendix C makes the application of all discussed methods reproducible for users.

REFERENCES

Ahmed, M. S. and A. R. Cook, *Analysis of freeway traffic time-series data by using Box-Jenkins techniques*, no. 722 (1979).

Akaike, H., "A new look at the statistical model identification", IEEE transactions on automatic control **19**, 6, 716–723 (1974).

Akaike, H., "Prediction and entropy", in "Selected Papers of Hirotugu Akaike", pp. 387–410 (Springer, 1985).

Albouy, C., L. Velez, M. Coll, F. Colloca, F. Loc'h, D. Mouillot and D. Gravel, "From projected species distribution to food-web structure under climate change", Global change biology **20**, 3, 730–741 (2014).

Andrews, D. F. and C. L. Mallows, "Scale mixtures of normal distributions", Journal of the Royal Statistical Society. Series B (Methodological) **36**, 1, 99–102 (1974).

Annunziato, M., F. Moretti and S. Pizzuti, "Urban traffic flow forecasting using neural-statistic hybrid modeling", in "Soft Computing Models in Industrial and Environmental Applications", pp. 183–190 (Springer, 2013).

Arlot, S., A. Celisse *et al.*, "A survey of cross-validation procedures for model selection", Statistics surveys **4**, 40–79 (2010).

Armagan, A., D. B. Dunson and J. Lee, "Generalized double pareto shrinkage", Statistica Sinica **23**, 1, 119 (2013).

Battaglia, F. and M. K. Protopapas, "An analysis of global warming in the alpine region based on nonlinear nonstationary time series models", Statistical Methods & Applications **21**, 3, 315–334 (2012).

Benyahya, L., D. Caissie, A. St-Hilaire, T. B. Ouarda and B. Bobée, "A review of statistical water temperature models", Canadian Water Resources Journal **32**, 3, 179–192 (2007).

Bergmeir, C. and J. M. Benítez, "On the use of cross-validation for time series predictor evaluation", Information Sciences **191**, 192–213 (2012).

Bergmeir, C., M. Costantini and J. M. Benítez, "On the usefulness of cross-validation for directional forecast evaluation", Computational Statistics & Data Analysis **76**, 132–143 (2014).

Bergmeir, C., R. J. Hyndman and B. Koo, "A note on the validity of cross-validation for evaluating autoregressive time series prediction", Computational Statistics & Data Analysis **120**, 70–83 (2018).

Bhadra, A., J. Datta, N. G. Polson, B. Willard *et al.*, "The horseshoe+ estimator of ultra-sparse signals", Bayesian Analysis (2016).

Bhattacharya, A., A. Chakraborty and B. K. Mallick, "Fast sampling with gaussian scale mixture priors in high-dimensional regression", Biometrika p. asw042 (2016).

Bhattacharya, A., D. Pati, N. S. Pillai and D. B. Dunson, "Dirichletlaplace priors for optimal shrinkage", Journal of the American Statistical Association **110**, 512, 1479–1490, URL `http://dx.doi.org/10.1080/01621459.2014.960967`, pMID: 27019543 (2015).

Box, G. E. and G. M. Jenkins, *Time Series Analysis: Forecasting and Control, Revised Ed.* (Holden-Day, 1976).

Brockwell, P. J. and R. A. Davis, *Introduction to time series and forecasting* (springer, 2016).

Broemeling, L. D. and P. Cook, "Bayesian analysis of threshold autoregressions", Communications in Statistics-Theory and Methods **21**, 9, 2459–2482 (1992).

Burman, P., E. Chow and D. Nolan, "A cross-validatory method for dependent data", Biometrika **81**, 2, 351–358 (1994).

Burnham, K. P. and D. R. Anderson, *Model selection and multimodel inference: a practical information-theoretic approach* (Springer Science & Business Media, 2003).

Burnham, K. P. and D. R. Anderson, "Multimodel inference: understanding aic and bic in model selection", Sociological methods & research **33**, 2, 261–304 (2004).

Caissie, D., N. El-Jabi and M. G. Satish, "Modelling of maximum daily water temperatures in a small stream using air temperatures", Journal of Hydrology **251**, 1, 14–28 (2001).

Caissie, D., N. El-Jabi and A. St-Hilaire, "Stochastic modelling of water temperatures in a small stream using air to water relations", Canadian Journal of Civil Engineering **25**, 2, 250–260 (1998).

Campbell, E. P., "Bayesian selection of threshold autoregressive models", Journal of time series analysis. **25**, 4, 467–482 (2004).

Campbell, G. and J. Mosimann, "Multivariate methods for proportional shape", in "ASA Proceedings of the Section on Statistical Graphics", vol. 1, pp. 10–17 (Washington, 1987).

Carlin, B. P. and S. Chib, "Bayesian model choice via markov chain monte carlo methods", Journal of the Royal Statistical Society. Series B (Methodological) pp. 473–484 (1995).

Carvalho, C. M., N. G. Polson and J. G. Scott, "Handling sparsity via the horseshoe", in "Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics", edited by D. van Dyk and M. Welling, vol. 5 of *Proceedings of Machine Learning Research*, pp. 73–80 (PMLR, Hilton Clearwater Beach Resort, Clearwater Beach, Florida USA, 2009), URL `http://proceedings.mlr.press/v5/carvalho09a.html`.

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

Casella, G., "Empirical bayes gibbs sampling", Biostatistics **2**, 4, 485–500 (2001).

Castillo, I., A. van der Vaart *et al.*, "Needles and straw in a haystack: Posterior concentration for possibly sparse sequences", The Annals of Statistics **40**, 4, 2069–2101 (2012).

Chan, K.-S. and B. Ripley, *TSA: Time Series Analysis*, URL `https://CRAN.R-project.org/package=TSA`, r package version 1.01 (2012).

Chan, N. H., C.-K. Ing, Y. Li and C. Y. Yau, "Threshold estimation via group orthogonal greedy algorithm", Journal of Business & Economic Statistics **35**, 2, 334–345 (2017).

Chan, N. H., C. Y. Yau and R.-M. Zhang, "Lasso estimation of threshold autoregressive models", Journal of Econometrics **189**, 2, 285–296 (2015).

Chen, C. W., "A bayesian analysis of generalized threshold autoregressive models", Statistics & probability letters **40**, 1, 15–22 (1998).

Chen, C. W. and J. C. Lee, "Bayesian inference of threshold autoregressive models", Journal of Time Series Analysis **16**, 5, 483–492 (1995).

Chen, K. and K.-S. Chan, "Subset arma selection via the adaptive lasso", Statistics and its Interface **4**, 2, 197–205 (2011).

Cryer, J. D. and K.-S. Chan, "Seasonal models", Time series analysis: with applications in R pp. 227–246 (2008).

Datta, J. and J. K. Ghosh, "Asymptotic properties of bayes risk for the horseshoe prior", Bayesian Analysis **8**, 1, 111–132 (2013).

Datta, J. and J. K. Ghosh, "In search of optimal objective priors for model selection and estimation", Current Trends in Bayesian Methodology with Applications p. 225 (2015).

Davis, R. A., T. C. M. Lee and G. A. Rodriguez-Yam, "Structural break estimation for nonstationary time series models", Journal of the American Statistical Association **101**, 473, 223–239 (2006).

De Mol, C., E. De Vito and L. Rosasco, "Elastic-net regularization in learning theory", Journal of Complexity **25**, 2, 201–230 (2009).

Dellaportas, P., J. Forster and I. Ntzoufras, "On bayesian model and variable selection using mcmc", Statistics and Computing **12**, 1, 27–36 (2002).

Deschamps, P. J., "Comparing smooth transition and Markov switching autoregressive models of US unemployment", Journal of Applied Econometrics **23**, 4, 435–462, URL `http://dx.doi.org/10.1002/jae.1014` (2008).

Dunne, S. and B. Ghosh, "Regime-based short-term multivariate traffic condition forecasting algorithm", Journal of Transportation Engineering **138**, 4, 455–466 (2012).

Dupuis, J. A. and C. P. Robert, "Variable selection in qualitative models via an entropic explanatory power", Journal of Statistical Planning and Inference **111**, 1, 77–94 (2003).

Efron, B., T. Hastie, I. Johnstone, R. Tibshirani *et al.*, "Least angle regression", The Annals of statistics **32**, 2, 407–499 (2004).

Fan, J. and R. Li, "Variable selection via nonconcave penalized likelihood and its oracle properties", Journal of the American statistical Association **96**, 456, 1348–1360 (2001).

Fan, J. and J. Lv, "Nonconcave penalized likelihood with np-dimensionality", IEEE Transactions on Information Theory **57**, 8, 5467–5484 (2011).

Fan, J. and H. Peng, "Nonconcave penalized likelihood with a diverging number of parameters", The Annals of Statistics **32**, 3, 928–961 (2004).

Franses, P. H. and D. Van Dijk, *Non-linear time series models in empirical finance* (Cambridge University Press, 2000).

Friedman, J., T. Hastie and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent", Journal of statistical software **33**, 1, 1 (2010).

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

Gelman, A., J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari and D. B. Rubin, *Bayesian data analysis*, vol. 2 (CRC press Boca Raton, FL, 2014).

Gelman, A. and D. B. Rubin, "Inference from iterative simulation using multiple sequences", Statistical science pp. 457–472 (1992).

Geman, S. and D. Geman, "Stochastic relaxation, gibbs distributions, and the bayesian restoration of images", in "Readings in Computer Vision", pp. 564–584 (Elsevier, 1987).

George, E. and D. P. Foster, "Calibration and empirical bayes variable selection", Biometrika **87**, 4, 731–747 (2000).

George, E. I. and R. E. McCulloch, "Variable selection via gibbs sampling", Journal of the American Statistical Association **88**, 423, 881–889, URL `http://dx.doi.org/10.1080/01621459.1993.10476353` (1993).

Gerlach, R. and C. W. Chen, "Bayesian inference and model comparison for asymmetric smooth transition heteroskedastic models", Statistics and Computing **18**, 4, 391–408 (2008).

Geweke, J. and N. Terui, “Bayesian threshold autoregressive models for nonlinear time series”, Journal of Time Series Analysis **14**, 5, 441–454 (1993).

Ghaddar, D. and H. Tong, “Data transformation and self-exciting threshold autoregression”, Applied Statistics **30**, 3, URL `http://search.proquest.com/docview/1299676897/` (1981).

Ghosh, B., B. Basu and M. OMahony, “Time-series modelling for forecasting vehicular traffic flow in dublin”, in “Proceedings of the 85th Transportation Research Board Annual Meeting, Washington, DC”, (2005).

Ghosh, B., B. Basu and M. OMahony, “Bayesian time-series model for short-term traffic flow forecasting”, Journal of transportation engineering **133**, 3, 180–189 (2007).

Goutis, C. and C. P. Robert, “Model choice in generalised linear models: A bayesian approach via kullback-leibler projections”, Biometrika **85**, 1, 29–37 (1998).

Granger, C. W. J., “A statistical model for sunspot activity”, Astrophysical Journal **126**, 152 (1957).

Green, P. J., “Reversible jump markov chain monte carlo computation and bayesian model determination”, (1995).

Hahn, P. R., J. He and H. Lopes, “Elliptical slice sampling for bayesian shrinkage regression with applications to causal inference”, U RL http://faculty. chicagobooth. edu/richard. hahn/research. html (2016).

Hall, F., “Traffic stream characteristics”, (1992).

Hamilton, J. D., *Time series analysis*, vol. 2 (Princeton university press Princeton, 1994).

Hannan, E. J. and L. Kavalieris, “A method for autoregressive-moving average estimation”, Biometrika **71**, 2, 273–280 (1984).

Hannan, E. J. and J. Rissanen, “Recursive estimation of mixed autoregressive-moving average order”, Biometrika **69**, 1, 81–94 (1982).

Hans, C., “Bayesian lasso regression”, Biometrika **96**, 4, 835–845 (2009).

Hastie, T., R. Tibshirani and J. Friedman, “The elements of statistical learning 2nd edition”, (2009).

Hastings, W. K., “Monte carlo sampling methods using markov chains and their applications”, Biometrika **57**, 1, 97–109 (1970).

Hazelton, M., “Estimating vehicle speed from traffic count and occupancy data”, **2**, 231–244 (2004).

Hebiri, M. and J. Lederer, “How correlations influence lasso prediction”, IEEE Transactions on Information Theory **59**, 3, 1846–1854 (2013).

Hellendoorn, J., S. K. Zegeye and J. W. Zwarteveen, "Urban traffic flow modeling in addis ababa", pp. 620–625 (IEEE Publishing, 2011).

Hijazi, R. and R. Jernigan, "Modelling compositional data using dirichlet regression models", **4**, 77–91 (2009).

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

Hoeting, J. A., D. Madigan, A. E. Raftery and C. T. Volinsky, "Bayesian model averaging", in "Proceedings of the AAAI Workshop on Integrating Multiple Learned Models", vol. 335, pp. 77–83 (Citeseer, 1998).

Hoeting, J. A., D. Madigan, A. E. Raftery and C. T. Volinsky, "Bayesian model averaging: a tutorial", Statistical science pp. 382–401 (1999).

Hsiang, T. C., "A bayesian view on ridge regression", Journal of the Royal Statistical Society. Series D (The Statistician) **24**, 4, 267–268, URL `http://www.jstor.org/stable/2987923` (1975).

Hyndman, R. J. and Y. Khandakar, "Automatic time series forecasting: the forecast package for R", Journal of Statistical Software **26**, 3, 1–22, URL `http://www.jstatsoft.org/article/view/v027i03` (2008).

Hyndman, R. J. and A. B. Koehler, "Another look at measures of forecast accuracy", International journal of forecasting **22**, 4, 679–688 (2006).

Isaak, D. J., C. H. Luce, B. E. Rieman, D. E. Nagel, E. E. Peterson, D. L. Horan, S. Parkes and G. L. Chandler, "Effects of climate change and wildfire on stream temperatures and salmonid thermal habitat in a mountain river network", Ecological Applications **20**, 5, 1350–1371 (2010).

Ishwaran, H. and J. S. Rao, "Spike and slab variable selection: Frequentist and bayesian strategies", (2005).

Ishwaran, H. and J. S. Rao, "Consistency of spike and slab regression", Statistics & Probability Letters **81**, 12, 1920–1928 (2011).

Johnstone, I. M., B. W. Silverman *et al.*, "Needles and straw in haystacks: Empirical bayes estimates of possibly sparse sequences", The Annals of Statistics **32**, 4, 1594–1649 (2004).

Jun, Z. and L. Jun, "Nonlinear characteristics of short - term traffic flow and their influences to forecasting", pp. 847–851 (IEEE, 2007).

Kamarianakis, Y., S. V. Ayuso, E. C. Rodríguez and M. T. Velasco, "Water temperature forecasting for spanish rivers by means of nonlinear mixed models", Journal of Hydrology: Regional Studies **5**, 226–243 (2016).

Kamarianakis, Y., H. O. Gao and P. Prastacos, "Characterizing regimes in daily cycles of urban traffic using smooth-transition regressions", Transportation Research Part C: Emerging Technologies **18**, 5, 821–840 (2010).

Kamarianakis, Y., W. Shen and L. Wynter, “Real-time road traffic forecasting using regime-switching space-time models and adaptive lasso”, Applied stochastic models in business and industry **28**, 4, 297–315 (2012).

Klein, L. and M. Kelley, “Detection technology for ivhs: Final report”, Federal Highway Administration FHWA-RD-95-100 (1996).

Konzen, E. and F. A. Ziegelmann, “Lasso-type penalties for covariate selection and forecasting in time series”, Journal of Forecasting **35**, 7, 592–612 (2016).

Kullback, S. and R. A. Leibler, “On information and sufficiency”, The annals of mathematical statistics **22**, 1, 79–86 (1951).

Kumar, S. V. and L. Vanajakshi, “Short-term traffic flow prediction using seasonal arima model with limited input data”, European Transport Research Review **7**, 3, 1–9 (2015).

Kuo, L. and B. Mallick, “Variable selection for regression models”, The Indian Journal of Statistics, Series B (1960-2002) **60**, 1, 65–81 (1998).

Lee, T.-H., H. White and C. W. Granger, “Testing for neglected nonlinearity in time series models: A comparison of neural network methods and alternative tests”, Journal of Econometrics **56**, 3, 269–290 (1993).

Leng, C., M.-N. Tran and D. Nott, “Bayesian adaptive lasso”, Annals of the Institute of Statistical Mathematics **66**, 2, 221–244 (2014).

Levin, M. and Y.-D. Tsao, “On forecasting freeway occupancies and volumes (abridgment)”, Transportation Research Record , 773 (1980).

Li, D. and S. Ling, “On the least squares estimation of multiple-regime threshold autoregressive models”, Journal of Econometrics **167**, 1, 240–253 (2012).

Li, Q. and N. Lin, “The bayesian elastic net”, Bayesian Analysis **5**, 1, 151–170 (2010).

Li, Q., R. Xi, N. Lin *et al.*, “Bayesian regularized quantile regression”, Bayesian Analysis **5**, 3, 533–556 (2010).

Lin, J.-L. and C. W. J. Granger, “Forecasting from non-linear models in practice”, Journal of Forecasting **13**, 1, 1–9 (1994).

Livingston Jr., G. and D. Nur, “Bayesian inference for smooth transition autoregressive (star) model: A prior sensitivity analysis”, Communications in Statistics - Simulation and Computation **46**, 7, 5440–5461, URL `http://dx.doi.org/10.1080/03610918.2016.1161794` (2017).

Lopes, H. F. and E. Salazar, “Bayesian model uncertainty in smooth transition autoregressions”, Journal of Time Series Analysis **27**, 1, 99–117, URL `http://dx.doi.org/10.1111/j.1467-9892.2005.00455.x` (2006).

Lubrano, M., “Bayesian analysis of nonlinear time series models with a threshold”, (2000).

Lundbergh, S. and T. Teräsvirta, *Forecasting with smooth transition autoregressive models* (Wiley Online Library, 2002).

Lykou, A. and I. Ntzoufras, “On bayesian lasso variable selection and the specification of the shrinkage parameter”, Statistics and Computing **23**, 3, 361–390 (2013).

Madigan, D. and A. E. Raftery, “Model selection and accounting for model uncertainty in graphical models using occam's window”, Journal of the American Statistical Association **89**, 428, 1535–1546 (1994).

Makalic, E. and D. F. Schmidt, “A simple sampler for the horseshoe estimator”, IEEE Signal Processing Letters **23**, 1, 179–182 (2016).

Mallick, H. and N. Yi, “Bayesian methods for high dimensional linear models”, Journal of biometrics & biostatistics **1**, 005 (2013).

Marcellino, M., J. H. Stock and M. W. Watson, “A comparison of direct and iterated multistep ar methods for forecasting macroeconomic time series”, Journal of Econometrics **135**, 1, 499–526 (2006).

Martin, A. D., K. M. Quinn and J. H. Park, “MCMCpack: Markov chain monte carlo in R”, Journal of Statistical Software **42**, 9, 22, URL `http://www.jstatsoft.org/v42/i09/` (2011).

Metcalfe, A. V. and P. S. Cowpertwait, *Introductory Time Series with R*, vol. 11 of *Use R* (Springer New York, New York, NY, 2009).

Metropolis, N., A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller, “Equation of state calculations by fast computing machines”, The Journal of Chemical Physics **21**, 6, 1087–1092 (1953).

Min, X., J. Hu and Z. Zhang, “Urban traffic network modeling and short-term traffic flow forecasting based on gstarima model”, pp. 1535–1540 (IEEE Publishing, 2010).

Mitchell, T. J. and J. J. Beauchamp, “Bayesian variable selection in linear regression”, Journal of the American Statistical Association **83**, 404, 1023–1032 (1988).

Mohseni, O., H. G. Stefan and T. R. Erickson, “A nonlinear regression model for weekly stream temperatures”, Water Resources Research **34**, 10, 2685–2692, URL `http://dx.doi.org/10.1029/98WR01877` (1998).

Montgomery, A. L., V. Zarnowitz, R. S. Tsay and G. C. Tiao, “Forecasting the u.s. unemployment rate”, Journal of the American Statistical Association **93**, 442, 478–493 (1998).

Moretti, F., S. Pizzuti, S. Panzieri and M. Annunziato, “Urban traffic flow forecasting through statistical and neural network bagging ensemble hybrid modeling”, Neurocomputing **167**, 3–7 (2015).

Nardi, Y. and A. Rinaldo, “Autoregressive process modeling via the lasso procedure”, Journal of Multivariate Analysis **102**, 3, 528–549 (2011).

Nott, D. J. and C. Leng, “Bayesian projection approaches to variable selection in generalized linear models”, Computational Statistics & Data Analysis **54**, 12, 3227–3241 (2010).

O’Hara, R. B. and M. J. Sillanpaa, “A review of bayesian variable selection methods: what, how and which”, Bayesian Anal. **4**, 1, 85–117, URL `http://dx.doi.org/10.1214/09-BA403` (2009).

Pan, J., Q. Xia and J. Liu, “Bayesian analysis of multiple thresholds autoregressive model”, Computational Statistics **32**, 1, 219–237, URL `http://dx.doi.org/10.1007/s00180-016-0673-3` (2017).

Park, H. and F. Sakaori, “Lag weighted lasso for time series model”, Computational Statistics **28**, 2, 493–504 (2013).

Park, T. and G. Casella, “The bayesian lasso”, (2008).

Peguin-Feissolle, A., “Bayesian estimation and forecasting in non-linear models application to an lstar model”, Economics Letters **46**, 3, 187–194 (1994).

Peltola, T., A. S. Havulinna, V. Salomaa and A. Vehtari, “Hierarchical bayesian survival analysis and projective covariate selection in cardiovascular event risk prediction”, in “Proceedings of the Eleventh UAI Conference on Bayesian Modeling Applications Workshop - Volume 1218”, BMAW’14, pp. 79–88 (CEUR-WS.org, Aachen, Germany, Germany, 2014), URL `http://dl.acm.org/citation.cfm?id=3020299.3020308`.

Petruccelli, J. D. and S. W. Woolford, “A threshold ar (1) model”, Journal of Applied Probability **21**, 2, 270–286 (1984).

Piironen, J. and A. Vehtari, “Projection predictive model selection for gaussian processes”, (2015a).

Piironen, J. and A. Vehtari, “Projection predictive variable selection using stan+r”, (2015b).

Piironen, J. and A. Vehtari, “On the hyperprior choice for the global shrinkage parameter in the horseshoe prior”, (2016).

Piironen, J. and A. Vehtari, “Comparison of bayesian predictive methods for model selection”, Statistics and Computing **27**, 3, 711–735 (2017).

Plummer, M., “Jags: A program for analysis of bayesian graphical models using gibbs sampling”, **124** (2003).

Polson, N. G. and J. G. Scott, “Shrink globally, act locally: Sparse bayesian regularization and prediction”, Bayesian Statistics **9**, 501–538 (2010).

Polson, N. G. and J. G. Scott, "On the half-cauchy prior for a global scale parameter", Bayesian Analysis **7**, 4, 887–902 (2012).

Priestley, M. B., "Non-linear and non-stationary time series analysis", (1988).

Queen, C. M. and C. J. Albers, "Intervention and causality: Forecasting traffic flows using a dynamic bayesian network", Journal of the American Statistical Association **104**, 486, 669–681, URL `http://dx.doi.org/10.1198/jasa.2009.0042` (2009).

R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, URL `https://www.R-project.org/` (2017).

Racine, J., "Consistent cross-validatory model-selection for dependent data: hv-block cross-validation", Journal of econometrics **99**, 1, 39–61 (2000).

Raftery, A. E., D. Madigan and J. A. Hoeting, "Bayesian model averaging for linear regression models", Journal of the American Statistical Association **92**, 437, 179–191 (1997).

Rahel, F. J. and J. D. Olden, "Assessing the effects of climate change on aquatic invasive species", Conservation Biology **22**, 3, 521–533 (2008).

Rue, H., "Fast sampling of gaussian markov random fields", Journal of the Royal Statistical Society: Series B (Statistical Methodology) **63**, 2, 325–338 (2001).

Schmidt, D. F. and E. Makalic, *bayesreg: Bayesian Regression Models with Continuous Shrinkage Priors*, URL `https://CRAN.R-project.org/package=bayesreg`, r package version 1.0 (2016).

Schwarz, G., "Estimating the dimension of a model", The annals of statistics **6**, 2, 461–464 (1978).

Shang, P., X. Li and S. Kamae, "Nonlinear analysis of traffic time series at different temporal scales", Physics Letters A **357**, 4, 314–318 (2006).

Shao, J., "An asymptotic theory for linear model selection", Statistica Sinica pp. 221–242 (1997).

Smith, B. L. and M. J. Demetsky, "Traffic flow forecasting: Comparison of modeling approaches", Journal of Transportation Engineering **123**, 4, 261–266 (1997).

So, M. K. and C. W. Chen, "Subset threshold autoregression", Journal of Forecasting **22**, 1, 49–66 (2003).

Stankiewicz, S., "Forecasting euro area macroeconomic variables with bayesian adaptive elastic net", (2015).

Stathopoulos, A. and M. G. Karlaftis, "A multivariate state space approach for urban traffic flow modeling and prediction", Transportation Research Part C **11**, 2, 121–135 (2003).

Stock, J. H. and M. W. Watson, “A comparison of linear and nonlinear univariate models for forecasting macroeconomic time series”, URL `http://www.nber.org/papers/w6607.pdf` (1998).

Stone, M., “Cross-validatory choice and assessment of statistical predictions”, Journal of the royal statistical society. Series B (Methodological) pp. 111–147 (1974).

Terasvirta, T. and H. M. Anderson, “Characterizing nonlinearities in business cycles using smooth transition autoregressive models”, Journal of Applied Econometrics **7**, S1 (1992).

Teräsvirta, T., D. Tjøstheim and C. W. J. Granger, *Modelling nonlinear economic time series* (Oxford University Press Oxford, 2010).

Tersvirta, T., “Forecasting economic variables with nonlinear models”, IDEAS Working Paper Series from RePEc URL `http://search.proquest.com/docview/1698460820/` (2005).

Theofilatos, A., G. Yannis, E. I. Vlahogianni and J. C. Golias, “Modeling the effect of traffic regimes on safety of urban arterials: The case study of athens”, Journal of Traffic and Transportation Engineering **4**, 3, 240–251, URL `https://doaj.org/article/5277085c2ef141049e4aec23d98ea726` (2017).

Tibshirani, R., “Regression shrinkage and selection via the lasso”, Journal of the Royal Statistical Society. Series B (Methodological) **58**, 1, 267–288 (1996).

Tong, H., “Non-linear time series”, A Dynamical System Approach (1990).

Troughton, P. T. and S. J. Godsill, “A reversible jump sampler for autoregressive time series, employing full conditionals to achieve efficient model space moves”, (1997).

Van Der Pas, S., B. Kleijn and A. Van Der Vaart, “The horseshoe estimator: Posterior concentration around nearly black vectors”, Electronic Journal of Statistics **8**, 2, 2585–2618 (2014).

van Dijk, D., T. Teräsvirta and P. H. Franses, “Smooth transition autoregressive modelsa survey of recent developments”, Econometric reviews **21**, 1, 1–47 (2002).

Vehtari, A. and J. Ojanen, “A survey of bayesian predictive methods for model assessment, selection and comparison”, Statistics Surveys **6**, 142–228 (2012).

Vermaak, J., C. Andrieu, A. Doucet and S. J. Godsill, “Reversible jump markov chain monte carlo strategies for bayesian model selection in autoregressive processes”, Journal of Time Series Analysis **25**, 6, 785–809 (2004).

Vlahogianni, E. and M. Karlaftis, “Temporal aggregation in traffic data: implications for statistical characteristics and model choice”, Transportation Letters **3**, 1, 37–49 (2011).

Vlahogianni, E. I., M. G. Karlaftis and J. C. Golias, “Short-term traffic forecasting: Where we are and where were going”, Transportation Research Part C: Emerging Technologies **43**, 3–19 (2014).

Wahba, G. and P. Craven, “Smoothing noisy data with spline functions. estimating the correct degree of smoothing by the method of generalized cross-validation.”, Numerische Mathematik **31**, 377–404 (1978).

Wallis, K. F., “Time series analysis of bounded economic variables”, Journal of Time Series Analysis **8**, 1, 115–123 (1987).

Wand, M. P., J. T. Ormerod, S. A. Padoan and R. Frühwirth, “Mean field variational bayes for elaborate distributions”, Bayesian Analysis **6**, 4, 847–900 (2011).

Williams, B. M. and L. A. Hoel, “Modeling and forecasting vehicular traffic flow as a seasonal stochastic time series process”, Tech. rep. (1999).

Williams, B. M. and L. A. Hoel, “Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results”, Journal of transportation engineering **129**, 6, 664–672 (2003).

Yao, Y.-C., “Estimating the number of change-points via schwarz’ criterion”, Statistics & Probability Letters **6**, 3, 181–189 (1988).

Yuan, M. and Y. Lin, “Efficient empirical Bayes variable selection and estimation in linear models”, Journal of the American Statistical Association **100**, 472, 1215–1225, URL `http://dx.doi.org/10.1198/016214505000000367` (2005).

Yuan, M. and Y. 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).

Zhang, N., Y. Zhang and H. Lu, “Seasonal autoregressive integrated moving average and support vector machine models: Prediction of short-term traffic flow on freeways”, Transportation Research Record: Journal of the Transportation Research Board , 2215, 85–92 (2011).

Zhao, P. and B. Yu, “On model selection consistency of lasso”, Journal of Machine learning research **7**, Nov, 2541–2563 (2006).

Zivot, E. and J. Wang, “Modelling financial time series with s-plus”, Springer pp. 429–478 (2006).

Zou, H., “The adaptive lasso and its oracle properties”, Journal of the American statistical association **101**, 476, 1418–1429 (2006).

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

Zou, H., T. Hastie, R. Tibshirani *et al.*, “On the degrees of freedom of the lasso”, The Annals of Statistics **35**, 5, 2173–2192 (2007).

Zou, H. and H. H. Zhang, “On the adaptive elastic-net with a diverging number of parameters”, Annals of statistics **37**, 4, 1733 (2009).

APPENDIX A

R CODE FOR CHAPTER 2

```
#####################################################################################
#Paper:"BAYESIAN SHRINKAGE ESTIMATES OF LOGISTIC SMOOTH TRANSITION AUTOREGRESSIONS"
#Authors:Mario Giacomazzo (Arizona State University)
#        Yiannis Kamarianakis (Arizona State University)
#Year:2017
#
#Comments: This code requires a working installation of JAGS by Martyn Plummer
#####################################################################################

###################
#Required R Packages
###################
library(runjags) #Needed for Calling JAGS through R
library(doParallel) #Needed for Parallelization of MCMC chains
library(datasets) #Contains Sunspot DATA
library(tsDyn) #Used for Frequentist Estimation of LSTAR Models

#########################################
#Simulate 100 Replicates of LSTAR(3) Model
#########################################

#Specify Autoregressive (AR) Coefficients in Low and High Regimes
##Regime Specific AR models must reflect Stationarity Within Regimes
a0<-0
a1<-0
a2<-0
a3<--0.6
b0<-0.02
b1<-0
b2<-0
b3<-0.75

#Specify Transition Function Parameters
slope<-120
thresh<-0.02
delay<-2

#Specify Standard Deviation of Error Term
sigma<-0.02

#Simulation Information
S=3 #Number of Replications
N=1000 #Length of Each Simulated Time Series Desired
burn=2000 #Burn-in Size for Each Replication

#Function Used to Generate 1 Replication of LSTAR Model
generate.func<-function(x){
  set.seed(x) #Needed to Obtain Different Replications That Are Reproducible
  y=rnorm((N+burn),0,0.02) #Initialize Time Series
  e=rnorm((N+burn),0,sigma) #Create iid Errors
  for(i in 4:(N+burn)){
    wt=1/(1+exp(-slope*(y[i-delay]-thresh)))
    y[i]=(a0+a1*y[i-1]+a2*y[i-2]+a3*y[i-3])*(1-wt)+
      (b0+b1*y[i-1]+b2*y[i-2]+b3*y[i-3])*wt+e[i]
  }
  return(y[-(1:burn)]) #Output Time Series After Beginning Burn-in Period
}

all.data<-lapply(1:S,generate.func) #Create List of Many Replications

################################################
#Function Required for Obtaining Lagged Time Series
################################################
lag.func<-function(x,k=1){
  t=length(x)
  y=c(rep(NA,t))
  for(i in (k+1):t){
    y[i]=x[i-k]
  }
```

```
  return(y)
}

##################################################
#Necessary Elements for MCMC Sampling for JAGS
#Used to Model LSTAR(4) Model
#Using Bayesian Horseshoe Priors for AR Parameters
#With Dirichlet Prior for Threshold Variable
##################################################

#Function Used to Obtain Replication Specific Data
datafunction<-function(i){
  y=all.data[[i]]
  X=matrix(NA,nrow=length(y),ncol=4)
  for(j in 1:4){
    X[,j]=lag.func(y,k=j)
  }
  X=cbind(1,X)
  return(list(y=all.data[[i]], #The i-th Replicated Time Series
              #Model Matrix For Each Regime
              #(Made up of Lags 1 to 4 of Endogenous Series)
              X=X,
              N=length(all.data[[i]]), #Length of Time Series
              #Minimum Hyperparameter for Uniform Prior on Threshold
              min.thresh=quantile(all.data[[i]],0.15),
              #Maximum Hyperparameter for Uniform Prior on Threshold
              max.thresh=quantile(all.data[[i]],0.85),
              #Standard Deviation of Endogenous Time Series
              #Used to Scale Slope for Threshold Variable
              sdy=sd(y),
              #Matrix Containing All Delays Considered for Threshold Variable
              X2=X[,-1],
              #Hyperparameter for Dirichlet Distribution
              #(length must equal number of columns in X2;
              #   elements must sum to 1)
              prop.prior=c(.25,.25,.25,.25)))

}

#JAGS Model Represented as a String
#(Horseshoe Priors are Used for Shrinkage and
#Dirichlet Used for Threshold Variable)
#Notice: We do not Monitor Tuning Parameters and
#We Monitor the Raw Unscaled Slope
MOD<-"model{
    #Likelihood Function (Starts at p+1 Which in Our Case is 5)
    for(i in 5:N){
y[i]~dnorm(mu[i],tau)
w[i]<-1.0/(1.0+exp(-(preslope/sdy)*(inprod(prop[],X2[i,])-thresh)))
mu[i]<-(inprod(alpha[],X[i,]))*(1.0-w[i])+(inprod(beta[],X[i,]))*(w[i])
}

tau~dgamma(.001,.001) #Prior for Error Precision
preslope~dlnorm(3,1) #Prior for Scaled Transition Slope Parameter
                     #(Can Use Gamma, Truncated Normal, etc.)
thresh~dunif(min.thresh,max.thresh) #Prior for Threshold Variable
#Prior for Weights of Linear Combination of Possible Threshold Variables
prop~ddirch(prop.prior)

global.squared<-global^2 #Global Shrinkage Parameter

for(k in 1:5){
#Local Shrinkage Parameters for Low Regime
local1.squared[k]<-(local1[k])^2
#Local Shrinkage Parameters for High Regime
local2.squared[k]<-(local2[k])^2
#Priors for Shrinkage Parameters for Low Regime
```

```
alpha[k]~dnorm(0,tau/(global.squared*local1.squared[k]))
#Priors for Shrinkage Parameters for High Regime
beta[k]~dnorm(0,tau/(global.squared*local2.squared[k]))
}

#Bayesian Global-Local Prior Hierarchy Using Half-Cauchy Distributions
#t-Distribution with 1 df -> Cauchy
#T(0,) Truncates Distribution from 0 to Infinity
for(k in 1:5){
local1[k]~dt(0,1,1)T(0,)
local2[k]~dt(0,1,1)T(0,)
}
global~dt(0,1,1)T(0,)

#Find Raw Unscaled Transition Slope Parameter
slope<-preslope/sdy

#modules# runjags
#monitor# tau, slope, thresh, alpha, beta, prop
}
"

#Function Used to Obtain Initial Values Needed for All Parameters
#Different Initial Values for Different Chains for AR Parameters
initsfunction<-function(chain){
  tau<-c(0.01,20,100)[chain]
  preslope<-c(20,50,100)[chain]
  thresh<-0.02
  prop<-c(0.25,.25,0.25,0.25)
  set.seed(chain)
  alpha<-rnorm(5,0,1)
  global<-0.5
  local1<-rep(0.5,5)
  local2<-rep(0.5,5)
  set.seed(chain+1)
  beta<-rnorm(5,0,1)
  .RNG.seed<-c(1,2,3)[chain]
  .RNG.name<-c("base::Super-Duper","base::Wichmann-Hill","base::Super-Duper")
  return(list(tau=tau,preslope=preslope,thresh=thresh,prop=prop,
              alpha=alpha,beta=beta,local1=local1,local2=local2,global=global,
              .RNG.seed=.RNG.seed,.RNG.name=.RNG.name))
}

#################################################
#MCMC Posterior Sampling for Each Replication
#################################################

#Parallelization is Used Across the Many Replications Using Foreach Package
cl2<-makeCluster(1) #Number of Clusters if Access to >3 Cores
registerDoParallel(cl2)

#Foreach Package Outputs as a List Where Each Element is a Different Replication
hs.out=foreach(v=1:S,.packages=c("runjags","parallel")) %dopar%{
  #Parallelization Used Also for Different MCMC Chains
  cl<-makeCluster(3)
  #Initialize JAGS Model for Specific Replication Using 3 Chains
  model2<-run.jags(MOD,data=datafunction(v),n.chains=3,inits=initsfunction,
             mutate=list(prec2sd, 'tau'),adapt=5000,burnin=10000,sample=1000,
             thin=10,method="rjparallel",method.options=list(cl=cl))
  #Obtain Initial Maximum PSRF Convergence Statistic
  #for Chain Convergence Across All Parameters
  max.psrf=max(summary(model2)[,"psrf"],na.rm=T)
  #Obtain Initial Minimum Effective Sample Size Across All Saved Parameters
  min.ess=min(summary(model2)[,"SSeff"],na.rm=T)
  i=1 #Identify This as Initialized Model

  #If Convergence is not Met, Then Update Model With 1000*i samples
  #Repeat Until Convergence is Met or 20 updates have occurred
```

```
  #(Could Take a Long Time)
  #The Maximum Number of Updates is currently 20 but may be reduced
  while((max.psrf>1.05|min.ess<150)&i<20){
    model2<-extend.jags(model2,adapt=1000,burnin=0,
                        sample=1000*i,silent.jags=T)
    max.psrf=max(summary(model2)[,"psrf"],na.rm=T)
    min.ess=min(summary(model2)[,"SSeff"],na.rm=T)
    i=i+1
    print(max.psrf)

  }
  stopCluster(cl)

  #For each replication we output a list containing the final model,
  #convergence results and total computation time
  out=list(model2=model2,max.psrf=max.psrf,min.ess=min.ess,
           time=round(as.numeric(model2$timetaken)/60,1))
}
stopCluster(cl2)

####################################################
#Analyzing Output From Simulation Study
####################################################

#True Parameters for Simulated Nonlinear LSTAR(3) Under Assumption that p=4
true<-c(0,0,0,-0.6,0,0.02,0,0,0.75,0,0.02,120,0.02)
true.delay2<-c(0,1,0,0)

#Function to Output final PSRF Statistic Which Determines
#if Convergence was Met for Each Replication
conv.func<-function(x){
  return(x$max.psrf)
}

#Check Convergence

#Obtain Max PSRF for Each Replication
hsdlp2.conv=unlist(lapply(hs.out,conv.func))
#Check Which Replications Converged
id.hsdlp2.conv=which(hsdlp2.conv<1.05)
#Calculate Convergence Percentage
hsdlp2.per=length(id.hsdlp2.conv)/100


#Check # of Samples Required For Replications Where Convergence Was Met
hsdlp2.samples=rep(NA,100)
for (k in 1:100){
  #Obtain Number of Samples Required For Convergence For Each Replication
  hsdlp2.samples[k]=hs.out[[k]]$model2$sample
}
#Replace Number of Samples with NA For Replications that Didn't Converge
hsdlp2.samples[-id.hsdlp2.conv]=NA


#Get Tables of Posterior Estimates of Nonlinear Parameters
HSDLP2.EST.PARAMS=matrix(NA,100,13)
HSDLP2.EST.THVAR=matrix(NA, 100,4 )
for (k in 1:100){
  HSDLP2.EST.PARAMS[k,]=summary(hs.out[[k]]$model2)[c(4:13,18,2:3),"Mean"]
  HSDLP2.EST.THVAR[k,]=summary(hs.out[[k]]$model2)[14:17,"Mean"]
}

#Plotting the Threshold Variable for The Second Threshold Variable
png(file="hsthvar2.png",height=600,width=600)
z1=c(0,1,0,-1)
z2=c(1,0,-1,0)
par(mar=c(1.1,1.1,1.1,1.1))
```

```
plot(z1,z2,plot="n",pch=".",xlim=c(-1.2,1.2),ylim=c(-1.2,1.2),
      xaxt="n",yaxt="n",xlab="",ylab="",bty="n")
points(x=1,y=0,pch=16,col="black",add=T)
polygon(z1,z2,border="black")
polygon(z1/2,z2/2,border="black")
polygon(z1/4,z2/4,border="black")
polygon(3*z1/4,3*z2/4,border="black")
text(0,1.1,expression(y[t-1]),cex=2,col="black")
text(1.18,0,expression(y[t-2]),cex=2,col="black")
text(0,-1.1,expression(y[t-3]),cex=2,col="black")
text(-1.16,0,expression(y[t-4]),cex=2,col="black")
text(-0.18,0.18,0.25,col="black")
text(-0.31,0.31,0.5,col="black")
text(-0.44,0.44,0.75,col="black")
text(-0.58,0.58,"1.00",col="black")

for(k in id.hsdlp2.conv){
  x=HSDLP2.EST.THVAR[k,]
  x1=x
  y1=x
  x1[1]=0
  y1[2]=0
  x1[3]=0
  y1[3]=-x[3]
  x1[4]=-x[4]
  y1[4]=0
  polygon(x1,y1,col=rgb(176/255,48/255,96/255,0.1),border=NA)
}
dev.off()

#Obtain RMSE for Each Parameter Obtained
#Using Posterior Estimates Across All Simulations
rmse.func<-function(x.est){
  x.sqdiff=(x.est-true)^2
  return(x.sqdiff)
}
HSDLP2.RMSE=sqrt(rowMeans(apply(HSDLP2.EST[id.hsdlp2.conv,],1,rmse.func)))

##################################################
#Practical Application to Annual Sunspot Numbers
##################################################
#Import Training and Testing Datasets of Sunspots
ss.year=window(sunspot.year,start=1700, end=1748)
ss.month1=window(aggregate(sunspot.month, FUN=mean),start=1749,end=1979)
#Create Training Dataset Using Years 1700-1979
Train=c(as.vector(ss.year),as.vector(ss.month1))
#Create Testing Dataset Using Years 1980-2006
Test=window(aggregate(sunspot.month, FUN=mean),start=1980,end=2006)

#Apply Classical Square Root Transformation To Train and Test Data
Train.transform=2*(sqrt(Train+1)-1)
Test.transform=2*(sqrt(Test+1)-1)

###################################################
#Necessary Elements for MCMC Sampling With JAGS
#Specific To Sunspot Data Using Both
#Bayesian Lasso and Bayesian Horseshoe Priors
#Along with Dirichlet Prior for Threshold Variable
###################################################
#General Data Function Data Function
datafunction<-function(){
  y=Train.transform
  X=matrix(NA,nrow=length(y),ncol=10)
  for(j in 1:10){
    X[,j]=lag.func(y,k=j)
  }
  X=cbind(1,X)
```

```
  return(list(y=y,X=X,N=length(y),
              min.thresh=quantile(y,0.15),
              max.thresh=quantile(y,0.85),
              sdy=sd(y)*sqrt(length(y)-1)/sqrt(length(y))))
}

#Bayesian Lasso Model With Regime Specific Shrinkage Parameters
MOD1<-"
model{
for(i in 11:N){
y[i]~dnorm(mu[i],tau)
w[i]<-1.0/(1.0+exp(-(preslope/sdy)*(y[i-2]-thresh)))
mu[i]<-(inprod(alpha[],X[i,]))*(1.0-w[i])+(inprod(beta[],X[i,]))*(w[i])
}

tau~dgamma(.001,.001)
preslope~dlnorm(3,1)
thresh~dunif(min.thresh,max.thresh)

for(k in 1:11){
alpha[k]~dnorm(0,tau*alphatau[k])
beta[k]~dnorm(0,tau*betatau[k])
}

lt1<-lambda1.squared/2
lt2<-lambda2.squared/2

for(k in 1:11){
alphatau[k]~dexp(lt1)
betatau[k]~dexp(lt2)
}
#Gamma Prior for Low Regime Shrinkage Parameter
lambda1.squared~dgamma(1,1.78)
#Gamma Prior for High Regime Shrinkage Parameter
lambda2.squared~dgamma(1,1.78)

#modules# runjags
#monitor# tau,preslope,thresh,alpha,beta
}
"
initsfunction1<-function(chain){
  tau<-c(0.01,20,100)[chain]
  preslope<-c(5,10,15)[chain]
  thresh<-10.8
  set.seed(chain)
  alpha<-rnorm(11,0,1)
  set.seed(chain+1)
  beta<-rnorm(11,0,1)
  alphatau<-rgamma(11,.01,.01)
  betatau<-rgamma(11,.01,.01)
  lambda1.squared=0.67^2
  lambda2.squared=0.67^2
  .RNG.seed<-c(1,2,3)[chain]
  .RNG.name<-c("base::Super-Duper","base::Wichmann-Hill","base::Super-Duper")
  return(list(tau=tau,preslope=preslope,thresh=thresh,
             alpha=alpha,beta=beta,alphatau=alphatau,betatau=betatau,
             lambda1.squared=lambda1.squared,lambda2.squared=lambda2.squared,
             .RNG.seed=.RNG.seed,.RNG.name=.RNG.name))
}

#Bayesian Horseshoe Model
MOD2<-"
model{
for(i in 11:N){
y[i]~dnorm(mu[i],tau)
w[i]<-1.0/(1.0+exp(-(preslope/sdy)*(y[i-2]-thresh)))
```

```
mu[i]<-(inprod(alpha[],X[i,]))*(1.0-w[i])+(inprod(beta[],X[i,]))*(w[i])
}

tau~dgamma(.001,.001)
preslope~dlnorm(3,1)
thresh~dunif(min.thresh,max.thresh)

global.squared<-global^2

for(k in 1:11){
local1.squared[k]<-(local1[k])^2
local2.squared[k]<-(local2[k])^2
alpha[k]~dnorm(0,tau/(global.squared*local1.squared[k]))
beta[k]~dnorm(0,tau/(global.squared*local2.squared[k]))
}

for(k in 1:11){
local1[k]~dt(0,1,1)T(0,)
local2[k]~dt(0,1,1)T(0,)
}

global~dt(0,1,1)T(0,)

#modules# runjags
#monitor# tau, preslope, thresh, alpha, beta
}
"

#Simulation Setting Starting value for
initsfunction2<-function(chain){
  tau<-c(0.01,20,100)[chain]
  preslope<-c(5,10,15)[chain]
  thresh<-10.8
  set.seed(chain)
  alpha<-rnorm(11,0,1)
  global<-0.5
  local1<-rep(0.5,11)
  local2<-rep(0.5,11)
  set.seed(chain+1)
  beta<-rnorm(11,0,1)
  .RNG.seed<-c(1,2,3)[chain]
  .RNG.name<-c("base::Super-Duper","base::Wichmann-Hill","base::Super-Duper")
  return(list(tau=tau,preslope=preslope,thresh=thresh,
              alpha=alpha,beta=beta,local1=local1,local2=local2,global=global,
              .RNG.seed=.RNG.seed,.RNG.name=.RNG.name))
}

#Loop Through the Two Different Models
MOD<-list(MOD1,MOD2)
INITS<-list(initsfunction1,initsfunction2)
#If you have more than 6 cores available, change the number of clusters to 2
cl2<-makeCluster(1)
registerDoParallel(cl2)

sunspot.out=foreach(v=1:2,.packages=c("runjags","parallel")) %dopar%{
  cl<-makeCluster(3)
  model2<-run.jags(MOD[[v]],data=datafunction(),n.chains=3,inits=INITS[[v]],
                   mutate=list(prec2sd, 'tau'),adapt=10000,
                   burnin=40000,sample=1000,thin=10,
                   method="rjparallel",method.options=list(cl=cl))
  max.psrf=max(summary(model2)[,"psrf"])
  min.ess=min(summary(model2)[,"SSeff"])
  i=1

  while((max.psrf>1.05|min.ess<150)&i<20){
    model2<-extend.jags(model2,adapt=1000,burnin=0,sample=1000*i,
```

```
                        silent.jags=T)
    max.psrf=max(summary(model2)[,"psrf"])
    min.ess=min(summary(model2)[,"SSeff"])
    i=i+1
    print(max.psrf)

  }
  stopCluster(cl)
  out1=list(model2=model2,max.psrf=max.psrf,min.ess=min.ess,
            time=round(as.numeric(model2$timetaken)/60,1))
}
stopCluster(cl2)

##########################################################
#Plot Data From Training and Test Set (Raw and Transformed)
##########################################################
All.Data=c(Train,Test)
All.Data.transform=c(Train.transform,Test.transform)

png(file="AnnualSunspot.png",height=600,width=850)
par(mfrow=c(2,1))
plot(1700:2006,All.Data,type="l",ylab="",xlab="Year",
     main="Annual Sunspot Number")
points(1980:2006,Test,col="red",type="l")
plot(1700:2006,All.Data.transform,type="l",ylab="",xlab="Year",
     main="Square Root Transformed Annual Sunspot Number")
points(1980:2006,Test.transform,col="red",type="l")
dev.off()

##########################################################
#Get Posterior Estimates from Both Models
##########################################################
SUNSPOT.ESTIMATES<-matrix(NA,ncol=2,nrow=26)
for(k in 1:2){
  #For Bayesian Lasso
  if (k==1){
    SUNSPOT.ESTIMATES[1:25,k]=
            summary(sunspot.out[[k]]$model2)[c(4:26,2,3),"Median"]
  }
  #For Bayesian Horseshoe
  if( k==2){
    SUNSPOT.ESTIMATES[1:25,k]=
            summary(sunspot.out[[k]]$model2)[c(4:26,2,3),"Mean"]
  }
  SUNSPOT.ESTIMATES[26,k]=sunspot.out[[k]]$model2$sample
}



############################################################################
#Functions Required For Recursive Forecasts
#Using a Rolling Window Without Reestimation
#Using the Bootstrap Method for Nonlinear Model Forecasting
############################################################################

#Function Specific For Obtaining a One Step Ahed Forecast
OneStep.func<-function(params,data,time,s=sd(Train.transform)){
  data2=c(1,data[(time-1):(time-10)])
  pred=(data2%*%params[1:11])*(1-
    (1/(1+exp(-(params[24]/s)*(data2[3]-params[25])))))+
    (data2%*%params[12:22])*(1/(1+exp(-(params[24]/s)*(data2[3]-params[25]))))
  return(pred)
}

#Function That Loops Through the Data Using OneStep.func for each time
#Train.Data is used to obtain residuals for
```

```
#Bootstrapped Sampling Errors for Forecasts
MultiStep.func<-function(params, train.data, test.data,
                         s=sd(Train.transform), n.ahead){
  #n.ahead specifies how many time periods you would like to forecast ahead
  full.data=c(train.data, test.data, rep(NA, n.ahead))
  n.used=length(c(train.data, test.data))
  n.full=length(full.data)

  X.left=matrix(NA, nrow=length(train.data), ncol=10)
  for (j in 1:10){
    X.left[,j]=lag.func(train.data, k=j)
  }
  X.left=cbind(1,X.left)*(1-(1/(1+exp(-(params[24]/s)*
          (lag.func(train.data, k=2)-params[25])))))
  X.right=matrix(NA, nrow=length(train.data), ncol=10)
  for (j in 1:10){
    X.right[,j]=lag.func(train.data, k=j)
  }
  X.right=cbind(1,X.right)*(1/(1+exp(-(params[24]/s)*
          (lag.func(train.data, k=2)-params[25]))))
  X=cbind(X.left,X.right)
  predict=X%*%c(params[1:22])
  resid=train.data-predict
  resid.sample=sample(na.omit(resid), size=n.ahead, replace=T)

  #Forecast for long horizons using previous forecast plus random  noise
  for(i in (n.used+1):(n.used+n.ahead)){
    full.data[i]=OneStep.func(params=params, data=full.data, time=i)
    #Add randomly selected value from resampling of errors
    #from Training Data (Bootstrap Forecasts)
    full.data[i]=full.data[i]+resid.sample[i-n.used]
  }
  return(full.data[(n.used+1):n.full])
}

#Function Used for Bootstrapping to Obtain the Pseudo-Distribution
#Of Predictions for a Specific Horizon And Can be Modified
#to Also Output Specific Forecast Quantiles
Forecast.func<-function(boot, params, train.data, test.data,
                        s=sd(Train.transform), n.ahead){
  #boot=Number of Forecasts for Each Time Period
  boot.reps=replicate(boot, MultiStep.func(params=params,
                                            train.data=train.data,
                                            test.data=test.data,
                                            n.ahead=n.ahead))
  #Point forecast is the mean across all Bootstrap Sampled Forecasts

  #Used if Forecasting One Step Ahead
  if(is.null(dim(boot.reps))) forecast=mean(boot.reps, na.rm=T)
  #Used if Forecasting More than One step Ahead
  if(!is.null(dim(boot.reps))) forecast=rowMeans(boot.reps, na.rm=T)
  return(forecast)
}

###############################################################################
#Obtaining Forecasts Using Bayesian Lasso
#for Horizons 1 to 5 on Transformed Test Data
#Calculating RMSFE for all Horizons by Comparing Truth to Forecasts
###############################################################################
BLASSO.FORECASTS.12345=matrix(NA, nrow=length(Test.transform), ncol=5)
for(j in 1:5){
  for(k in j:(length(Test.transform))){
    if((j-k)==0){
      BLASSO.FORECASTS.12345[k,j]=Forecast.func(boot=500,
                                                params=SUNSPOT.ESTIMATES[,1],
                                                train.data=Train.transform,
                                                test.data=NULL,
```

```
                                               n.ahead=j)[j]
    }else{
      BLASSO.FORECASTS.12345[k,j]=Forecast.func(boot=500,
                                               params=SUNSPOT.ESTIMATES[,1],
                                               train.data=Train.transform,
                                               test.data=Test.transform[1:(k-j)],
                                               n.ahead=j)[j]
    }
  }
}

RMSFE1=rep(NA,5)
for(k in 1:5){
  #Other Metrics for Forecast Evaluation Can be Replaced Here
  RMSFE1[k]=sqrt(mean((Test.transform-BLASSO.FORECASTS.12345[,k])^2,na.rm=T))
}

###############################################################################
#Obtaining Forecasts Using Bayesian Horseshoe
#for Horizons 1 to 5 on Transformed Test Data
#Calculating RMSFE for all Horizons by Comparing Truth to Forecasts
###############################################################################
BHS.FORECASTS.12345=matrix(NA,nrow=length(Test.transform),ncol=5)
for(j in 1:5){
  for(k in j:(length(Test.transform))){
    if((j-k)==0){
      BHS.FORECASTS.12345[k,j]=Forecast.func(boot=500,
                                             params=SUNSPOT.ESTIMATES[,2],
                                             train.data=Train.transform,
                                             test.data=NULL,
                                             n.ahead=j)[j]
    }else{
      BHS.FORECASTS.12345[k,j]=Forecast.func(boot=500,
                                             params=SUNSPOT.ESTIMATES[,2],
                                             train.data=Train.transform,
                                             test.data=Test.transform[1:(k-j)],
                                             n.ahead=j)[j]
    }
  }
}

RMSFE2=rep(NA,5)
for(k in 1:5){
  RMSFE2[k]=sqrt(mean((Test.transform-BHS.FORECASTS.12345[,k])^2,na.rm=T))
}
```

APPENDIX B

BAYESIAN REGULARIZATION OF DISTRIBUTED LAG MODELS

WE

B.1 Introduction

Given a linear regression model represented by $\boldsymbol{y} = \mu + \boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{\epsilon}$, Bayesian regularization methods were developed to achieve sparsity in the posterior estimate $\hat{\boldsymbol{\theta}}$ when only a subset of the variables in $\boldsymbol{X}$ are considered important *a priori*. Often in practice, information criteria, i.e. AIC, BIC, DIC, or posterior model probabilities, help discriminate between various submodels obtained through re-estimation. Stepwise algorithms are helpful but limited when the set of covariates in $\boldsymbol{X}$ in large. Bayesian regularization methods are computationally efficient and bypass the need to explore the entire model space. Mallick and Yi (2013) provide a detailed comparison of Bayesian and frequentist variable selection in high dimensional linear models.

Many parametric time series models have a linear matrix form. This includes models with autoregressive, moving average, distributed lag, and exogenous predictors. Beyond the coefficients, order parameters describe the extent to which historical information is relevant for prediction. For cross-sectional studies, the dimensionality of $\boldsymbol{\theta}$ for the full model is fixed by the number of available explanatory variables. In the time series context, this dimensionality is a parameter itself. The reversible jump Markov Chain Monte Carlo (RJMCMC) technique of Green (1995) is capable of simultaneously sampling from the posterior distribution for unknown orders and updating the dimension of the full model. This approach has been seen in models defined by a single order parameter p such as autoregressive models AR(p) (Troughton and Godsill, 1997; Vermaak *et al.*, 2004), threshold autoregressive models TAR(p) (Campbell, 2004), and smooth transition autoregressive models STAR(p) (Lopes and Salazar, 2006). For a model with multiple orders, i.e. autoregressive moving average model ARMA(p, q), RJMCMC becomes less easy to implement.

Subset models can also be represented by $\boldsymbol{y} = \mu + \boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{\epsilon}$ where the true $\boldsymbol{\theta}$ is sparse. By itself, RJMCMC is incapable of handling this problem. For each covariate θ_i, the uncertainty of relevancy is captured via probalistic spikes at 0 (Mitchell and Beauchamp, 1988). Bernouilli distributed inclusion parameters with discrete mixture priors automate posterior model selection and estimation (George and McCulloch, 1993; Carlin and Chib, 1995; Kuo and Mallick, 1998; Dellaportas *et al.*, 2002). In high dimensional cases where $\boldsymbol{\theta}$ contains many coefficients, exploring the entire model space under this paradigm becomes a computational challenge.

Bayesian regularization techniques approximate sparse estimation by shrinking irrelevant effects to 0. Continuous scale mixture priors concentrated around 0 promote sparsity. Machine learning penalized regression paths, such as ridge (Hoerl and Kennard, 1970), LASSO (Tibshirani, 1996), and elastic net (Hastie *et al.*, 2009), are similar to posterior mean profiles under different hierarchical representations (Hsiang, 1975; Park and Casella, 2008; Li *et al.*, 2010). These methods and many others fall in the class of global-local shrinkage priors (Polson and Scott, 2010).

Misspecification of model orders in time series studies detrimentally impacts forecasting short term forecasting accuracy. Rather than applying distributions to AR, MA, and DL orders, these parameters represent maximum restrictions on the model's complexity and chosen *a priori*. The fixed choices should be large enough to cover all long-term and seasonal effects. This handling introduces many irrelevant lagged covariates in $\boldsymbol{\theta}$, but good shrinkage priors can combat overfitting. Using a simulated distributed lag model (DLM) containing two exogenous time series, the horseshoe prior (BHS) of (Carvalho *et al.*, 2009, 2010) and the extended horseshoe+ (BHS$^+$) of (Bhadra *et al.*, 2016) are effective. Under a very simple data generating process, performance is examined as the assumed order parameters increase beyond the truth. To fully appreciate the signal detection accuracy of BHS and BHS$^+$, the Bayesian LASSO (BLASSO) hierarchy is used as a baseline (Park and Casella, 2008). In Section B.2, the prior hierarchies of BHS, BHS$^+$, and BLASSO are stated. Section B.3 defines the parametric DLM structure and describes its purpose in this context. Monte Carlo experiments comparing all three methods are provided in Section B.4.

B.2 Bayesian Regularization

In the class of global-local scale mixture priors, the horseshoe prior of Carvalho *et al.* (2010) has become a preference in sparse signal estimation. Consider the full linear model $\boldsymbol{y} \sim \mathcal{N}(\mu + \boldsymbol{X}\boldsymbol{\theta}, \sigma^2)$, where $\boldsymbol{\theta} = [\theta_1, \cdots, \theta_i, \cdots, \theta_P]'$. Without loss of generality, $\boldsymbol{X}$ is assumed to be standardized matrix of predictors. Furthermore, Jeffrey's prior is used for the variance and a flat prior for μ. The BHS hierarchy in Equation B.1 is specified for each individual coefficient θ_i where $\mathcal{N}$ and $\mathcal{C}^+$ respectively denote *normal* and *half-Cauchy* distributions.

$$\begin{aligned} \theta_i|\lambda_i, \tau, \sigma^2 &\sim \mathcal{N}(0, \lambda_i^2\tau^2\sigma^2) \\ \lambda_i &\sim \mathcal{C}^+(0,1) \\ \tau &\sim \mathcal{C}^+(0,1) \end{aligned} \tag{B.1}$$

Let $\mathcal{IG}$ denote the *inverse-gamma* distribution. Based on the work by Wand *et al.* (2011), if $\lambda_i^2|\nu_i \sim \mathcal{IG}(\frac{1}{2}, \frac{1}{\nu_i})$ and $\nu_i \sim \mathcal{IG}(\frac{1}{2}, 1)$, then $\lambda_i^2 \sim \mathcal{C}^+(0,1)$. Makalic and Schmidt (2016) exploit this scale-mixture decomposition so that posterior sampling of the coefficients can be obtained via Gibbs.

Bhadra *et al.* (2016) extended BHS to the BHS$^+$ hierarchy expressed in Equation B.2. BHS$^+$ priors have a shrinkage profile that improves signal detection when $\boldsymbol{\theta}$ is "ultra sparse" or "nearly black." Under $0-1$ loss, both BHS and BHS$^+$ estimation methods are oracle procedures (Datta and Ghosh, 2013; Bhadra *et al.*, 2016). In high dimensional cases, Bhadra *et al.* (2016) proved that the posterior mean squared error is lower under BHS$^+$. The additional vector of tuning parameters $\boldsymbol{\eta} = [\eta_1, \cdots, \eta_i, \cdots, eta_P]'$ naturally increases the computational cost, but the *inverse-gamma* decomposition of the *half-Cauchy* can be utilized in Gibbs sampling.

$$\begin{aligned} \theta_i|\lambda_i, \tau, \sigma^2 &\sim \mathcal{N}(0, \lambda_i^2\tau^2\sigma^2) \\ \lambda_i &\sim \mathcal{C}^+(0,\eta_i) \\ \eta_i &\sim \mathcal{C}^+(0,1) \\ \tau &\sim \mathcal{C}^+(0,1) \end{aligned} \tag{B.2}$$

To serve as a baseline, consider the Bayesian LASSO hierarchy expressed in Equation B.3 where *EXP* denotes the *exponential* distribution and $\mathcal{G}$ denotes the *gamma* distribution (Park and Casella, 2008). The *gamma* prior for τ maintains conjugacy. Hyperparameters a and b should be small to ensure the prior remains unformative. Using posterior modes, BLASSO is capable of simultaneously performing model selection and parameter estimation; however, Castillo *et al.* (2012) showed the full posterior distributions contract at suboptimal rates. Like BHS and BHS^{+}, posterior sampling is extremely efficient using Gibbs.

$$\begin{aligned} \theta_i|\lambda_i,\tau,\sigma^2 &\sim \mathcal{N}(0,\lambda_i^2\sigma^2) \\ \lambda_i^2 &\sim EXP^{+}(\tau) \\ \tau^2 &\sim \mathcal{G}(a,b) \end{aligned} \tag{B.3}$$

B.3 Distributed Lag Model

Suppose our primary interest is the modeling of time series $\{Y_t\}$ using a sample of T realizations $\{y_t : t = 1, \cdots, T\} = \{y_t\}$. Endogenous series $\{A_t\}$ and $\{B_t\}$ sampled concurrently are believed to impact the behavior of $\{Y_t\}$ according to the finite DLM in Equation B.4.

$$Y_t = \mu + \sum_{j=1}^{P_1} \alpha_j A_{t-j} + \sum_{k=1}^{P_2} \beta_k B_{t-k} + \epsilon_t \tag{B.4}$$

The marginal effect $\{A_t\}$ and $\{B_t\}$ have on $\{Y_t\}$ are distributed across multiple lags. In classic DLMs, A_t and B_t are included; but in time series applications, the forecast $\hat{y}_t$ is unobtainable without first predicting unknown inputs $\hat{a}_t$ and $\hat{b}_t$. Without loss of generality, only past information is included in the DL structure.

Let $\boldsymbol{y} = [y_m, \cdots, y_T]'$, $\boldsymbol{\theta} = [\alpha_1, \cdots, \alpha_{P_1}, \beta_1, \cdots, \beta_{P_2}]'$, $\boldsymbol{\epsilon} = [\epsilon_m, \cdots, \epsilon_T]'$, and

$$\boldsymbol{X} = [\boldsymbol{X}_a, \boldsymbol{X}_b] = \begin{bmatrix} a_{m-1} & \cdots & a_{m-P_1} & b_{m-1} & \cdots & b_{m-P_2} \\ a_m & \cdots & a_{m-P_1+1} & b_m & \cdots & b_{m-P_2+1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{t-1} & \cdots & a_{t-P_1} & b_{t-1} & \cdots & b_{t-P_2} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{T-1} & \cdots & a_{T-P_1} & b_{T-2} & \cdots & b_{T-P_2} \end{bmatrix}.$$

The DLM in Equation B.4 can now be written in matrix form $\boldsymbol{y} = \mu + \boldsymbol{X}\boldsymbol{\theta} + \boldsymbol{\epsilon}$. Ordinary least squares (OLS) regression is the most popular estimation method for linear models. Specifically for DL models, the structure of the model matrix $\boldsymbol{X}$ naturally breeds collinearity. Predictors in the submatrices $\boldsymbol{X}_a$ and $\boldsymbol{X}_b$ are built using lagged values from two separate input time series, and if strong temporal dependency exists within $\{a_t\}$ and $\{b_t\}$, then multicollinearity is invevitable. Furthermore, correlation between $\boldsymbol{X}_a$ and $\boldsymbol{X}_b$ may be induced by strong cross-correlation between $\{a_t\}$ and $\{b_t\}$. The OLS estimate $\hat{\boldsymbol{\theta}}$ remains unbiased but any induced collinearity in the set of $P_1 + P_2$ predictors in $\boldsymbol{X}$ increases the uncertainty regarding this estimate.

Suppose $\{a_t\}$ and $\{b_t\}$ are generated independently by stationary AR(1) processes seen in Equation B.5. The random variables $\{\epsilon_{A,t}\}$ and $\{\epsilon_{B,t}\}$ are uncorrelated Gaussian white noise where $\{\epsilon_{A,t}\} \sim \mathcal{N}(0, \sigma_A^2)$ and $\{\epsilon_{B,t}\}$i.i.d. $\sim \mathcal{N}(0, \sigma_B^2)$.

$$\begin{aligned} a_t &= \phi_A a_{t-1} + \epsilon_{A,t} \\ b_t &= \phi_B b_{t-1} + \epsilon_{B,t} \end{aligned} \tag{B.5}$$

By construction, the correlation $\text{Corr}\{A_{t-j}, B_{t-k}\} \approx 0 \ \forall j, k$ since $\{a_t\}$ and $\{b_t\}$ are generated independently; however, the multicollinearity within matrices $\boldsymbol{X}_a$ and $\boldsymbol{X}_b$ can be approximated using the theoretical autocorrelation function of AR processes. For any $h \in \{1, 2, \cdots\}$, $\text{Corr}\{A_t, A_{t+h}\} \approx \phi_A^h$ and $\text{Corr}\{B_t, B_{t+h}\} \approx \phi_B^h$. In simulation studies, the strength of correlation between predictors can be controlled through specifying ϕ_A and ϕ_B that satisfy regulatory stationary conditions $|\phi_A| < 1$ and $|\phi_B| < 1$.

DL models were popularized in econometrics to explain dynamic temporal relationships between economic variables. These models become more complex when lags of the endogenous series $\{y_t\}$ are included in $\boldsymbol{X}$, infinite representations are used, or nonlinear relationships are explored. The focus of this article is not on new applications of DLMs. The DLM framework is only used to compare BLASSO, BHS, and BHS$^+$ regression methods for different degrees of sparsity and different strengths of multicollinearity. The presented results can be used to guide statisticians towards shrinkage priors in DLMs.

B.4 Monte Carlo Experiment

B.4.1 Simulation Design

Consider the time series $\{y_t\}$ generated according to the DL model in Equation B.6 with Gaussian white noise $\{\epsilon_t\} \sim \mathcal{N}(0, 10)$. Multicollinearity is introduced and controlled by generating $\{a_t\}$ and $\{b_t\}$ using independent AR(1) processes according to Equation B.5 with $\sigma_A^2 = 0.2$ and $\sigma_B^2 = 1$. Simulated series $\{a_t\}$, $\{b_t\}$, and $\{y_t\}$ of length $T \in \{50, 200\}$ are considered.

$$y_t = 80 + 8a_{t-1} - 6a_{t-3} - 8b_{t-2} + 6b_{t-4} + \epsilon_t \tag{B.6}$$

Prior to estimation, assume $P_1 = P_2 = P$ and safely select $P > 4$. The true DL model is a subset of the overparameterized linear regression $\boldsymbol{y} = \mu + \boldsymbol{X\theta} + \boldsymbol{\epsilon}$ where $\boldsymbol{\theta} = [\alpha_1, \cdots, \alpha_P, \beta_1, \cdots, \beta_P]'$, $m = P + 1$, and

$$\boldsymbol{X} = [\boldsymbol{X}_a, \boldsymbol{X}_b] = \begin{bmatrix} a_{m-1} & \cdots & a_{m-P} & b_{m-1} & \cdots & b_{m-P} \\ a_m & \cdots & a_{m-P+1} & b_m & \cdots & b_{m-P+1} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{t-1} & \cdots & a_{t-P} & b_{t-1} & \cdots & b_{t-P} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ a_{T-1} & \cdots & a_{T-P} & b_{T-2} & \cdots & b_{T-P} \end{bmatrix}.$$

For choices of $P \in \{5, 10, 20\}$, BLASSO, BHS, and BHS$^+$ shrinkage priors are applied to each $\theta \in \{\alpha_1, \cdots, \alpha_P, \beta_1, \cdots, \beta_P\}$. The degree of sparsity in $\boldsymbol{\theta}$ increases with the uncertainty around P. After Gibbs sampling, the three methods are evaluated using posterior 10%, 50%, and 90% quantiles abbreviated $\hat{\theta}_{0.1}$, $\hat{\theta}_{0.5}$, and $\hat{\theta}_{0.9}$, respectively.

The high to moderate multicollinearity in $\boldsymbol{X}$ is introduced using combinations of $(\phi_A, \phi_B) \in \{(0.9, 0.9), (0.9, -0.5), (0.5, -0.5)\}$ in the simulation of exogenous information $\{a_t\}$ and $\{b_t\}$. The correlation existing between lagged predictors in $\boldsymbol{X}_a$ and $\boldsymbol{X}_b$ is highest when $\phi_A = 0.9$ and $\phi_B = -0.9$ and lowest when $\phi_A = 0.5$ and $\phi_B = -0.5$.

Experiments are replicated 100 times under all options of T, P, ϕ_A and ϕ_B specified above. All three stages – simulation, estimation, and evaluation – are conducted in **R** (R Core Team, 2017). The **bayesreg** package efficiently implements BLASSO, BHS, and BHS$^+$ methods (Schmidt and Makalic, 2016). Posterior inference is based on $S = 1000$ posterior samples from $p(\mu, \boldsymbol{\theta}, \sigma^2 | \{a_t\}, \{b_t\}, \{y_t\})$ starting after a burn-in period of 5000. To reduce autocorrelation within MCMC chains, only every fifth sample is retained.

B.4.2 Comparing Methods on the Parameter Level

For each scenario, 100 posterior medians $\{\hat{\boldsymbol{\theta}}_{0.5}^{(1)}, \hat{\boldsymbol{\theta}}_{0.5}^{(2)}, \cdots, \hat{\boldsymbol{\theta}}_{0.5}^{(100)}\}$ are obtained under each Bayesian regularization method. Across all replications, the estimation consistency for each lag weight parameter is measured using root mean squared error (RMSE). Given weight $\theta \in \{\alpha_1, \cdots, \alpha_P, \beta_1, \cdots, \beta_P\}$ and its corresponding estimate $\hat{\theta}_{0.5}$, define RMSE(θ) according to Equation B.7.

$$\text{RMSE}(\theta) = \sqrt{\frac{1}{100} \sum_{r=1}^{100} (\theta^{(r)} - \hat{\theta}_{0.5}^{(r)})^2} \tag{B.7}$$

Given P, there are $2P$ lag weights estimated in each replicated experiment. Since the dimensionality of θ changes with P, the results are divided accordingly. Table B.1 presents results for $P = 5$ and Table B.2 for $P = 10$. When $P = 20$, the vector $\boldsymbol{\theta}$ contains 40 predictors; therefore, results are also split based on series length $T = 50$ and $T = 200$ in Tables B.3 and B.4, respectively.

As expected, RMSE decreases when the length of the time series increases. For each combination of P and T, the linear DLM regression is based on $T - P$ observations. This means that estimation is based on a limited 30 joint observations $[y_t, a_{t-1}, \cdots, a_{t-20}, b_{t-1}, \cdots, b_{t-20}]$ in the extreme case when $T = 50$ and $P = 20$. Naturally, RMSE is highest for this situation. Lowest RMSE occurs for $T = 200$ and $P = 5$. This pattern is consistent for all Bayesian estimation methods and the three different pairings of (ϕ_A, ϕ_B).

Interestingly, there is no consistent pattern for the change in RMSE in the move from high to moderately correlated predictors. The strength of correlation within $\boldsymbol{X}_a$ and $\boldsymbol{X}_b$ are approximately equivalent when $(\phi_A, \phi_B) \in \{(0.9, -0.9), (0.5, -0.5)\}$. The RMSE($\alpha$) > RMSE($\beta$) indicating better estimation of the lag weights for $\{b_t\}$. This phenomena results from the generation of $\{a_t\}$ and $\{b_t\}$ where $0.2 = \sigma_A^2 \neq \sigma_B^2 = 1$. The option to generate input series that perfectly follow AR(1), i.e. $\sigma^2 = 0$, was a consideration, but this does not reflect real life usage of DLMs.

Most importantly, BLASSO underperforms both BHS and BHS$^+$ for all lag weights. This difference in RMSE is largest when $P = 20$. All methods are computationally equivalent, but BLASSO does a poor job estimating sparse $\boldsymbol{\theta}$. The RMSE(θ^*)

for nonzero $\theta^* \in \{\alpha_1, \alpha_3, \beta_2, \beta_4\}$ is considerably larger than RMSE for truly irrelevant lag weights. In terms of percentage change, the reduction in RMSE(θ^*) when horseshoe priors are utilized is not as staggering as the reduction seen for the zero parameters. However, for the ultra-sparse case $P = 20$, this discrepancy is more apparent across all θ. Methods BHS and BHS$^+$ almost cut RMSE(θ) in half in this extreme case regardless of the strength of correlation. Although the uncertainty around $\theta^* \in \{\alpha_1, \alpha_3, \beta_2, \beta_4\}$ increases with P, posteriors under BHS and BHS$^+$ lead to better identification of zero coefficients when more are considered in the model.

Also, there is general advantage of using BHS$^+$ over BHS, especially when data is limited and the dimension of $\boldsymbol{\theta}$ is large. This observation is not caused by changes in correlation strength but more to the change in sample size. In all cases, the BHS$^+$ hierarchy is recommended over the BHS hierarchy if the increase in computational time is reasonable for the user.

Table B.1: Parameter RMSE Comparison When $P = 5$

		Multicollinearity Control: (ϕ_A, ϕ_B)								
		$(0.9, -0.9)$			$(0.9, -0.5)$			$(0.5, -0.5)$		
	Truth	BLASSO	BHS	BHS$^+$	BLASSO	BHS	BHS$^+$	BLASSO	BHS	BHS$^+$
$T = 50$	$\alpha_1 = 8$	1.18	1.12	1.09	1.13	1.04	0.98	1.25	1.20	1.15
	$\alpha_2 = 0$	1.25	0.88	0.72	1.28	1.00	0.88	1.33	0.99	0.83
	$\alpha_3 = -6$	1.56	1.46	1.37	1.48	1.45	1.35	1.45	1.40	1.33
	$\alpha_4 = 0$	1.38	0.88	0.67	1.20	0.89	0.73	1.09	0.71	0.54
	$\alpha_5 = 0$	1.02	0.67	0.51	0.92	0.62	0.49	0.92	0.61	0.48
	$\beta_1 = 0$	0.52	0.35	0.28	0.49	0.35	0.27	0.54	0.38	0.31
	$\beta_2 = -8$	0.72	0.56	0.51	0.68	0.55	0.50	0.68	0.53	0.47
	$\beta_3 = 0$	0.57	0.37	0.29	0.54	0.39	0.31	0.57	0.39	0.31
	$\beta_4 = 6$	0.65	0.49	0.44	0.55	0.46	0.42	0.59	0.48	0.44
	$\beta_5 = 0$	0.41	0.28	0.22	0.43	0.31	0.25	0.34	0.22	0.16
$T = 200$	$\alpha_1 = 8$	0.76	0.68	0.64	0.72	0.63	0.60	0.77	0.69	0.66
	$\alpha_2 = 0$	0.96	0.68	0.55	0.87	0.60	0.48	0.92	0.69	0.59
	$\alpha_3 = -6$	1.02	0.91	0.83	1.09	0.96	0.88	0.99	0.89	0.82
	$\alpha_4 = 0$	0.96	0.65	0.51	0.84	0.54	0.41	0.85	0.56	0.43
	$\alpha_5 = 0$	0.72	0.46	0.35	0.63	0.39	0.29	0.70	0.47	0.36
	$\beta_1 = 0$	0.30	0.20	0.15	0.33	0.24	0.19	0.34	0.25	0.21
	$\beta_2 = -8$	0.48	0.37	0.32	0.43	0.38	0.35	0.44	0.37	0.35
	$\beta_3 = 0$	0.45	0.31	0.24	0.38	0.28	0.22	0.37	0.26	0.20
	$\beta_4 = 6$	0.48	0.39	0.35	0.40	0.34	0.32	0.41	0.34	0.31
	$\beta_5 = 0$	0.29	0.20	0.16	0.30	0.21	0.16	0.29	0.20	0.15

B.4.3 Comparing Methods on Overall Accuracy

For each replication r and choice of P, the resulting vector of posterior medians $\hat{\boldsymbol{\theta}}_{0.5}^{(r)} = [\hat{\alpha}_{1,0.5}^{(r)}, \cdots, \hat{\alpha}_{P,0.5}^{(r)}, \hat{\beta}_{1,0.5}^{(r)}, \cdots, \hat{\beta}_{P,0.5}^{(r)}]'$ acts as a point estimate of the unknown parameter vector $\boldsymbol{\theta} = [\alpha_1, \cdots, \alpha_P, \beta_1, \cdots, \beta_P]'$. In Section B.4.2, the error between $\hat{\boldsymbol{\theta}}_{0.5}^{(r)}$ and $\boldsymbol{\theta}$ was quantified separately for each parameter using RMSE. Along with $\hat{\boldsymbol{\theta}}_{0.5}^{(r)}$, Bayesian 80% credible regions for $\boldsymbol{\theta}$ constructed from

$$\hat{\boldsymbol{\theta}}_{0.1}^{(r)} = [\hat{\alpha}_{1,0.1}^{(r)}, \cdots, \hat{\alpha}_{P,0.1}^{(r)}, \hat{\beta}_{1,0.1}^{(r)}, \cdots, \hat{\beta}_{P,0.1}^{(r)}]' \text{ and}$$

$$\hat{\boldsymbol{\theta}}_{0.9}^{(r)} = [\hat{\alpha}_{1,0.9}^{(r)}, \cdots, \hat{\alpha}_{P,0.9}^{(r)}, \hat{\beta}_{1,0.9}^{(r)}, \cdots, \hat{\beta}_{P,0.9}^{(r)}]'$$

quantify the posterior uncertainty regarding the unknown lag weights.

Table B.2: Parameter RMSE Comparison When $P = 10$

		Multicollinearity Control: (ϕ_A, ϕ_B)								
		$(0.9, -0.9)$			$(0.9, -0.5)$			$(0.5, -0.5)$		
	Truth	BLASSO	BHS	BHS$^+$	BLASSO	BHS	BHS$^+$	BLASSO	BHS	BHS$^+$
	$\alpha_1 = 8$	1.35	1.26	1.19	1.55	1.32	1.24	1.54	1.37	1.30
	$\alpha_2 = 0$	1.28	0.84	0.76	1.11	0.80	0.76	1.33	0.90	0.82
	$\alpha_3 = -6$	1.82	1.73	1.61	1.83	1.75	1.62	1.71	1.60	1.49
	$\alpha_4 = 0$	1.37	0.77	0.66	1.15	0.79	0.74	1.03	0.60	0.50
	$\alpha_5 = 0$	1.35	0.64	0.51	0.93	0.42	0.33	1.05	0.53	0.42
	$\alpha_6 = 0$	1.21	0.55	0.40	1.05	0.50	0.40	1.18	0.63	0.54
	$\alpha_7 = 0$	1.13	0.51	0.40	0.95	0.45	0.34	1.04	0.48	0.35
	$\alpha_8 = 0$	1.04	0.45	0.33	1.04	0.46	0.37	1.28	0.65	0.50
	$\alpha_9 = 0$	1.13	0.51	0.40	0.94	0.42	0.34	1.15	0.54	0.37
$T = 50$	$\alpha_{10} = 0$	1.13	0.56	0.43	1.00	0.54	0.48	0.92	0.50	0.42
	$\beta_1 = 0$	0.59	0.28	0.23	0.50	0.25	0.20	0.58	0.32	0.27
	$\beta_2 = -8$	0.91	0.55	0.52	0.75	0.50	0.47	0.89	0.58	0.54
	$\beta_3 = 0$	0.53	0.28	0.24	0.54	0.32	0.27	0.53	0.32	0.27
	$\beta_4 = 6$	0.86	0.55	0.51	0.77	0.52	0.48	0.74	0.51	0.48
	$\beta_5 = 0$	0.50	0.21	0.17	0.50	0.25	0.20	0.45	0.22	0.16
	$\beta_6 = 0$	0.55	0.25	0.19	0.44	0.24	0.20	0.48	0.23	0.19
	$\beta_7 = 0$	0.59	0.27	0.20	0.47	0.23	0.18	0.43	0.19	0.15
	$\beta_8 = 0$	0.48	0.21	0.17	0.47	0.24	0.18	0.50	0.23	0.19
	$\beta_9 = 0$	0.54	0.25	0.18	0.54	0.28	0.23	0.38	0.16	0.13
	$\beta_{10} = 0$	0.45	0.21	0.17	0.40	0.23	0.19	0.42	0.25	0.22
	$\alpha_1 = 8$	0.82	0.67	0.65	0.78	0.65	0.63	0.78	0.67	0.65
	$\alpha_2 = 0$	0.89	0.48	0.42	0.75	0.39	0.34	0.89	0.54	0.49
	$\alpha_3 = -6$	1.06	0.85	0.79	1.15	0.88	0.82	1.06	0.86	0.80
	$\alpha_4 = 0$	0.89	0.42	0.35	0.80	0.34	0.27	0.85	0.42	0.35
	$\alpha_5 = 0$	0.80	0.35	0.28	0.68	0.29	0.25	0.79	0.37	0.31
	$\alpha_6 = 0$	0.76	0.31	0.24	0.65	0.24	0.18	0.64	0.27	0.21
	$\alpha_7 = 0$	0.83	0.31	0.23	0.66	0.24	0.19	0.75	0.33	0.27
	$\alpha_8 = 0$	0.82	0.30	0.22	0.67	0.22	0.17	0.62	0.29	0.25
	$\alpha_9 = 0$	0.80	0.27	0.19	0.65	0.24	0.19	0.67	0.28	0.22
$T = 200$	$\alpha_{10} = 0$	0.72	0.31	0.24	0.52	0.23	0.18	0.62	0.30	0.25
	$\beta_1 = 0$	0.30	0.14	0.11	0.33	0.18	0.15	0.36	0.22	0.20
	$\beta_2 = -8$	0.50	0.31	0.30	0.47	0.36	0.34	0.50	0.37	0.36
	$\beta_3 = 0$	0.44	0.23	0.21	0.38	0.21	0.18	0.35	0.19	0.17
	$\beta_4 = 6$	0.55	0.39	0.37	0.45	0.34	0.32	0.46	0.34	0.33
	$\beta_5 = 0$	0.38	0.15	0.12	0.39	0.20	0.16	0.32	0.15	0.12
	$\beta_6 = 0$	0.36	0.15	0.12	0.35	0.16	0.13	0.33	0.16	0.14
	$\beta_7 = 0$	0.28	0.11	0.08	0.30	0.14	0.12	0.31	0.13	0.11
	$\beta_8 = 0$	0.33	0.12	0.09	0.34	0.14	0.11	0.32	0.13	0.10
	$\beta_9 = 0$	0.36	0.12	0.08	0.39	0.16	0.12	0.33	0.15	0.13
	$\beta_{10} = 0$	0.28	0.10	0.08	0.33	0.18	0.15	0.29	0.15	0.12

Using these 80% bounds per replication, the proportion of credible regions containing the true parameter is observed. Denoting this proportion $E_1^{(r)}$, the associated formula is seen in Equation B.8 where the indicator function $\mathbb{1}_{\{L<x<U\}}(x) = 1$ if $L < x < U$ and 0 otherwise.

$$E_1^{(r)} = \frac{1}{2P}\left[\sum_{j=1}^{P} \mathbb{1}_{\{\hat{\alpha}_{j,0.1}^{(r)}<\alpha_j<\hat{\alpha}_{j,0.9}^{(r)}\}}(\alpha_j) + \sum_{k=1}^{P} \mathbb{1}_{\{\hat{\beta}_{k,0.1}^{(r)}<\beta_k<\hat{\beta}_{k,0.9}^{(r)}\}}(\beta_k)\right] \tag{B.8}$$

The number of relevant lag weights is always 4, but the number of irrelevant lag weights depends on P. Still using the 80% credible regions, the proportion of relevant parameters contained in their corresponding intervals ($E_2^{(r)}$) and the proportion of irrelevant parameters (truly zero) contained in their corresponding intervals ($E_3^{(r)}$) is

Table B.3: Parameter RMSE Comparison When $P = 20$ and $T = 50$

	Multicollinearity Control: (ϕ_A, ϕ_B)								
	$(0.9, -0.9)$			$(0.9, -0.5)$			$(0.5, -0.5)$		
Truth	BLASSO	BHS	BHS$^+$	BLASSO	BHS	BHS$^+$	BLASSO	BHS	BHS$^+$
$\alpha_1 = 8$	3.31	2.01	1.86	3.66	2.48	2.29	3.18	2.54	2.41
$\alpha_2 = 0$	1.14	0.77	0.74	1.04	0.85	0.89	1.19	0.86	0.92
$\alpha_3 = -6$	3.33	2.42	2.29	3.51	2.67	2.47	2.96	2.55	2.45
$\alpha_4 = 0$	1.12	0.67	0.65	1.12	0.76	0.73	1.39	0.74	0.68
$\alpha_5 = 0$	1.05	0.44	0.36	0.95	0.42	0.37	1.06	0.44	0.37
$\alpha_6 = 0$	1.21	0.53	0.50	1.11	0.56	0.50	1.23	0.52	0.44
$\alpha_7 = 0$	1.28	0.64	0.57	0.92	0.37	0.31	1.10	0.51	0.48
$\alpha_8 = 0$	0.97	0.34	0.28	0.88	0.42	0.34	1.35	0.56	0.46
$\alpha_9 = 0$	1.04	0.44	0.38	0.91	0.46	0.40	1.16	0.52	0.42
$\alpha_{10} = 0$	1.14	0.41	0.37	0.99	0.50	0.49	1.12	0.55	0.51
$\alpha_{11} = 0$	1.10	0.53	0.50	0.85	0.35	0.29	1.20	0.69	0.67
$\alpha_{12} = 0$	1.27	0.60	0.51	0.87	0.48	0.47	1.17	0.51	0.43
$\alpha_{13} = 0$	1.19	0.54	0.46	1.09	0.44	0.39	1.05	0.42	0.38
$\alpha_{14} = 0$	1.01	0.53	0.50	0.82	0.32	0.26	1.02	0.41	0.35
$\alpha_{15} = 0$	1.12	0.46	0.37	0.81	0.37	0.33	1.04	0.45	0.40
$\alpha_{16} = 0$	1.08	0.50	0.38	1.03	0.34	0.25	1.02	0.49	0.45
$\alpha_{17} = 0$	1.49	0.49	0.39	0.91	0.44	0.38	1.05	0.44	0.37
$\alpha_{18} = 0$	1.09	0.37	0.27	1.06	0.41	0.33	1.29	0.70	0.66
$\alpha_{19} = 0$	1.31	0.50	0.36	0.91	0.44	0.40	1.02	0.57	0.55
$\alpha_{20} = 0$	1.12	0.49	0.44	1.07	0.48	0.40	1.21	0.61	0.59
$\beta_1 = 0$	0.71	0.20	0.15	0.65	0.23	0.21	0.56	0.28	0.25
$\beta_2 = -8$	3.09	0.68	0.61	2.00	0.69	0.65	1.92	0.64	0.61
$\beta_3 = 0$	0.34	0.29	0.28	0.47	0.27	0.24	0.47	0.34	0.32
$\beta_4 = 6$	2.90	0.68	0.60	2.02	0.79	0.73	1.88	0.76	0.73
$\beta_5 = 0$	0.59	0.24	0.18	0.62	0.30	0.29	0.55	0.22	0.19
$\beta_6 = 0$	0.37	0.13	0.12	0.49	0.29	0.28	0.43	0.19	0.16
$\beta_7 = 0$	0.40	0.16	0.14	0.41	0.18	0.14	0.41	0.19	0.16
$\beta_8 = 0$	0.37	0.16	0.12	0.42	0.20	0.16	0.52	0.24	0.22
$\beta_9 = 0$	0.35	0.16	0.11	0.50	0.30	0.28	0.45	0.31	0.31
$\beta_{10} = 0$	0.31	0.14	0.12	0.37	0.19	0.16	0.46	0.25	0.24
$\beta_{11} = 0$	0.33	0.13	0.10	0.45	0.24	0.21	0.41	0.18	0.17
$\beta_{12} = 0$	0.35	0.13	0.10	0.44	0.21	0.17	0.39	0.22	0.19
$\beta_{13} = 0$	0.30	0.13	0.11	0.44	0.21	0.16	0.50	0.30	0.28
$\beta_{14} = 0$	0.37	0.19	0.18	0.40	0.20	0.18	0.39	0.24	0.25
$\beta_{15} = 0$	0.39	0.15	0.12	0.41	0.20	0.18	0.41	0.25	0.24
$\beta_{16} = 0$	0.39	0.18	0.17	0.35	0.17	0.15	0.36	0.16	0.14
$\beta_{17} = 0$	0.30	0.14	0.12	0.43	0.19	0.15	0.36	0.22	0.20
$\beta_{18} = 0$	0.34	0.16	0.13	0.38	0.16	0.12	0.42	0.19	0.17
$\beta_{19} = 0$	0.30	0.13	0.11	0.48	0.23	0.18	0.47	0.25	0.23
$\beta_{20} = 0$	0.31	0.11	0.09	0.45	0.25	0.24	0.42	0.19	0.17

measured. Formulations for $E_2^{(r)}$ and $E_3^{(r)}$ are defined in Equation B.9.

$$
\begin{aligned}
E_2^{(r)} &= \frac{1}{4}\left[\sum_{j=1}^{P} \mathbb{1}_{\{\alpha_j \neq 0 \cap \hat{\alpha}_{j,0.1}^{(r)} < \alpha_j < \hat{\alpha}_{j,0.9}^{(r)}\}}(\alpha_j) + \sum_{k=1}^{P} \mathbb{1}_{\{\beta_k \neq 0 \cap \hat{\beta}_{k,0.1}^{(r)} < \beta_k < \hat{\beta}_{k,0.9}^{(r)}\}}(\beta_k)\right] \\
E_3^{(r)} &= \frac{\sum_{j=1}^{P} \mathbb{1}_{\{\alpha_j = 0 \cap \hat{\alpha}_{j,0.1}^{(r)} < \alpha_j < \hat{\alpha}_{j,0.9}^{(r)}\}}(\alpha_j) + \sum_{k=1}^{P} \mathbb{1}_{\beta_k = 0 \cap \{\hat{\beta}_{k,0.1}^{(r)} < \beta_k < \hat{\beta}_{k,0.9}^{(r)}\}}(\beta_k)}{2P - 4}
\end{aligned} \tag{B.9}
$$

Next, to evaluate the overall difference between the point estimate $\hat{\boldsymbol{\theta}}_{0.5}$ and the true $\boldsymbol{\theta}$, the geometric distance measure ($D^{(r)}$) in Equation B.10 is used. This measure allows the user to assess the inaccuracy in a point estimate across all lag weights in

Table B.4: Parameter RMSE Comparison When $P = 20$ and $T = 200$

	Multicollinearity Control: (ϕ_A, ϕ_B)								
	$(0.9, -0.9)$			$(0.9, -0.5)$			$(0.5, -0.5)$		
Truth	BLASSO	BHS	BHS$^+$	BLASSO	BHS	BHS$^+$	BLASSO	BHS	BHS$^+$
$\alpha_1 = 8$	0.97	0.69	0.68	1.00	0.72	0.70	0.91	0.68	0.66
$\alpha_2 = 0$	0.90	0.34	0.32	0.66	0.26	0.25	0.83	0.38	0.37
$\alpha_3 = -6$	1.27	0.85	0.81	1.47	0.93	0.90	1.33	0.86	0.82
$\alpha_4 = 0$	0.86	0.28	0.24	0.79	0.33	0.32	0.74	0.25	0.23
$\alpha_5 = 0$	0.73	0.22	0.18	0.56	0.17	0.15	0.77	0.27	0.24
$\alpha_6 = 0$	0.73	0.19	0.17	0.51	0.14	0.13	0.66	0.21	0.19
$\alpha_7 = 0$	0.84	0.20	0.16	0.60	0.15	0.13	0.68	0.27	0.23
$\alpha_8 = 0$	0.79	0.21	0.18	0.57	0.14	0.12	0.57	0.19	0.16
$\alpha_9 = 0$	0.73	0.19	0.16	0.59	0.17	0.14	0.73	0.24	0.21
$\alpha_{10} = 0$	0.70	0.15	0.13	0.66	0.18	0.16	0.78	0.27	0.24
$\alpha_{11} = 0$	0.57	0.17	0.14	0.57	0.13	0.11	0.62	0.17	0.14
$\alpha_{12} = 0$	0.64	0.18	0.14	0.49	0.12	0.10	0.64	0.20	0.18
$\alpha_{13} = 0$	0.80	0.20	0.17	0.57	0.16	0.14	0.64	0.24	0.23
$\alpha_{14} = 0$	0.72	0.18	0.15	0.61	0.16	0.14	0.67	0.24	0.21
$\alpha_{15} = 0$	0.78	0.16	0.12	0.60	0.17	0.15	0.69	0.22	0.18
$\alpha_{16} = 0$	0.72	0.15	0.12	0.71	0.21	0.19	0.67	0.21	0.17
$\alpha_{17} = 0$	0.78	0.28	0.27	0.66	0.17	0.15	0.75	0.25	0.22
$\alpha_{18} = 0$	0.83	0.22	0.20	0.60	0.16	0.13	0.62	0.17	0.15
$\alpha_{19} = 0$	0.71	0.19	0.16	0.50	0.13	0.11	0.71	0.26	0.25
$\alpha_{20} = 0$	0.50	0.13	0.10	0.49	0.15	0.13	0.58	0.24	0.21
$\beta_1 = 0$	0.33	0.10	0.09	0.32	0.12	0.11	0.35	0.15	0.14
$\beta_2 = -8$	0.59	0.30	0.29	0.53	0.35	0.34	0.64	0.37	0.36
$\beta_3 = 0$	0.37	0.12	0.12	0.37	0.17	0.17	0.33	0.13	0.12
$\beta_4 = 6$	0.68	0.39	0.38	0.54	0.35	0.35	0.58	0.35	0.34
$\beta_5 = 0$	0.33	0.08	0.08	0.34	0.11	0.10	0.28	0.08	0.07
$\beta_6 = 0$	0.33	0.08	0.08	0.30	0.09	0.08	0.34	0.13	0.13
$\beta_7 = 0$	0.29	0.09	0.08	0.28	0.09	0.08	0.30	0.12	0.12
$\beta_8 = 0$	0.28	0.07	0.07	0.31	0.09	0.08	0.26	0.08	0.07
$\beta_9 = 0$	0.33	0.07	0.06	0.30	0.09	0.08	0.30	0.10	0.09
$\beta_{10} = 0$	0.31	0.06	0.05	0.27	0.09	0.07	0.34	0.11	0.10
$\beta_{11} = 0$	0.25	0.09	0.08	0.32	0.10	0.09	0.28	0.09	0.08
$\beta_{12} = 0$	0.30	0.07	0.06	0.35	0.17	0.16	0.28	0.09	0.07
$\beta_{13} = 0$	0.31	0.10	0.09	0.32	0.10	0.09	0.28	0.07	0.05
$\beta_{14} = 0$	0.33	0.08	0.07	0.33	0.10	0.08	0.29	0.09	0.07
$\beta_{15} = 0$	0.28	0.06	0.05	0.33	0.12	0.11	0.29	0.11	0.10
$\beta_{16} = 0$	0.29	0.08	0.07	0.30	0.09	0.07	0.27	0.09	0.08
$\beta_{17} = 0$	0.29	0.07	0.06	0.30	0.10	0.09	0.31	0.10	0.09
$\beta_{18} = 0$	0.31	0.07	0.05	0.27	0.09	0.07	0.29	0.09	0.08
$\beta_{19} = 0$	0.34	0.06	0.05	0.30	0.09	0.08	0.27	0.09	0.08
$\beta_{20} = 0$	0.29	0.07	0.05	0.25	0.09	0.07	0.24	0.08	0.07

high dimension $\mathbb{R}^{2P}$.

$$D^{(r)} = \sqrt{\sum_{j=1}^{P} (\alpha_j - \hat{\alpha}^{(r)}_{j,0.5})^2 + \sum_{k=1}^{P} (\beta_k - \hat{\beta}^{(r)}_{k,0.5})^2} \tag{B.10}$$

Table B.5 reports means and standard deviations of $E_1^{(r)}$, $E_2^{(r)}$, $E_3^{(r)}$, and $D^{(r)}$ across the 100 replications. Under all circumstances, horseshoe priors not only produce better credible regions but also lead to better posterior point estimates. In the comparison of BHS to BHS$^+$, the horseshoe+ hierarchy results in larger E_1, E_2, and E_3, and lower D across all scenarios.

Table B.5: Overall Measures of Error: Mean (SD) Reported from 100 Replications

			Multicollinearity Control: (ϕ_A, ϕ_B)					
				$(0.9, -0.9)$			$(0.5, -0.5)$	
P	T	Error	BLASSO	BHS	BHS+	BLASSO	BHS	BHS+
5	50	E_1	0.84 (0.14)	0.9 (0.1)	0.92 (0.1)	0.83 (0.14)	0.89 (0.1)	0.92 (0.08)
		E_2	0.83 (0.19)	0.83 (0.2)	0.82 (0.22)	0.8 (0.21)	0.82 (0.18)	0.84 (0.17)
		E_3	0.84 (0.17)	0.95 (0.09)	0.98 (0.06)	0.86 (0.14)	0.94 (0.1)	0.98 (0.06)
		D	3 (1.03)	2.33 (0.9)	2.05 (0.87)	2.83 (1)	2.3 (0.91)	2.05 (0.88)
	200	E_1	0.8 (0.16)	0.89 (0.12)	0.92 (0.1)	0.82 (0.15)	0.89 (0.12)	0.92 (0.1)
		E_2	0.79 (0.21)	0.83 (0.2)	0.83 (0.21)	0.8 (0.21)	0.83 (0.21)	0.84 (0.2)
		E_3	0.81 (0.18)	0.93 (0.11)	0.97 (0.07)	0.84 (0.17)	0.93 (0.11)	0.97 (0.08)
		D	2.02 (0.85)	1.52 (0.72)	1.3 (0.66)	1.96 (0.7)	1.52 (0.63)	1.32 (0.61)
10	50	E_1	0.87 (0.1)	0.95 (0.06)	0.96 (0.05)	0.87 (0.11)	0.94 (0.06)	0.96 (0.05)
		E_2	0.8 (0.22)	0.82 (0.21)	0.82 (0.22)	0.76 (0.23)	0.82 (0.2)	0.82 (0.21)
		E_3	0.89 (0.11)	0.98 (0.05)	0.99 (0.03)	0.9 (0.11)	0.98 (0.05)	0.99 (0.03)
		D	4.36 (1.26)	2.75 (1.08)	2.42 (1.04)	4.17 (1.16)	2.75 (1)	2.44 (0.96)
	200	E_1	0.85 (0.1)	0.96 (0.05)	0.96 (0.04)	0.86 (0.09)	0.95 (0.06)	0.96 (0.05)
		E_2	0.79 (0.21)	0.83 (0.2)	0.83 (0.2)	0.79 (0.21)	0.83 (0.2)	0.82 (0.2)
		E_3	0.86 (0.1)	0.99 (0.03)	0.99 (0.02)	0.88 (0.09)	0.98 (0.04)	0.99 (0.03)
		D	2.82 (0.81)	1.5 (0.59)	1.31 (0.56)	2.63 (0.66)	1.53 (0.59)	1.37 (0.57)
20	50	E_1	0.92 (0.05)	0.97 (0.03)	0.97 (0.03)	0.92 (0.06)	0.97 (0.04)	0.97 (0.03)
		E_2	0.54 (0.32)	0.78 (0.23)	0.78 (0.22)	0.61 (0.3)	0.77 (0.24)	0.78 (0.24)
		E_3	0.96 (0.04)	0.99 (0.02)	1 (0.01)	0.96 (0.05)	0.99 (0.02)	0.99 (0.02)
		D	7.84 (2.35)	3.65 (1.73)	3.31 (1.66)	7.08 (1.87)	4.16 (1.87)	3.87 (1.91)
	200	E_1	0.9 (0.07)	0.98 (0.02)	0.98 (0.02)	0.9 (0.06)	0.98 (0.02)	0.98 (0.02)
		E_2	0.77 (0.2)	0.81 (0.24)	0.8 (0.24)	0.74 (0.23)	0.82 (0.21)	0.82 (0.2)
		E_3	0.91 (0.07)	1 (0.01)	1 (0)	0.91 (0.07)	1 (0.01)	1 (0.01)
		D	3.8 (0.84)	1.42 (0.56)	1.32 (0.54)	3.58 (0.81)	1.56 (0.52)	1.45 (0.5)

APPENDIX C

R CODE FOR CHAPTER 4

```
#########################################################
#Paper:"REGULARIZATION METHODS FOR SUBSET ARMA SELECTION"
#Authors:Mario Giacomazzo (Arizona State University)
#        Yiannis Kamarianakis (Arizona State University)
#Year:2018
#########################################################

################################################################################
#Required R Packages
################################################################################
library(doParallel) #Needed for Potential Parallel Processing
library(foreach) #Needed for Potential Parallel Processing
library(glmnet)  #Needed for Adaptive Lasso/Adaptive Elastic Net Estimation
library(MCMCpack) #Needed for Long AR Regression for Estimating Innovations
library(bayesreg) #Needed for Bayesian Horseshoe/Bayesian Horseshoe+ Estimation
library(forecast) #Needed for Producing Forecasts
library(datasets) #Needed to Load CO2 Data for Mauna Loa, HI, US

options(scipen=999)
################################################################################


####################################################
#Function Required for Obtaining Lagged Time Series
#
#Arguments: x = time series
#           k = lag
####################################################
lag.func<-function(x,k=1){
  t=length(x)
  y=c(rep(NA,t))
  for(i in (k+1):t){
    y[i]=x[i-k]
  }
  return(y)
}
####################################################



##########################################################################
#Data Used In Illustrations (Monthly CO2 Measurements in Mauna Loa, Hawaii)
##########################################################################

#Obtain Dataset and Apply Seasonal and Single Lag Differencing
maunaloa.co2=as.numeric(co2)
maunaloa.co2.time=as.numeric(time(co2))
maunaloa.co2.seasdiff.co2=c(rep(NA,12),diff(maunaloa.co2,12))
maunaloa.co2.final=c(rep(NA,1),diff(maunaloa.co2.seasdiff.co2,1))

#Split Into Modeling and Validation Periods (Mauna Loa)
MODEL.PERIOD=maunaloa.co2.time<1990
VALIDATION.PERIOD=maunaloa.co2.time>=1990

#Plot of Final Stationary Time Series Used in Illustration
plot(x=maunaloa.co2.time,y=maunaloa.co2.final,xlab="",ylab="",type="n",
     main="CO2 After Seasonal and Regular Differencing")
points(x=maunaloa.co2.time[MODEL.PERIOD],
       y=maunaloa.co2.final[MODEL.PERIOD],
       type="l",col="black")
points(x=maunaloa.co2.time[VALIDATION.PERIOD],
       y=maunaloa.co2.final[VALIDATION.PERIOD],
       type="l",col="black",lty=3)

#Remove NA's and Subset Data For Model Fitting Period and
#Model Validation Period
```

```
#Data Fit
maunaloa.co2.train=as.numeric(na.omit(maunaloa.co2.final[MODEL.PERIOD]))
#Forecast
maunaloa.co2.val=as.numeric(na.omit(maunaloa.co2.final[VALIDATION.PERIOD]))
##########################################################################


##########################################################################
#Functions to Evaluate Subset Model Selection of Various Methods
#Requires Knowing True Coefficients of Data Generating Process
#Used in Simulation Experiments For Evaluation of Subset ARMA Selection
#
#Arguments: truecoef = True Known Coefficients
#           estcoef  = Estimated Coefficients
##########################################################################

#Identifies if Non-Zero Parameters Have Been Selected in Final Model (C)
id.sig.coef.func<-function(truecoef,estcoef){
  all(which(truecoef!=0) %in% which(estcoef!=0))
}

#Identifies if the True Model Has Been Selected (I)
id.true.coef.func<-function(truecoef,estcoef){
  if(length(which(truecoef!=0))==length(which(estcoef!=0))){
    all(sort(which(truecoef!=0))==as.numeric(sort(which(estcoef!=0))))
  }else{
    FALSE
  }
}

#Identifies the Proportion of Truly Non-zero Parameters Missed in Final Model (-)
false.neg.func<-function(truecoef,estcoef){
  mean(estcoef[which(truecoef!=0)]==0)
}

#Identifies the Proportion of Truly Zero Parameters Selected in Final Model (+)
false.pos.func<-function(truecoef,estcoef){
  mean(estcoef[which(truecoef==0)]!=0)
}

#Outputs a Vector of Summary Statistics (C,I,-,+)
fulleval.func<-function(truecoef,estcoef){
  return(c(id.sig.coef.func(truecoef,estcoef), #C
           id.true.coef.func(truecoef,estcoef), #I
           false.neg.func(truecoef,estcoef), #-
           false.pos.func(truecoef,estcoef))) #+
}
##########################################################################


##########################################################################
#Function to Perform ADLASSO or ADENET Subse ARMA Estimation With
#Tuning Parameter Selection Based on Minimization of AIC or BIC
#
#Arguments: x = Time Series to Be Modeled Using subset ARMA(maxP,maxQ)
#           h = Horizon Specific Model (Defaults to 1)
#           long.ar.select = Indicator Determining if Model Selection
#                            Should Be Performed in the Initial Modeling
#                            of Long AR Process (Defaults to F)
#           maxP = Maximum AR Order
#           maxQ = Maximum MA Order
#           updateMA = Indicator Determining if Moving Average Terms Should
#                      Be Updated After Initial Coefficients Selected
#                      (Defaults to F)
#           BIC1 = Indicator Determining if BIC Should Be Used in Stage 1
#                  Estimation of Weights (Defaults to T)
```

```
#            BIC2 = Indicator Determining if BIC Should Be Used in Stage 2
#                   Final Model Selection (Defaults to T)
#            alpha = Elastic Net Mixing Parameter
#                    (0=Ridge,1=Lasso,Other=Elastic Net)
#                    (Defaults to Sequence 0,0.1,0.2,...,1)
#            eta = Exponent Applied to Weights (Defaults to 2)
#            Method = Choose Either "ADLASSO" or "ADENET"
#
#Source: Wang and Leng(2007) and Efron et al.(2004) and Zou and Hastie(2005)
###########################################################################

#Creation of Function
adshrink123.func<-function(x,h=1,long.ar.select=F,maxP,maxQ,updateMA=F,
                           BIC1=T,BIC2=T,eta=2,alpha=seq(0,1,0.1),
                           Method=c("ADLASSO","ADENET")){

  #Package Required
  require(glmnet) #Performs Ridge, Lasso, Elastic Net Estimation

  Method=match.arg(Method)

  Nt=length(x) #Length of Input Time Series

  #Fit Long AR Model to Estimate Innovations
  max.ar.order=ceiling(10*log10(Nt)) #Maximum Autoregressive Order
  init.mod.est=ar(x,aic=long.ar.select, #Allows Stepwise Selection
              order.max=max.ar.order,demean=T)
  init.mod.error=residuals(init.mod.est)
  init.mod.order=length(which(is.na(init.mod.error)))

  #Create Model Matrix of AR and MA terms
  dataP=foreach(p=1:maxP,.combine=cbind)%do%{
    lag.func(x,k=(p+h-1))
  }
  dataQ=foreach(q=1:maxQ,.combine=cbind)%do%{
    lag.func(init.mod.error,k=(q+h-1))
  }
  first.modX=as.matrix(cbind(dataP,
              dataQ))[-(1:(init.mod.order+max(maxP,maxQ)+h-1)),]
  first.y=x[-(1:(init.mod.order+max(maxP,maxQ)+h-1))]

  #Number of Alphas For Elastic Net
  n.alpha=length(alpha)

  #Estimation Via ADLASSO
  if(Method=="ADLASSO"){
    #Initial LASSO Weights
    first.mod.est=glmnet(y=first.y,x=first.modX,standardize=T,alpha=1)
    first.mod.RSS=colSums((first.y-predict(first.mod.est,
                                  newx=first.modX))^2)
    if(BIC1){
      first.bic.out=log(length(first.y))*first.mod.est$df+
        length(first.y)*log(as.vector(first.mod.RSS)/length(first.y))
      first.mod.lambda=first.mod.est$lambda[which.min(first.bic.out)]
    }else{
      first.aic.out=2*first.mod.est$df+
        length(first.y)*log(as.vector(first.mod.RSS)/length(first.y))
      first.mod.lambda=first.mod.est$lambda[which.min(first.aic.out)]
    }
    first.mod.coef=as.numeric(coef(first.mod.est,
                  s=first.mod.lambda,method="lambda"))[-1]
    first.mod.mu=as.numeric(coef(first.mod.est,
                             s=first.mod.lambda,method="lambda"))[1]
    weights=abs(first.mod.coef+1/length(first.y))^(-eta)

    #Update Model Matrix of AR and MA terms Based off Initial Estimation
```

```
  if(updateMA){
    update.mod.predict=rep(NA,length(x))
    update.mod.error=rep(0,length(x))
    for(v in (h+max(maxP,maxQ)):Nt){
      update.mod.predict[v]=first.mod.mu+
        x[(v-h):(v-maxP-h+1)]%*%first.mod.coef[1:maxP]+
        update.mod.error[(v-h):(v-maxQ-h+1)]%*%first.mod.coef[-(1:maxP)]
      update.mod.error[v]=x[v]-update.mod.predict[v]
    }
    update.dataQ=foreach(q=1:maxQ,.combine=cbind)%do%{
      lag.func(update.mod.error,k=(q+h-1))
    }
    second.modX=as.matrix(cbind(dataP,
                    update.dataQ))[-(1:(max(maxP,maxQ)+h-1)),]
    second.y=x[-(1:(max(maxP,maxQ)+h-1))]
  }else{
    second.modX=first.modX
    second.y=first.y
  }

  second.mod.est=glmnet(y=second.y,x=second.modX,standardize=T,alpha=1,
                        thresh=1e-16, penalty.factor=weights)
  second.mod.RSS=colSums((second.y-predict(second.mod.est,
                                           newx=second.modX))^2)
  if(BIC2){
    second.bic.out=log(length(second.y))*second.mod.est$df+
      length(second.y)*log(as.vector(second.mod.RSS)/length(second.y))
    second.mod.lambda=second.mod.est$lambda[which.min(second.bic.out)]
  }else{
    second.aic.out=2*second.mod.est$df+
      length(second.y)*log(as.vector(second.mod.RSS)/length(second.y))
    second.mod.lambda=second.mod.est$lambda[which.min(second.aic.out)]
  }

  final.mod.coef=as.numeric(coef(second.mod.est,s=second.mod.lambda,
                  method="lambda"))[-1]
  nonzero.select=which(final.mod.coef!=0)
  final.mod.int=as.numeric(coef(second.mod.est,s=second.mod.lambda,
                 method="lambda"))[1]
  final.mod.s2=sum((second.y-predict(second.mod.est,newx=second.modX,
               s=second.mod.lambda,method="lambda"))^2)/(length(second.y)-
               sum(final.mod.coef[nonzero.select]!=0)-1)

  out=list(final.mod.coef=final.mod.coef, #Final Selection of Coefficients
           final.mod.int=final.mod.int,   #Final Estimated Intercept
           final.mod.s2=final.mod.s2,     #Final Estimated Noise Variance
           nonzero.select=nonzero.select) #Identifies the Nonzero Parameters
}

#Estimation Via ADENET
if(Method=="ADENET"){
  #Initial LASSO Weights
  first.mod.est=glmnet(y=first.y,x=first.modX,standardize=T,alpha=1)
  first.mod.RSS=colSums((first.y-predict(first.mod.est,newx=first.modX))^2)
  if(BIC1){
    first.bic.out=log(length(first.y))*first.mod.est$df+
      length(first.y)*log(as.vector(first.mod.RSS)/length(first.y))
    first.mod.lambda=first.mod.est$lambda[which.min(first.bic.out)]
    first.cv.out=c(1,first.mod.lambda,min(first.bic.out))
  }else{
    first.aic.out=2*first.mod.est$df+
      length(first.y)*log(as.vector(first.mod.RSS)/length(first.y))
    first.mod.lambda=first.mod.est$lambda[which.min(first.aic.out)]
    first.cv.out=c(1,first.mod.lambda,min(first.aic.out))
  }
```

```r
first.mod.alpha=1
first.mod.lambda=first.cv.out[2]
first.mod.est=glmnet(y=first.y,x=first.modX,standardize=T,
              alpha=first.mod.alpha,lambda=first.mod.lambda)
first.mod.coef=as.numeric(coef(first.mod.est))[-1]
first.mod.mu=as.numeric(coef(first.mod.est))[1]

weights=abs(first.mod.coef+1/length(first.y))^(-eta)

#Update Model Matrix of AR and MA terms Based off Initial Estimation
if(updateMA){
  update.mod.predict=rep(NA,length(x))
  update.mod.error=rep(0,length(x))
  for(v in (h+max(maxP,maxQ)):Nt){
    update.mod.predict[v]=first.mod.mu+
      x[(v-h):(v-maxP-h+1)]%*%first.mod.coef[1:maxP]+
      update.mod.error[(v-h):(v-maxQ-h+1)]%*%first.mod.coef[-(1:maxP)]
    update.mod.error[v]=x[v]-update.mod.predict[v]
  }
  update.dataQ=foreach(q=1:maxQ,.combine=cbind)%do%{
    lag.func(update.mod.error,k=(q+h-1))
  }
  second.modX=as.matrix(cbind(dataP,
                              update.dataQ))[-(1:(max(maxP,maxQ)+h-1)),]
  second.y=x[-(1:(max(maxP,maxQ)+h-1))]
}else{
  second.modX=first.modX
  second.y=first.y
}

#Final Elastic Net Estimates (Search Through All Lambdas)
second.cv.out=foreach(a=1:n.alpha,.combine=rbind)%do%{
  second.mod.est=glmnet(y=second.y,x=second.modX,standardize=T,
                  alpha=alpha[a],penalty.factor=weights)
  second.mod.RSS=colSums((second.y-predict(second.mod.est,
                                          newx=second.modX))^2)
  if(BIC2){
    second.bic.out=log(length(second.y))*second.mod.est$df+
      length(second.y)*log(as.vector(second.mod.RSS)/length(second.y))
    second.mod.lambda=second.mod.est$lambda[which.min(second.bic.out)]
    result=c(alpha[a],second.mod.lambda,min(second.bic.out))
  }else{
    second.aic.out=2*second.mod.est$df+
      length(second.y)*log(as.vector(second.mod.RSS)/length(second.y))
    second.mod.lambda=second.mod.est$lambda[which.min(second.aic.out)]
    result=c(alpha[a],second.mod.lambda,min(second.aic.out))
  }
  result
}

second.mod.alpha=alpha[which.min(second.cv.out[,3])]
second.mod.lambda=second.cv.out[which.min(second.cv.out[,3]),2]
second.mod.est=glmnet(y=second.y,x=second.modX,
                      standardize=T,alpha=second.mod.alpha,
                      lambda=second.mod.lambda,penalty.factor=weights)
second.mod.coef=as.numeric(coef(second.mod.est))[-1]
second.mod.mu=as.numeric(coef(second.mod.est))[1]

final.mod.coef=second.mod.coef
nonzero.select=which(final.mod.coef!=0)
final.mod.int=second.mod.mu
final.mod.s2=sum((second.y-predict(second.mod.est,
                 newx=second.modX))^2)/(length(second.y)-
                 sum(final.mod.coef[nonzero.select]!=0)-1)
```

```
    out=list(final.mod.coef=final.mod.coef, #Final Selection of Coefficients
             final.mod.int=final.mod.int,    #Final Estimated Intercept
             final.mod.s2=final.mod.s2,      #Final Estimated Noise Variance
             nonzero.select=nonzero.select) #Identifies the Nonzero Parameters
  }

  return(out)
}

#Illustration of Function for ADLASSO Estimation
adlasso1=adshrink123.func(x=maunaloa.co2.train,h=1,
          long.ar.select=F,maxP=14,maxQ=14,
          updateMA=F,BIC1=F,BIC2=F,eta=2,
          alpha=seq(0,1,0.1),Method="ADLASSO")
adlasso2=adshrink123.func(x=maunaloa.co2.train,h=1,
          long.ar.select=F,maxP=14,maxQ=14,
          updateMA=F,BIC1=F,BIC2=T,eta=2,
          alpha=seq(0,1,0.1),Method="ADLASSO")
adlasso3=adshrink123.func(x=maunaloa.co2.train,h=1,
          long.ar.select=F,maxP=14,maxQ=14,
          updateMA=F,BIC1=T,BIC2=T,eta=2,
          alpha=seq(0,1,0.1),Method="ADLASSO")

#Illustration of Function for ADENET Estimation
adenet1=adshrink123.func(x=maunaloa.co2.train,h=1,
         long.ar.select=F,maxP=14,maxQ=14,
         updateMA=F,BIC1=F,BIC2=F,eta=2,
         alpha=seq(0,1,0.1),Method="ADENET")
adenet2=adshrink123.func(x=maunaloa.co2.train,h=1,
         long.ar.select=F,maxP=14,maxQ=14,
         updateMA=F,BIC1=F,BIC2=T,eta=2,
         alpha=seq(0,1,0.1),Method="ADENET")
adenet3=adshrink123.func(x=maunaloa.co2.train,h=1,
         long.ar.select=F,maxP=14,maxQ=14,
         updateMA=F,BIC1=T,BIC2=T,eta=2,
         alpha=seq(0,1,0.1),Method="ADENET")
##########################################################################


##########################################################################
#Functions Used to Partition Data into a Train Set containing the
#   first (1-test.per)x100% of data and a Test Set containing the
#   last (test.per)x100% of data. The second function
#   institutes a gap of length max.pq to remove temporal dependencies
#Arguments: x = time series
#           max.pq = maximum ar/ma order to consider
#                    (covers temporal dependence)
#           test.per = Percent of Data to Consider in Test Set
#                      (Defaults to 0.2)
#
#Output: Vector of O's (Fit) and 1's (Tuning Parameter Selection)
##########################################################################

#Function Splitting Data According to Classic Out-of-Sample Procedure
OOS.IndepCV.func<-function(x,test.per=0.20){
  N=length(x)
  test.N=ceiling(test.per*N)
  if(test.per>0.5){
    warning("Less than Half the Dataset Is Being Used for Training ")
  }
  Block.Vector=rep(0,N)
  Block.Vector[(N-test.N+1):N]=1
  return(as.matrix(Block.Vector))
}

#Function Splitting OOS But Removing Last max.pq from End of Training Set
OOS.DepCV.func<-function(x,max.pq=NULL,test.per=0.20){
```

```
  N=length(x)
  test.N=ceiling(test.per*N)
  if(is.null(max.pq)) max.pq=ceiling(10*log10(test.N))
  if(test.per>0.5){
    warning("Less than Half the Dataset Is Being Used for Training ")
  }
  if(max.pq>(N-test.N)) warning("max.pq > Number of Obs in Training")
  Block.Vector=rep(0,N)
  Block.Vector[(N-test.N+1):N]=1
  Block.Vector[(N-test.N-max.pq+1):(N-test.N)]=NA
  return(as.matrix(Block.Vector))
}
################################################################################


################################################################################
#Function to Perform ADLASSO or ADENET Subse ARMA Estimation With
#Tuning Parameter Selection Based on Out-Of-Sample Period
#
#Arguments: x = Time Series to Be Modeled Using subset ARMA(maxP,maxQ)
#           h = Horizon Specific Model (Defaults to 1)
#           long.ar.select = Indicator Determining if Model Selection Is
#                            Performed in the Initial Modeling of
#                            Long AR Process (Defaults to F)
#           maxP = Maximum AR Order
#           maxQ = Maximum MA Order
#           test.per = Percentage of Data Removed for
#                      Tuning Parameter Selection (Defaults to 0.2 => 20%)
#           max.pq = Maximimum Temporal Dependence Considered in depOOS
#                    (Defaults to max(maxP,maxQ))
#           updateMA = Indicator Determining if Moving Average Terms Is
#                      Updated After Initial Coefficients Selected
#                      (Defaults to F)
#           BIC1 = Indicator Determining if BIC Should Be Used in Stage 1
#                  Estimation of Weights (Defaults to T)
#           BIC2 = Indicator Determining if BIC Should Be Used in Stage 2
#                  Final Model Selection (Defaults to T)
#           alpha = Elastic Net Mixing Parameter
#                   (0=Ridge,1=Lasso,Other=Elastic Net)
#                   (Defaults to Sequence 0,0.1,0.2,...,1)
#           eta = Exponent Applied to Weights (Defaults to 2)
#           Method = Choose Either "ADLASSO" or "ADENET"
#           CV = Choose Either "OOS" or "depOOS
#           ADENET.final = Appoach for Choosing Final Lambda for Each Alpha
#                          "min" = Choose Based on Minimum MSE
#                          "1se" = Choose Largest Lambda that Results in MSE
#                                  within 1 Standard Error of the Minimum
#
#Source: Wang and Leng(2007) and Efron et al.(2004) and Zou and Hastie(2005)
################################################################################

#Creation of Function
adshrink45.func<-function(x,h=1,long.ar.select=F,maxP,maxQ,updateMA=F,
                          BIC1=T,BIC2=T,eta=2,alpha=seq(0,1,0.1),test.per=0.2,
                          max.pq = max(maxP,maxQ),ADENET.final=c("min","1se"),
                          Method=c("ADLASSO","ADENET"),CV=c("OOS","depOOS")){

  #Package Required
  require(glmnet) #Performs Ridge, Lasso, Elastic Net Estimation

  Method=match.arg(Method)
  CV=match.arg(CV)
  ADENET.final=match.arg(ADENET.final)

  Nt=length(x) #Length of Input Time Series

  #Fit Long AR Model to Estimate Innovations
```

```
max.ar.order=ceiling(10*log10(Nt)) #Maximum Autoregressive Order
init.mod.est=ar(x,aic=long.ar.select, #Allows Stepwise Selection
                order.max=max.ar.order,demean=T)
init.mod.error=residuals(init.mod.est)
init.mod.order=length(which(is.na(init.mod.error)))

#Create Model Matrix of AR and MA terms
dataP=foreach(p=1:maxP,.combine=cbind)%do%{
  lag.func(x,k=(p+h-1))
}
dataQ=foreach(q=1:maxQ,.combine=cbind)%do%{
  lag.func(init.mod.error,k=(q+h-1))
}
first.modX=as.matrix(cbind(dataP,
                dataQ))[-(1:(init.mod.order+max(maxP,maxQ)+h-1)),]
first.y=x[-(1:(init.mod.order+max(maxP,maxQ)+h-1))]

#Number of Alphas For Elastic Net
n.alpha=length(alpha)

#Estimation Via ADLASSO
if(Method=="ADLASSO"){
  if(CV=="OOS"){
    Fold.Vector=OOS.IndepCV.func(x=first.y,test.per=test.per)

    in.train=which(Fold.Vector==0)
    in.test=which(Fold.Vector==1)

    first.mod.est=glmnet(y=first.y[in.train],x=first.modX[in.train,],
                  standardize=T,alpha=1)
    first.mod.res=(first.y[in.test]-predict(first.mod.est,
                  newx=first.modX[in.test,]))^2

    OOS.MSE1=apply(first.mod.res,2,mean)
    lambda1.min=first.mod.est$lambda[which.min(OOS.MSE1)]
    lambda1.1se=first.mod.est$lambda[min(which(OOS.MSE1<(min(OOS.MSE1)+
                sd(OOS.MSE1)/sqrt(length(OOS.MSE1))))))]

    first.mod.coef=as.numeric(coef(first.mod.est,s=lambda1.min))[-1]
    first.mod.mu=as.numeric(coef(first.mod.est,s=lambda1.min))[1]
    weights=abs(first.mod.coef+1/length(first.y))^(-eta)

    if(updateMA){
      update.mod.predict=rep(NA,length(x))
      update.mod.error=rep(0,length(x))
      for(v in (h+max(maxP,maxQ)):Nt){
        update.mod.predict[v]=first.mod.mu+
          x[(v-h):(v-maxP-h+1)]%*%first.mod.coef[1:maxP]+
          update.mod.error[(v-h):(v-maxQ-h+1)]%*%first.mod.coef[-(1:maxP)]
        update.mod.error[v]=x[v]-update.mod.predict[v]
      }
      update.dataQ=foreach(q=1:maxQ,.combine=cbind)%do%{
        lag.func(update.mod.error,k=(q+h-1))
      }
      second.modX=as.matrix(cbind(dataP,
                      update.dataQ))[-(1:(max(maxP,maxQ)+h-1)),]
      second.y=x[-(1:(max(maxP,maxQ)+h-1))]
    }else{
      second.modX=first.modX
      second.y=first.y
    }

    second.mod.est=glmnet(y=second.y[in.train],x=second.modX[in.train,],
                  standardize=T,alpha=1,penalty.factor=weights)
    second.mod.res=(second.y[in.test]-
                  predict(second.mod.est,newx=second.modX[in.test,]))^2
```

```
  OOS.MSE2=apply(second.mod.res,2,mean)
  lambda2.min=second.mod.est$lambda[which.min(OOS.MSE2)]
  lambda2.1se=second.mod.est$lambda[min(which(OOS.MSE2<(min(OOS.MSE2)+
              sd(OOS.MSE2)/sqrt(length(OOS.MSE2)))))]

  second.mod.coef=as.numeric(coef(second.mod.est,s=lambda2.1se))[-1]
  second.mod.mu=as.numeric(coef(second.mod.est,s=lambda2.1se))[1]

  final.mod.coef=second.mod.coef
  nonzero.select=which(final.mod.coef!=0)
  final.mod.int=second.mod.mu
  final.mod.s2=sum((second.y-predict(second.mod.est,newx=second.modX,
               s=lambda2.1se,method="lambda"))^2)/(length(second.y)-
              sum(final.mod.coef[nonzero.select]!=0)-1)

  out=list(final.mod.coef=final.mod.coef, #Final Selected of Coefficients
           final.mod.int=final.mod.int,   #Final Estimated Intercept
           final.mod.s2=final.mod.s2,     #Final Estimated Noise Variance
           nonzero.select=nonzero.select) #Identifies Nonzero Parameters
}

if(CV=="depOOS"){
  Fold.Vector=OOS.DepCV.func(x=first.y,test.per=test.per)

  in.train=which(Fold.Vector==0)
  in.test=which(Fold.Vector==1)

  first.mod.est=glmnet(y=first.y[in.train],
                 x=first.modX[in.train,],standardize=T,alpha=1)
  first.mod.res=(first.y[in.test]-predict(first.mod.est,
                 newx=first.modX[in.test,]))^2

  OOS.MSE1=apply(first.mod.res,2,mean)
  lambda1.min=first.mod.est$lambda[which.min(OOS.MSE1)]
  lambda1.1se=first.mod.est$lambda[min(which(OOS.MSE1<(min(OOS.MSE1)+
              sd(OOS.MSE1)/sqrt(length(OOS.MSE1)))))]

  first.mod.coef=as.numeric(coef(first.mod.est,s=lambda1.min))[-1]
  first.mod.mu=as.numeric(coef(first.mod.est,s=lambda1.min))[1]
  weights=abs(first.mod.coef+1/length(first.y))^(-eta)

  if(updateMA){
    update.mod.predict=rep(NA,length(x))
    update.mod.error=rep(0,length(x))
    for(v in (h+max(maxP,maxQ)):Nt){
      update.mod.predict[v]=first.mod.mu+
        x[(v-h):(v-maxP-h+1)]%*%first.mod.coef[1:maxP]+
        update.mod.error[(v-h):(v-maxQ-h+1)]%*%first.mod.coef[-(1:maxP)]
      update.mod.error[v]=x[v]-update.mod.predict[v]
    }
    update.dataQ=foreach(q=1:maxQ,.combine=cbind)%do%{
      lag.func(update.mod.error,k=(q+h-1))
    }
    second.modX=as.matrix(cbind(dataP,
                        update.dataQ))[-(1:(max(maxP,maxQ)+h-1)),]
    second.y=x[-(1:(max(maxP,maxQ)+h-1))]
  }else{
    second.modX=first.modX
    second.y=first.y
  }

  second.mod.est=glmnet(y=second.y[in.train],x=second.modX[in.train,],
                  standardize=T,alpha=1,penalty.factor=weights)
  second.mod.res=(second.y[in.test]-predict(second.mod.est,
                  newx=second.modX[in.test,]))^2
```

```r
    OOS.MSE2=apply(second.mod.res,2,mean)
    lambda2.min=second.mod.est$lambda[which.min(OOS.MSE2)]
    lambda2.1se=second.mod.est$lambda[min(which(OOS.MSE2<(min(OOS.MSE2)+
                 sd(OOS.MSE2)/sqrt(length(OOS.MSE2))))))]

    second.mod.coef=as.numeric(coef(second.mod.est,s=lambda2.1se))[-1]
    second.mod.mu=as.numeric(coef(second.mod.est,s=lambda2.1se))[1]

    final.mod.coef=second.mod.coef
    nonzero.select=which(final.mod.coef!=0)
    final.mod.int=second.mod.mu
    final.mod.s2=sum((second.y-predict(second.mod.est,newx=second.modX,
                 s=lambda2.1se,method="lambda"))^2)/
                 (length(second.y)-sum(final.mod.coef[nonzero.select]!=0)-1)

    out=list(final.mod.coef=final.mod.coef, #Final Selected of Coefficients
             final.mod.int=final.mod.int,   #Final Estimated Intercept
             final.mod.s2=final.mod.s2,     #Final Estimated Noise Variance
             nonzero.select=nonzero.select) #Identifies Nonzero Parameters
    }
}

#Estimation Via ADENET
if(Method=="ADENET"){
  if(CV=="OOS"){
    Fold.Vector=OOS.IndepCV.func(x=first.y,test.per=test.per)

    in.train=which(Fold.Vector==0)
    in.test=which(Fold.Vector==1)

    first.mod.est=glmnet(y=first.y[in.train],
                         x=first.modX[in.train,],
                         standardize=T,alpha=1)
    first.mod.res=(first.y[in.test]-
                   predict(first.mod.est,newx=first.modX[in.test,]))^2
    OOS.MSE1=apply(first.mod.res,2,mean)
    lambda1.min=first.mod.est$lambda[which.min(OOS.MSE1)]
    lambda1.1se=first.mod.est$lambda[min(which(OOS.MSE1<(min(OOS.MSE1)+
                 sd(OOS.MSE1)/sqrt(length(OOS.MSE1))))))]
    first.cv.out=c(1,lambda1.min,OOS.MSE1[which.min(OOS.MSE1)])

    first.mod.alpha=1
    first.mod.lambda=first.cv.out[2]
    first.mod.est=glmnet(y=first.y,x=first.modX,standardize=T,
                   alpha=first.mod.alpha,lambda=first.mod.lambda)
    first.mod.coef=as.numeric(coef(first.mod.est))[-1]
    first.mod.mu=as.numeric(coef(first.mod.est))[1]

    weights=abs(first.mod.coef+1/length(first.y))^(-eta)

    if(updateMA){
      update.mod.predict=rep(NA,length(x))
      update.mod.error=rep(0,length(x))
      for(v in (h+max(maxP,maxQ)):Nt){
        update.mod.predict[v]=first.mod.mu+
          x[(v-h):(v-maxP-h+1)]%*%first.mod.coef[1:maxP]+
          update.mod.error[(v-h):(v-maxQ-h+1)]%*%first.mod.coef[-(1:maxP)]
        update.mod.error[v]=x[v]-update.mod.predict[v]
      }
      update.dataQ=foreach(q=1:maxQ,.combine=cbind)%do%{
        lag.func(update.mod.error,k=(q+h-1))
      }
      second.modX=as.matrix(cbind(dataP,
                      update.dataQ))[-(1:(max(maxP,maxQ)+h-1)),]
      second.y=x[-(1:(max(maxP,maxQ)+h-1))]
```

```
  }else{
    second.modX=first.modX
    second.y=first.y
  }

  second.cv.out=foreach(a=1:n.alpha,.combine=rbind)%do%{
    second.mod.est=glmnet(y=second.y[in.train],
                          x=second.modX[in.train,],standardize=T,
                          alpha=alpha[a])
    second.mod.res=(second.y[in.test]-
                   predict(second.mod.est,newx=second.modX[in.test,]))^2
    OOS.MSE2=apply(second.mod.res,2,mean)
    lambda2.min=second.mod.est$lambda[which.min(OOS.MSE2)]
    lambda2.1se=second.mod.est$lambda[min(which(OOS.MSE2<(min(OOS.MSE2)+
                sd(OOS.MSE2)/sqrt(length(OOS.MSE2)))))]
    if(ADENET.final=="min") out=c(alpha[a],lambda2.min,min(OOS.MSE2))
    if(ADENET.final=="1se") out=c(alpha[a],lambda2.1se,
            OOS.MSE2[min(which(OOS.MSE2<(min(OOS.MSE2)+
            sd(OOS.MSE2)/sqrt(length(OOS.MSE2)))))])
    out
  }

  second.mod.alpha=alpha[which.min(second.cv.out[,3])]
  second.mod.lambda=second.cv.out[which.min(second.cv.out[,3]),2]
  second.mod.est=glmnet(y=second.y,x=second.modX,standardize=T,
                 alpha=second.mod.alpha,lambda=second.mod.lambda,
                 penalty.factor=weights)
  second.mod.coef=as.numeric(coef(second.mod.est))[-1]
  second.mod.mu=as.numeric(coef(second.mod.est))[1]

  final.mod.coef=second.mod.coef
  nonzero.select=which(final.mod.coef!=0)
  final.mod.int=second.mod.mu
  final.mod.s2=sum((second.y-
               predict(second.mod.est,
               newx=second.modX))^2)/(length(second.y)-
               sum(final.mod.coef[nonzero.select]!=0)-1)

  out=list(final.mod.coef=final.mod.coef, #Final Selected of Coefficients
           final.mod.int=final.mod.int,   #Final Estimated Intercept
           final.mod.s2=final.mod.s2,     #Final Estimated Noise Variance
           nonzero.select=nonzero.select) #Identifies Nonzero Parameters
}

if(CV=="depOOS"){
  Fold.Vector=OOS.DepCV.func(x=first.y,test.per=test.per)

  in.train=which(Fold.Vector==0)
  in.test=which(Fold.Vector==1)

  first.mod.est=glmnet(y=first.y[in.train],
                       x=first.modX[in.train,],
                       standardize=T,alpha=1)
  first.mod.res=(first.y[in.test]-
                 predict(first.mod.est,newx=first.modX[in.test,]))^2
  OOS.MSE1=apply(first.mod.res,2,mean)
  lambda1.min=first.mod.est$lambda[which.min(OOS.MSE1)]
  lambda1.1se=first.mod.est$lambda[min(which(OOS.MSE1<(min(OOS.MSE1)+
              sd(OOS.MSE1)/sqrt(length(OOS.MSE1)))))]
  first.cv.out=c(1,lambda1.min,OOS.MSE1[which.min(OOS.MSE1)])

  first.mod.alpha=1
  first.mod.lambda=first.cv.out[2]
  first.mod.est=glmnet(y=first.y,x=first.modX,standardize=T,
                 alpha=first.mod.alpha,lambda=first.mod.lambda)
  first.mod.coef=as.numeric(coef(first.mod.est))[-1]
```

```
    first.mod.mu=as.numeric(coef(first.mod.est))[1]

    weights=abs(first.mod.coef+1/length(first.y))^(-eta)

    if(updateMA){
      update.mod.predict=rep(NA,length(x))
      update.mod.error=rep(0,length(x))
      for(v in (h+max(maxP,maxQ)):Nt){
        update.mod.predict[v]=first.mod.mu+
          x[(v-h):(v-maxP-h+1)]%*%first.mod.coef[1:maxP]+
          update.mod.error[(v-h):(v-maxQ-h+1)]%*%first.mod.coef[-(1:maxP)]
        update.mod.error[v]=x[v]-update.mod.predict[v]
      }
      update.dataQ=foreach(q=1:maxQ,.combine=cbind)%do%{
        lag.func(update.mod.error,k=(q+h-1))
      }
      second.modX=as.matrix(cbind(dataP,
                      update.dataQ))[-(1:(max(maxP,maxQ)+h-1)),]
      second.y=x[-(1:(max(maxP,maxQ)+h-1))]
    }else{
      second.modX=first.modX
      second.y=first.y
    }

    second.cv.out=foreach(a=1:n.alpha,.combine=rbind)%do%{
      second.mod.est=glmnet(y=second.y[in.train],
                            x=second.modX[in.train,],standardize=T,
                            alpha=alpha[a])
      second.mod.res=(second.y[in.test]-
                      predict(second.mod.est,
                      newx=second.modX[in.test,]))^2
      OOS.MSE2=apply(second.mod.res,2,mean)
      lambda2.min=second.mod.est$lambda[which.min(OOS.MSE2)]
      lambda2.1se=second.mod.est$lambda[min(which(OOS.MSE2<(min(OOS.MSE2)+
                  sd(OOS.MSE2)/sqrt(length(OOS.MSE2))))))]
      if(ADENET.final=="min") out=c(alpha[a],lambda2.min,min(OOS.MSE2))
      if(ADENET.final=="1se") out=c(alpha[a],lambda2.1se,
                  OOS.MSE2[min(which(OOS.MSE2<(min(OOS.MSE2)+
                  sd(OOS.MSE2)/sqrt(length(OOS.MSE2)))))])
      out
    }

    second.mod.alpha=alpha[which.min(second.cv.out[,3])]
    second.mod.lambda=second.cv.out[which.min(second.cv.out[,3]),2]
    second.mod.est=glmnet(y=second.y,x=second.modX,standardize=T,
                   alpha=second.mod.alpha,lambda=second.mod.lambda,
                   penalty.factor=weights)
    second.mod.coef=as.numeric(coef(second.mod.est))[-1]
    second.mod.mu=as.numeric(coef(second.mod.est))[1]

    final.mod.coef=second.mod.coef
    nonzero.select=which(final.mod.coef!=0)
    final.mod.int=second.mod.mu
    final.mod.s2=sum((second.y-
                 predict(second.mod.est,
                 newx=second.modX))^2)/(length(second.y)-
                 sum(final.mod.coef[nonzero.select]!=0)-1)

    out=list(final.mod.coef=final.mod.coef, #Final Selected of Coefficients
             final.mod.int=final.mod.int,   #Final Estimated Intercept
             final.mod.s2=final.mod.s2,     #Final Estimated Noise Variance
             nonzero.select=nonzero.select) #Identifies Nonzero Parameters
  }
}

return(out)
```

```
}

#Illustration of Function for ADLASSO Estimation
adlasso4=adshrink45.func(x=maunaloa.co2.train,h=1,
          long.ar.select=F,maxP=14,maxQ=14,
          updateMA=F,BIC1=F,BIC2=F,eta=2,alpha=seq(0,1,0.1),Method="ADLASSO",
          test.per=0.2,CV="OOS")
adlasso5=adshrink45.func(x=maunaloa.co2.train,h=1,
          long.ar.select=F,maxP=14,maxQ=14,
          updateMA=F,BIC1=F,BIC2=T,eta=2,alpha=seq(0,1,0.1),Method="ADLASSO",
          test.per=0.2,max.pq=max(maxP,maxQ),CV="depOOS")

#Illustration of Function for ADENET Estimation
adenet4=adshrink45.func(x=maunaloa.co2.train,h=1,
         long.ar.select=F,maxP=14,maxQ=14,
         updateMA=F,BIC1=F,BIC2=F,eta=2,alpha=seq(0,1,0.1),Method="ADENET",
         ADENET.final="min",test.per=0.2,CV="OOS")
adenet5=adshrink45.func(x=maunaloa.co2.train,h=1,
         long.ar.select=F,maxP=14,maxQ=14,
         updateMA=F,BIC1=F,BIC2=T,eta=2,alpha=seq(0,1,0.1),Method="ADENET",
         ADENET.final="min",test.per=0.2,max.pq=max(maxP,maxQ),CV="depOOS")
################################################################################


################################################################################
#Function to Perform ADLASSO or ADENET Subse ARMA Estimation With
#Tuning Parameter Selection Based on Regular K-fold Cross Validation
#
#Arguments: x = Time Series to Be Modeled Using subset ARMA(maxP,maxQ)
#            h = Horizon Specific Model (Defaults to 1)
#            long.ar.select = Indicator Determining if Model Selection Is
#                             Performed in the Initial Modeling of
#                             Long AR Process (Defaults to F)
#            maxP = Maximum AR Order
#            maxQ = Maximum MA Order
#            updateMA = Indicator Determining if Moving Average Terms Is
#                       Updated After Initial Coefficients Selected
#                       (Defaults to F)
#            BIC1 = Indicator Determining if BIC Should Be Used in Stage 1
#                   Estimation of Weights (Defaults to T)
#            BIC2 = Indicator Determining if BIC Should Be Used in Stage 2
#                   Final Model Selection (Defaults to T)
#            alpha = Elastic Net Mixing Parameter
#                    (0=Ridge,1=Lasso,Other=Elastic Net)
#                    (Defaults to Sequence 0,0.1,0.2,...,1)
#            eta = Exponent Applied to Weights (Defaults to 2)
#            Method = Choose Either "ADLASSO" or "ADENET"
#            K = Number of Folds for General CV (Defaults to NULL => LOOCV)
#            ADENET.final = Appoach for Choosing Final Lambda for Each Alpha
#                           "min" = Choose Based on Minimum MSE
#                           "1se" = Choose Largest Lambda that Results in MSE
#                                   within 1 Standard Error of the Minimum
#
#Source: Wang and Leng(2007) and Efron et al.(2004) and Zou and Hastie(2005)
################################################################################

#Creation of Function
adshrink678.func<-function(x,h=1,long.ar.select=F,maxP,maxQ,updateMA=F,
                           BIC1=T,BIC2=T,eta=2,alpha=seq(0,1,0.1),
                           ADENET.final=c("min","1se"),
                           K=NULL,Method=c("ADLASSO","ADENET")){

  #Package Required
  require(glmnet) #Performs Ridge, Lasso, Elastic Net Estimation

  Method=match.arg(Method)
  ADENET.final=match.arg(ADENET.final)
```

```r
Nt=length(x) #Length of Input Time Series

#Fit Long AR Model to Estimate Innovations
max.ar.order=ceiling(10*log10(Nt)) #Maximum Autoregressive Order
init.mod.est=ar(x,aic=long.ar.select, #Allows Stepwise Selection
                order.max=max.ar.order,demean=T)
init.mod.error=residuals(init.mod.est)
init.mod.order=length(which(is.na(init.mod.error)))

#Create Model Matrix of AR and MA terms
dataP=foreach(p=1:maxP,.combine=cbind)%do%{
  lag.func(x,k=(p+h-1))
}
dataQ=foreach(q=1:maxQ,.combine=cbind)%do%{
  lag.func(init.mod.error,k=(q+h-1))
}
first.modX=as.matrix(cbind(dataP,
           dataQ))[-(1:(init.mod.order+max(maxP,maxQ)+h-1)),]
first.y=x[-(1:(init.mod.order+max(maxP,maxQ)+h-1))]

#Number of Alphas For Elastic Net
n.alpha=length(alpha)

#Estimation Via ADLASSO
if(Method=="ADLASSO"){
  if(is.null(K)){
    first.mod.est=cv.glmnet(y=first.y,x=first.modX,standardize=T,
                  alpha=1,parallel=F)
  }else{
    first.mod.est=cv.glmnet(y=first.y,x=first.modX,standardize=T,
                  alpha=1,nfolds=K,parallel=F)
  }

  first.mod.coef=coef(first.mod.est,s=first.mod.est$lambda.min)[-1]
  first.mod.mu=coef(first.mod.est,s=first.mod.est$lambda.min)[1]
  weights=abs(first.mod.coef+1/length(first.y))^(-2)

  if(updateMA){
    update.mod.predict=rep(NA,length(x))
    update.mod.error=rep(0,length(x))
    for(v in (h+max(maxP,maxQ)):Nt){
      update.mod.predict[v]=first.mod.mu+
        x[(v-h):(v-maxP-h+1)]%*%first.mod.coef[1:maxP]+
        update.mod.error[(v-h):(v-maxQ-h+1)]%*%first.mod.coef[-(1:maxP)]
      update.mod.error[v]=x[v]-update.mod.predict[v]
    }
    update.dataQ=foreach(q=1:maxQ,.combine=cbind)%do%{
      lag.func(update.mod.error,k=(q+h-1))
    }
    second.modX=as.matrix(cbind(dataP,
                update.dataQ))[-(1:(max(maxP,maxQ)+h-1)),]
    second.y=x[-(1:(max(maxP,maxQ)+h-1))]
  }else{
    second.modX=first.modX
    second.y=first.y
  }

  if(is.null(K)){
    second.mod.est=cv.glmnet(y=second.y,x=second.modX,standardize=T,
                   parallel=F,alpha=1,penalty.factor=weights)
  }else{
    second.mod.est=cv.glmnet(y=second.y,x=second.modX,standardize=T,
                   parallel=F,alpha=1,nfolds=K,penalty.factor=weights)
  }
```

```
  final.mod.coef=coef(second.mod.est,s=second.mod.est$lambda.1se)[-1]
  nonzero.select=which(final.mod.coef!=0)
  final.mod.int=coef(second.mod.est,s=second.mod.est$lambda.1se)[1]
  final.mod.s2=sum((second.y-predict(second.mod.est,newx=second.modX,
               s=second.mod.est$lambda.1se,method="lambda"))^2)/
               (length(second.y)-sum(final.mod.coef[nonzero.select]!=0)-1)

  out=list(final.mod.coef=final.mod.coef, #Final Selection of Coefficients
           final.mod.int=final.mod.int,   #Final Estimated Intercept
           final.mod.s2=final.mod.s2,     #Final Estimated Noise Variance
           nonzero.select=nonzero.select) #Identifies the Nonzero Parameters
}

#Estimation Via ADENET
if(Method=="ADENET"){
  if(is.null(K)){
    first.mod.est=cv.glmnet(parallel=F,y=first.y,x=first.modX,
                  standardize=T,alpha=1)
    first.cv.out=c(1,first.mod.est$lambda.min,
      first.mod.est$cvm[which(first.mod.est$lambda==first.mod.est$lambda.min)])
  }else{
    first.mod.est=cv.glmnet(parallel=F,y=first.y,x=first.modX,
                  standardize=T,alpha=1,nfolds=K)
    first.cv.out=c(1,first.mod.est$lambda.min,
      first.mod.est$cvm[which(first.mod.est$lambda==first.mod.est$lambda.min)])
  }

  first.mod.alpha=1
  first.mod.lambda=first.cv.out[2]
  first.mod.est=glmnet(y=first.y,x=first.modX,standardize=T,
                alpha=first.mod.alpha,lambda=first.mod.lambda)
  first.mod.coef=as.numeric(coef(first.mod.est))[-1]
  first.mod.mu=as.numeric(coef(first.mod.est))[1]

  weights=abs(first.mod.coef+1/length(first.y))^(-eta)


  if(updateMA){
    update.mod.predict=rep(NA,length(x))
    update.mod.error=rep(0,length(x))
    for(v in (h+max(maxP,maxQ)):Nt){
      update.mod.predict[v]=first.mod.mu+
        x[(v-h):(v-maxP-h+1)]%*%first.mod.coef[1:maxP]+
        update.mod.error[(v-h):(v-maxQ-h+1)]%*%first.mod.coef[-(1:maxP)]
      update.mod.error[v]=x[v]-update.mod.predict[v]
    }
    update.dataQ=foreach(q=1:maxQ,.combine=cbind)%do%{
      lag.func(update.mod.error,k=(q+h-1))
    }
    second.modX=as.matrix(cbind(dataP,
                update.dataQ))[-(1:(max(maxP,maxQ)+h-1)),]
    second.y=x[-(1:(max(maxP,maxQ)+h-1))]
  }else{
    second.modX=first.modX
    second.y=first.y
  }

  n.alpha=length(alpha)

  if(is.null(K)){
    second.cv.out=NULL
    for(a in 1:n.alpha){
      second.mod.est=cv.glmnet(parallel=F,y=second.y,x=second.modX,
                     standardize=T,alpha=alpha[a],penalty.factor=weights)
      if(ADENET.final=="min") second.cv.out=rbind(second.cv.out,
        c(alpha[a],second.mod.est$lambda.min,
```

```
    second.mod.est$cvm[which(second.mod.est$lambda==second.mod.est$lambda.min)]))
        if(ADENET.final=="1se") second.cv.out=rbind(second.cv.out,
          c(alpha[a],second.mod.est$lambda.1se,
    second.mod.est$cvm[which(second.mod.est$lambda==second.mod.est$lambda.1se)]))
      }
    }else{
      second.cv.out=NULL
      for(a in 1:n.alpha){
        second.mod.est=cv.glmnet(parallel=F,y=second.y,x=second.modX,
                        standardize=T,alpha=alpha[a],nfolds=K,
                        penalty.factor=weights)
        if(ADENET.final=="min") second.cv.out=rbind(second.cv.out,c(alpha[a],
            second.mod.est$lambda.min,
    second.mod.est$cvm[which(second.mod.est$lambda==second.mod.est$lambda.min)]))
        if(ADENET.final=="1se") second.cv.out=rbind(second.cv.out,
            c(alpha[a],second.mod.est$lambda.1se,
    second.mod.est$cvm[which(second.mod.est$lambda==second.mod.est$lambda.1se)]))
      }
    }

    second.mod.alpha=alpha[which.min(second.cv.out[,3])]
    second.mod.lambda=second.cv.out[which.min(second.cv.out[,3]),2]
    second.mod.est=glmnet(y=second.y,x=second.modX,standardize=T,
              alpha=second.mod.alpha,lambda=second.mod.lambda,
              penalty.factor=weights)
    second.mod.coef=as.numeric(coef(second.mod.est))[-1]
    second.mod.mu=as.numeric(coef(second.mod.est))[1]

    final.mod.coef=second.mod.coef
    nonzero.select=which(final.mod.coef!=0)
    final.mod.int=second.mod.mu
    final.mod.s2=sum((second.y-predict(second.mod.est,newx=second.modX))^2)/
                (length(second.y)-sum(final.mod.coef[nonzero.select]!=0)-1)

    out=list(final.mod.coef=final.mod.coef, #Final Selection of Coefficients
            final.mod.int=final.mod.int,    #Final Estimated Intercept
            final.mod.s2=final.mod.s2,      #Final Estimated Noise Variance
            nonzero.select=nonzero.select) #Identifies the Nonzero Parameters
  }

  return(out)
}

#Illustration of Function for ADLASSO Estimation
adlasso6=adshrink678.func(x=maunaloa.co2.train,h=1,
          long.ar.select=F,maxP=14,maxQ=14,
          updateMA=F,BIC1=F,BIC2=F,eta=2,alpha=seq(0,1,0.1),
          Method="ADLASSO",K=5)
adlasso7=adshrink678.func(x=maunaloa.co2.train,h=1,
          long.ar.select=F,maxP=14,maxQ=14,
          updateMA=F,BIC1=F,BIC2=T,eta=2,alpha=seq(0,1,0.1),
          Method="ADLASSO",K=10)
adlasso8=adshrink678.func(x=maunaloa.co2.train,h=1,
          long.ar.select=F,maxP=14,maxQ=14,
          updateMA=F,BIC1=F,BIC2=T,eta=2,alpha=seq(0,1,0.1),
          Method="ADLASSO")

#Illustration of Function for ADENET Estimation
adenet6=adshrink678.func(x=maunaloa.co2.train,h=1,
         long.ar.select=F,maxP=14,maxQ=14,
         updateMA=F,BIC1=F,BIC2=F,eta=2,alpha=seq(0,1,0.1),
         ADENET.final="min",Method="ADENET",K=5)
adenet7=adshrink678.func(x=maunaloa.co2.train,h=1,
                         long.ar.select=F,maxP=14,maxQ=14,
        updateMA=F,BIC1=F,BIC2=T,eta=2,alpha=seq(0,1,0.1),
        ADENET.final="min",Method="ADENET",K=10)
```

```
adenet8=adshrink678.func(x=maunaloa.co2.train,h=1,
         long.ar.select=F,maxP=14,maxQ=14,
         updateMA=F,BIC1=F,BIC2=T,eta=2,alpha=seq(0,1,0.1),
         ADENET.final="min",Method="ADENET")
################################################################################


################################################################################
#Functions Used  to Perform K-fold Non-Dependent Cross Validation.
#   All Three methods are based on dividing time series into blocks and
#   performing CV with the blocks removing data that has mutual dependence
#   with test and training sets
#Arguments: x = time series
#             max.pq = maximum ar/ma order to consider
#             (covers temporal dependence)
#             K = Number of Folds to Consider
#Source: Burman(2000), Racine(2000), and Bergmeir(2018)
################################################################################


#Method 1: My Method 1 (Partition Data Into Blocks of Length max.pq)
NonDepCV1.func<-function(x,max.pq=NULL,K=NULL){
  N=length(x)
  if(is.null(max.pq)) max.pq=ceiling(10*log10(N))
  if(max.pq>ceiling(10*log10(N))){
    warning("Maximum ARMA Order Considered
            Too Large Based on Time Series Length")
  }
  nblocks=floor(N/max.pq)
  if(is.null(K)) K=nblocks
  blocks=rep(nblocks,N)
  for(b in 1:(nblocks-1)){
    blocks[(b*max.pq-max.pq+1):(b*max.pq)]=b
  }
  if(K>nblocks){
    warning("Maximum Number of Folds Surpassed Based on Choice of max.pq
            and Time Series Length")
  }
  test.blocks=sort(sample(1:nblocks,K))
  Block.Matrix=matrix(0,N,K)
  for(v in 1:K){
    in.test=which(blocks==test.blocks[v])
    Block.Matrix[in.test,v]=1
    if(test.blocks[v]==1){
      out.ignore=which(blocks==2)
      Block.Matrix[out.ignore,v]=NA
    }else if(test.blocks[v]==nblocks){
      out.ignore=which(blocks==(nblocks-1))
      Block.Matrix[out.ignore,v]=NA
    }else{
      out.ignore=which(blocks %in% c(test.blocks[v]-1,test.blocks[v]+1))
      Block.Matrix[out.ignore,v]=NA
    }
  }
  return(Block.Matrix)
}

#Method 2: Begmeir Method (Divide Data Into K Blocks and
#            Remove Boundary Dependent Data)
NonDepCV2.func<-function(x,max.pq=NULL,K=NULL){
  N=length(x)
  obs.per.block=floor(N/K)
  if(is.null(max.pq)) max.pq=ceiling(10*log10(obs.per.block))
  if((max.pq/2)>obs.per.block){
    warning("Choice of max.pq and K Lead to No Observations")
```

```
  }
  blocks=rep(NA,N)
  for(b in 1:K){
    blocks[(b*obs.per.block-obs.per.block+
              ceiling(max.pq/2)+1):(b*obs.per.block-ceiling(max.pq)/2)]=b
  }

  Block.Matrix=matrix(NA,N,K)
  for(v in 1:K){
    in.test=which(blocks==(1:K)[v])
    in.train=which(blocks %in% (1:K)[-v])
    Block.Matrix[in.test,v]=1
    Block.Matrix[in.train,v]=0
  }

  return(Block.Matrix)
}
##############################################################################


##############################################################################
#Function to Perform ADLASSO or ADENET Subse ARMA Estimation With
#Tuning Parameter Selection Based on Dependent K-fold Cross Validation
#
#Arguments: x = Time Series to Be Modeled Using subset ARMA(maxP,maxQ)
#           h = Horizon Specific Model (Defaults to 1)
#           long.ar.select = Indicator Determining if Model Selection Is
#                            Performed in the Initial Modeling of
#                            Long AR Process(Defaults to F)
#           maxP = Maximum AR Order
#           maxQ = Maximum MA Order
#           updateMA = Indicator Determining if Moving Average Terms Is
#                      Updated After Initial Coefficients Selected
#                      (Defaults to F)
#           alpha = Elastic Net Mixing Parameter
#                   (0=Ridge,1=Lasso,Other=Elastic Net)
#                   (Defaults to Sequence 0,0.1,0.2,...,1)
#           eta = Exponent Applied to Weights (Defaults to 2)
#           Method = Choose Either "ADLASSO" or "ADENET"
#           K = Number of Folds for General CV (Defaults to NULL => LOOCV)
#           max.pq = Maximimum Temporal Dependence Considered in depOOS
#                    (Defaults to max(maxP,maxQ))
#           CV = Choose Either "KFOLD" or "LOBOCV"
#           ADENET.final = Appoach for Choosing Final Lambda for Each Alpha
#                          "min" = Choose Based on Minimum MSE
#                          "1se" = Choose Largest Lambda that Results in MSE
#                                  within 1 Standard Error of the Minimum
#
#Source: Wang and Leng(2007) and Efron et al.(2004) and Zou and Hastie(2005)
##############################################################################

#Creation of Function
adshrink91011.func<-function(x,h=1,long.ar.select=F,maxP,maxQ,updateMA=F,
                            eta=2,alpha=seq(0,1,0.1),
                            ADENET.final=c("min","1se"),max.pq = max(maxP,maxQ),
                            K=NULL,Method=c("ADLASSO","ADENET"),
                            CV=c("KFOLD","LOBOCV")){

  #Package Required
  require(glmnet) #Performs Ridge, Lasso, Elastic Net Estimation

  Method=match.arg(Method)
  CV=match.arg(CV)
  ADENET.final=match.arg(ADENET.final)

  Nt=length(x) #Length of Input Time Series
```

```r
#Fit Long AR Model to Estimate Innovations
max.ar.order=ceiling(10*log10(Nt)) #Maximum Autoregressive Order
init.mod.est=ar(x,aic=long.ar.select, #Allows Stepwise Selection
                order.max=max.ar.order,demean=T)
init.mod.error=residuals(init.mod.est)
init.mod.order=length(which(is.na(init.mod.error)))

#Create Model Matrix of AR and MA terms
dataP=foreach(p=1:maxP,.combine=cbind)%do%{
  lag.func(x,k=(p+h-1))
}
dataQ=foreach(q=1:maxQ,.combine=cbind)%do%{
  lag.func(init.mod.error,k=(q+h-1))
}
first.modX=as.matrix(cbind(dataP,
          dataQ))[-(1:(init.mod.order+max(maxP,maxQ)+h-1)),]
first.y=x[-(1:(init.mod.order+max(maxP,maxQ)+h-1))]

#Number of Alphas For Elastic Net
n.alpha=length(alpha)

#Estimation Via ADLASSO
if(Method=="ADLASSO"){
  if(CV=="KFOLD"){
    Fold.Matrix=NonDepCV2.func(x=first.y,max.pq=max.pq,K=K)
    nfolds=dim(Fold.Matrix)[2]

    lambda1.seq=cv.glmnet(y=first.y,x=first.modX,
                        standardize=T,alpha=1)$lambda

    SQDEV1=foreach(f=1:nfolds,.combine=rbind)%do%{
      in.train=which(Fold.Matrix[,f]==0)
      in.test=which(Fold.Matrix[,f]==1)
      first.mod.est=glmnet(y=first.y[in.train],x=first.modX[in.train,],
                   standardize=T,alpha=1,lambda=lambda1.seq)
      first.mod.res=(first.y[in.test]-predict(first.mod.est,
                    newx=first.modX[in.test,]))^2
      first.mod.res
    }

    CVM1=apply(SQDEV1,2,mean)
    lambda1.min=lambda1.seq[which.min(CVM1)]
    lambda1.1se=lambda1.seq[min(which(CVM1<(min(CVM1)+
              sd(CVM1)/sqrt(length(CVM1)))))]

    first.mod.est=glmnet(y=first.y,x=first.modX,standardize=T,
                 alpha=1,lambda=lambda1.min)
    first.mod.coef=as.numeric(coef(first.mod.est))[-1]
    first.mod.mu=as.numeric(coef(first.mod.est))[1]
    weights=abs(first.mod.coef+1/length(first.y))^(-eta)

    if(updateMA){
      update.mod.predict=rep(NA,length(x))
      update.mod.error=rep(0,length(x))
      for(v in (h+max(maxP,maxQ)):Nt){
        update.mod.predict[v]=first.mod.mu+
          x[(v-h):(v-maxP-h+1)]%*%first.mod.coef[1:maxP]+
          update.mod.error[(v-h):(v-maxQ-h+1)]%*%first.mod.coef[-(1:maxP)]
        update.mod.error[v]=x[v]-update.mod.predict[v]
      }
      update.dataQ=foreach(q=1:maxQ,.combine=cbind)%do%{
        lag.func(update.mod.error,k=(q+h-1))
      }
      second.modX=as.matrix(cbind(dataP,
                 update.dataQ))[-(1:(max(maxP,maxQ)+h-1)),]
```

```
    second.y=x[-(1:(max(maxP,maxQ)+h-1))]
  }else{
    second.modX=first.modX
    second.y=first.y
  }

  lambda2.seq=cv.glmnet(y=second.y,x=second.modX,standardize=T,
              parallel=F, alpha=1,penalty.factor=weights)$lambda

  SQDEV2=foreach(f=1:nfolds,.combine=rbind)%do%{
    in.train=which(Fold.Matrix[,f]==0)
    in.test=which(Fold.Matrix[,f]==1)
    first.mod.est=glmnet(y=second.y[in.train],x=second.modX[in.train,],
                 standardize=T,alpha=1,lambda=lambda2.seq,
                 penalty.factor=weights)
    first.mod.res=(second.y[in.test]-predict(first.mod.est,
                 newx=second.modX[in.test,]))^2
    first.mod.res
  }

  CVM2=apply(SQDEV2,2,mean)
  lambda2.min=lambda2.seq[which.min(CVM2)]
  lambda2.1se=lambda2.seq[min(which(CVM2<(min(CVM2)+
             sd(CVM2)/sqrt(length(CVM2)))))]

  final.mod.est=glmnet(y=second.y,x=second.modX,standardize=T,
                alpha=1,lambda=lambda2.1se,penalty.factor=weights)
  final.mod.coef=as.numeric(coef(final.mod.est))[-1]
  nonzero.select=which(final.mod.coef!=0)
  final.mod.int=as.numeric(coef(final.mod.est))[1]
  final.mod.s2=sum((second.y-predict(final.mod.est,
               newx=second.modX,s=lambda2.1se,
               method="lambda"))^2)/(length(second.y)-
              sum(final.mod.coef[nonzero.select]!=0)-1)

  out=list(final.mod.coef=final.mod.coef, #Final Selection of Coefficients
           final.mod.int=final.mod.int,   #Final Estimated Intercept
           final.mod.s2=final.mod.s2,     #Final Estimated Noise Variance
           nonzero.select=nonzero.select) #Identifies Nonzero Parameters
}else{
  Fold.Matrix=NonDepCV1.func(first.y,max.pq=max.pq,K=K)
  nfolds=dim(Fold.Matrix)[2]

  lambda1.seq=cv.glmnet(y=first.y,x=first.modX,
                        standardize=T,alpha=1)$lambda

  SQDEV1=foreach(f=1:nfolds,.combine=rbind)%do%{
    in.train=which(Fold.Matrix[,f]==0)
    in.test=which(Fold.Matrix[,f]==1)
    first.mod.est=glmnet(y=first.y[in.train],x=first.modX[in.train,],
                 standardize=T,alpha=1,lambda=lambda1.seq)
    first.mod.res=(first.y[in.test]-predict(first.mod.est,
                  newx=first.modX[in.test,]))^2
    first.mod.res
  }

  CVM1=apply(SQDEV1,2,mean)
  lambda1.min=lambda1.seq[which.min(CVM1)]
  lambda1.1se=lambda1.seq[min(which(CVM1<(min(CVM1)+
             sd(CVM1)/sqrt(length(CVM1)))))]

  first.mod.est=glmnet(y=first.y,x=first.modX,standardize=T,
                       alpha=1,lambda=lambda1.min)
  first.mod.coef=as.numeric(coef(first.mod.est))[-1]
  first.mod.mu=as.numeric(coef(first.mod.est))[1]
  weights=abs(first.mod.coef+1/length(first.y))^(-eta)
```

```
update.mod.predict=rep(NA,length(x))
update.mod.error=rep(0,length(x))
for(v in (h+max(maxP,maxQ)):Nt){
  update.mod.predict[v]=first.mod.mu+
    x[(v-h):(v-maxP-h+1)]%*%first.mod.coef[1:maxP]+
    update.mod.error[(v-h):(v-maxQ-h+1)]%*%first.mod.coef[-(1:maxP)]
  update.mod.error[v]=x[v]-update.mod.predict[v]
}
update.dataQ=foreach(q=1:maxQ,.combine=cbind)%do%{
  lag.func(update.mod.error,k=(q+h-1))
}

if(updateMA){
  update.mod.predict=rep(NA,length(x))
  update.mod.error=rep(0,length(x))
  for(v in (h+max(maxP,maxQ)):Nt){
    update.mod.predict[v]=first.mod.mu+
      x[(v-h):(v-maxP-h+1)]%*%first.mod.coef[1:maxP]+
      update.mod.error[(v-h):(v-maxQ-h+1)]%*%first.mod.coef[-(1:maxP)]
    update.mod.error[v]=x[v]-update.mod.predict[v]
  }
  update.dataQ=foreach(q=1:maxQ,.combine=cbind)%do%{
    lag.func(update.mod.error,k=(q+h-1))
  }
  second.modX=as.matrix(cbind(dataP,
              update.dataQ))[-(1:(max(maxP,maxQ)+h-1)),]
  second.y=x[-(1:(max(maxP,maxQ)+h-1))]
}else{
  second.modX=first.modX
  second.y=first.y
}

lambda2.seq=cv.glmnet(y=second.y,x=second.modX,standardize=T,parallel=F,
            alpha=1,penalty.factor=weights)$lambda
SQDEV2=foreach(f=1:nfolds,.combine=rbind)%do%{
  in.train=which(Fold.Matrix[,f]==0)
  in.test=which(Fold.Matrix[,f]==1)
  first.mod.est=glmnet(y=second.y[in.train],x=second.modX[in.train,],
                standardize=T,alpha=1,lambda=lambda2.seq,
                penalty.factor=weights)
  first.mod.res=(second.y[in.test]-predict(first.mod.est,
                newx=second.modX[in.test,]))^2
  first.mod.res
}

CVM2=apply(SQDEV2,2,mean)
lambda2.min=lambda2.seq[which.min(CVM2)]
lambda2.1se=lambda2.seq[min(which(CVM2<(min(CVM2)+
            sd(CVM2)/sqrt(length(CVM2)))))]

final.mod.est=glmnet(y=second.y,x=second.modX,standardize=T,
              alpha=1,lambda=lambda2.1se,penalty.factor=weights)
final.mod.coef=as.numeric(coef(final.mod.est))[-1]
nonzero.select=which(final.mod.coef!=0)
final.mod.int=as.numeric(coef(final.mod.est))[1]
final.mod.s2=sum((second.y-predict(final.mod.est,
             newx=second.modX,s=lambda2.1se,
             method="lambda"))^2)/(length(second.y)-
             sum(final.mod.coef[nonzero.select]!=0)-1)

out=list(final.mod.coef=final.mod.coef, #Final Selection of Coefficients
         final.mod.int=final.mod.int,   #Final Estimated Intercept
         final.mod.s2=final.mod.s2,     #Final Estimated Noise Variance
         nonzero.select=nonzero.select) #Identifies Nonzero Parameters
```

```
    }
  }

  #Estimation Via ADENET
  if(Method=="ADENET"){
    if(CV=="KFOLD"){
      Fold.Matrix=NonDepCV2.func(first.y,max.pq=max.pq,K=K)
      nfolds=dim(Fold.Matrix)[2]

      first.cv.out=NULL
      lambda1.seq=cv.glmnet(parallel=F,y=first.y,x=first.modX,standardize=T,
                 alpha=1)$lambda
      SQDEV1=foreach(f=1:nfolds,.combine=rbind)%do%{
        in.train=which(Fold.Matrix[,f]==0)
        in.test=which(Fold.Matrix[,f]==1)
        first.mod.est=glmnet(y=first.y[in.train],x=first.modX[in.train,],
                        standardize=T,alpha=1,lambda=lambda1.seq)
        first.mod.res=(first.y[in.test]-predict(first.mod.est,
                        newx=first.modX[in.test,]))^2
        first.mod.res
      }
      CVM1=apply(SQDEV1,2,mean)
      lambda1.min=lambda1.seq[which.min(CVM1)]
      lambda1.1se=lambda1.seq[min(which(CVM1<(min(CVM1)+
                 sd(CVM1)/sqrt(length(CVM1)))))]
      first.cv.out=rbind(first.cv.out,c(1,lambda1.min,min(CVM1)))


      first.mod.alpha=1
      first.mod.lambda=first.cv.out[which.min(first.cv.out[,3]),2]
      first.mod.est=glmnet(y=first.y,x=first.modX,standardize=T,
                    alpha=first.mod.alpha,lambda=first.mod.lambda)
      first.mod.coef=as.numeric(coef(first.mod.est))[-1]
      first.mod.mu=as.numeric(coef(first.mod.est))[1]

      weights=abs(first.mod.coef+1/length(first.y))^(-eta)

      if(updateMA){
        update.mod.predict=rep(NA,length(x))
        update.mod.error=rep(0,length(x))
        for(v in (h+max(maxP,maxQ)):Nt){
          update.mod.predict[v]=first.mod.mu+
            x[(v-h):(v-maxP-h+1)]%*%first.mod.coef[1:maxP]+
            update.mod.error[(v-h):(v-maxQ-h+1)]%*%first.mod.coef[-(1:maxP)]
          update.mod.error[v]=x[v]-update.mod.predict[v]
        }
        update.dataQ=foreach(q=1:maxQ,.combine=cbind)%do%{
          lag.func(update.mod.error,k=(q+h-1))
        }
        second.modX=as.matrix(cbind(dataP,
                    update.dataQ))[-(1:(max(maxP,maxQ)+h-1)),]
        second.y=x[-(1:(max(maxP,maxQ)+h-1))]
      }else{
        second.modX=first.modX
        second.y=first.y
      }

      second.cv.out=NULL
      for(a in 1:n.alpha){
        lambda2.seq=cv.glmnet(parallel=F,y=second.y,x=second.modX,standardize=T,
                    alpha=alpha[a],penalty.factor=weights)$lambda
        SQDEV2=foreach(f=1:nfolds,.combine=rbind)%do%{
          in.train=which(Fold.Matrix[,f]==0)
          in.test=which(Fold.Matrix[,f]==1)
          second.mod.est=glmnet(y=second.y[in.train],x=second.modX[in.train,],
                   standardize=T,alpha=alpha[a],
```

```
            penalty.factor=weights,lambda=lambda2.seq)
        second.mod.res=(second.y[in.test]-predict(second.mod.est,
                    newx=second.modX[in.test,]))^2
        second.mod.res
      }
      CVM2=apply(SQDEV2,2,mean)
      lambda2.min=lambda2.seq[which.min(CVM2)]
      lambda2.1se=lambda2.seq[min(which(CVM2<(min(CVM2)+
                  sd(CVM2)/sqrt(length(CVM2)))))]
      if(ADENET.final=="min") second.cv.out=rbind(second.cv.out,c(alpha[a],
                    lambda2.min,min(CVM2)))
      if(ADENET.final=="1se") second.cv.out=rbind(second.cv.out,c(alpha[a],
                    lambda2.1se,CVM2[min(which(CVM2<(min(CVM2)+
                    sd(CVM2)/sqrt(length(CVM2)))))]))
    }

    second.mod.alpha=alpha[which.min(second.cv.out[,3])]
    second.mod.lambda=second.cv.out[which.min(second.cv.out[,3]),2]
    second.mod.est=glmnet(y=second.y,x=second.modX,standardize=T,
                  alpha=second.mod.alpha,lambda=second.mod.lambda,
                  penalty.factor=weights)
    second.mod.coef=as.numeric(coef(second.mod.est))[-1]
    second.mod.mu=as.numeric(coef(second.mod.est))[1]

    final.mod.coef=second.mod.coef
    nonzero.select=which(final.mod.coef!=0)
    final.mod.int=second.mod.mu
    final.mod.s2=sum((second.y-predict(second.mod.est,newx=second.modX))^2)/
              (length(second.y)-sum(final.mod.coef[nonzero.select]!=0)-1)

    out=list(final.mod.coef=final.mod.coef, #Final Selection of Coefficients
          final.mod.int=final.mod.int,    #Final Estimated Intercept
          final.mod.s2=final.mod.s2,      #Final Estimated Noise Variance
          nonzero.select=nonzero.select) #Identifies Nonzero Parameters
  }else{
    Fold.Matrix=NonDepCV1.func(first.y,max.pq=max.pq,K=K)
    nfolds=dim(Fold.Matrix)[2]

    first.cv.out=NULL
    lambda1.seq=cv.glmnet(parallel=F,y=first.y,x=first.modX,standardize=T,
              alpha=1)$lambda
    SQDEV1=foreach(f=1:nfolds,.combine=rbind)%do%{
      in.train=which(Fold.Matrix[,f]==0)
      in.test=which(Fold.Matrix[,f]==1)
      first.mod.est=glmnet(y=first.y[in.train],x=first.modX[in.train,],
                  standardize=T,alpha=1,lambda=lambda1.seq)
      first.mod.res=(first.y[in.test]-predict(first.mod.est,
                  newx=first.modX[in.test,]))^2
      first.mod.res
    }
    CVM1=apply(SQDEV1,2,mean)
    lambda1.min=lambda1.seq[which.min(CVM1)]
    lambda1.1se=lambda1.seq[min(which(CVM1<(min(CVM1)+
              sd(CVM1)/sqrt(length(CVM1)))))]
    first.cv.out=rbind(first.cv.out,c(1,lambda1.min,min(CVM1)))


    first.mod.alpha=1
    first.mod.lambda=first.cv.out[which.min(first.cv.out[,3]),2]
    first.mod.est=glmnet(y=first.y,x=first.modX,standardize=T,
                alpha=first.mod.alpha,lambda=first.mod.lambda)
    first.mod.coef=as.numeric(coef(first.mod.est))[-1]
    first.mod.mu=as.numeric(coef(first.mod.est))[1]

    weights=abs(first.mod.coef+1/length(first.y))^(-eta)
```

```
if(updateMA){
  update.mod.predict=rep(NA,length(x))
  update.mod.error=rep(0,length(x))
  for(v in (h+max(maxP,maxQ)):Nt){
    update.mod.predict[v]=first.mod.mu+
      x[(v-h):(v-maxP-h+1)]%*%first.mod.coef[1:maxP]+
      update.mod.error[(v-h):(v-maxQ-h+1)]%*%first.mod.coef[-(1:maxP)]
    update.mod.error[v]=x[v]-update.mod.predict[v]
  }
  update.dataQ=foreach(q=1:maxQ,.combine=cbind)%do%{
    lag.func(update.mod.error,k=(q+h-1))
  }
  second.modX=as.matrix(cbind(dataP,
               update.dataQ))[-(1:(max(maxP,maxQ)+h-1)),]
  second.y=x[-(1:(max(maxP,maxQ)+h-1))]
}else{
  second.modX=first.modX
  second.y=first.y
}

second.cv.out=NULL
for(a in 1:n.alpha){
  lambda2.seq=cv.glmnet(parallel=F,y=second.y,x=second.modX,standardize=T,
               alpha=alpha[a],penalty.factor=weights)$lambda
  SQDEV2=foreach(f=1:nfolds,.combine=rbind)%do%{
    in.train=which(Fold.Matrix[,f]==0)
    in.test=which(Fold.Matrix[,f]==1)
    second.mod.est=glmnet(y=second.y[in.train],x=second.modX[in.train,],
                   standardize=T,alpha=alpha[a],penalty.factor=weights,
                   lambda=lambda2.seq)
    second.mod.res=(second.y[in.test]-predict(second.mod.est,
                    newx=second.modX[in.test,]))^2
    second.mod.res
  }
  CVM2=apply(SQDEV2,2,mean)
  lambda2.min=lambda2.seq[which.min(CVM2)]
  lambda2.1se=lambda2.seq[min(which(CVM2<(min(CVM2)+
               sd(CVM2)/sqrt(length(CVM2)))))]
  if(ADENET.final=="min") second.cv.out=rbind(second.cv.out,
                          c(alpha[a],lambda2.min,min(CVM2)))
  if(ADENET.final=="1se") second.cv.out=rbind(second.cv.out,c(alpha[a],
                          lambda2.1se,CVM2[min(which(CVM2<(min(CVM2)+
                          sd(CVM2)/sqrt(length(CVM2)))))]))
}

second.mod.alpha=alpha[which.min(second.cv.out[,3])]
second.mod.lambda=second.cv.out[which.min(second.cv.out[,3]),2]
second.mod.est=glmnet(y=second.y,x=second.modX,standardize=T,
                alpha=second.mod.alpha,lambda=second.mod.lambda,
                penalty.factor=weights)
second.mod.coef=as.numeric(coef(second.mod.est))[-1]
second.mod.mu=as.numeric(coef(second.mod.est))[1]

final.mod.coef=second.mod.coef
nonzero.select=which(final.mod.coef!=0)
final.mod.int=second.mod.mu
final.mod.s2=sum((second.y-predict(second.mod.est,newx=second.modX))^2)/
  (length(second.y)-sum(final.mod.coef[nonzero.select]!=0)-1)

out=list(final.mod.coef=final.mod.coef, #Final Selection of Coefficients
         final.mod.int=final.mod.int,   #Final Estimated Intercept
         final.mod.s2=final.mod.s2,     #Final Estimated Noise Variance
         nonzero.select=nonzero.select) #Identifies Nonzero Parameters
  }
}
```

```
  return(out)
}

#Illustration of Function for ADLASSO Estimation
adlasso9=adshrink91011.func(x=maunaloa.co2.train,h=1,
          long.ar.select=F,maxP=14,maxQ=14,
          updateMA=F,eta=2,alpha=seq(0,1,0.1),max.pq=max(14,14),
          Method="ADLASSO",K=5,CV="KFOLD")
adlasso10=adshrink91011.func(x=maunaloa.co2.train,h=1,
           long.ar.select=F,maxP=14,maxQ=14,
           updateMA=F,eta=2,alpha=seq(0,1,0.1),max.pq=max(14,14),
           Method="ADLASSO",K=10,CV="KFOLD")
adlasso11=adshrink91011.func(x=maunaloa.co2.train,h=1,
           long.ar.select=F,maxP=14,maxQ=14,
           updateMA=F,eta=2,alpha=seq(0,1,0.1),max.pq=max(14,14),
           Method="ADLASSO",CV="LOBOCV")

#Illustration of Function for ADENET Estimation
adenet9=adshrink91011.func(x=maunaloa.co2.train,h=1,
         long.ar.select=F,maxP=14,maxQ=14,
         updateMA=F,eta=2,alpha=seq(0,1,0.1),max.pq=max(14,14),
         ADENET.final="min",Method="ADENET",K=5,CV="KFOLD")
adenet10=adshrink91011.func(x=maunaloa.co2.train,h=1,
          long.ar.select=F,maxP=14,maxQ=14,
          updateMA=F,eta=2,alpha=seq(0,1,0.1),max.pq=max(14,14),
          ADENET.final="min",Method="ADENET",K=10,CV="KFOLD")
adenet11=adshrink91011.func(x=maunaloa.co2.train,h=1,
          long.ar.select=F,maxP=14,maxQ=14,
          updateMA=F,eta=2,alpha=seq(0,1,0.1),max.pq=max(14,14),
          ADENET.final="min",Method="ADENET",CV="LOBOCV")
##############################################################################


########################################################################
#Function to Obtain Projected Posterior Distribution from Full Posterior
#Arguments: fullpost = Posterior Distribution from BHS estimated model
#             X = Model Matrix of Full Model
#             indproj = Vector Indicating which Columns are Included
#Source: Piironen and Vehtari (2015)
########################################################################

#Essential Function for Predictive Posterior Projection Method
pms.proj.func<-function(fullpost,X,indproj){

  #Transpose Matrix of Posterior Samples of ARMA Coefficients
  lres.coef=t(fullpost[,-dim(fullpost)[2]])

  #Vector of Posterior Samples of Variance Parameter
  lres.s2=(fullpost[,dim(fullpost)[2]])
  S=length(lres.s2)   #Number of Posterior Samples Obtained
  N=dim(X)[1]   #Number of Points in Data Set

  P=dim(X)[2]
  COEF.PROJ=matrix(0,S,P)
  X.proj=X[,indproj]
  pred.proj=X%*%lres.coef

  coef.proj=solve(t(X.proj)%*%X.proj)%*%t(X.proj)%*%pred.proj
  var.proj=c(lres.s2)+colMeans((pred.proj-X.proj%*%coef.proj)^2)
  KL.PROJ=0.5*log(var.proj/lres.s2)
  COEF.PROJ[,indproj]=t(coef.proj)
  KL.MEAN=mean(KL.PROJ)
  COEF.MEAN=colMeans(COEF.PROJ)
  VAR.PROJ=var.proj
  VAR.MEAN=mean(var.proj)
  return(list(KL.MEAN=KL.MEAN,      #Average KL Divergence
```

```
                COEF.MEAN=COEF.MEAN, #Posterior Mean of Coefficients
                COEF.PROJ=COEF.PROJ, #Projected Posterior of Coefficients
                VAR.MEAN=VAR.MEAN,   #Posterior Mean of Variance
                VAR.PROJ=VAR.PROJ))  #Projected Posterior of Variance
}
###########################################################################


###########################################################################
#Function to Conduct ARMA Selection via Bayesian Projection Posterior
#    Predictive Distribution Implementing Relative Efficiency
#    for Final Model Selection
#
#Arguments: x = Time Series to Be Modeled Using ARMA Process
#           h = Horizon Specific Model (Defaults to 1)
#           maxP = Maximum Autoregressive Order
#           maxQ = Maximum Moving Average Order
#           updateMA = Indicator Determining if Moving Average Terms Should Be
#                      Updated After Initial Coefficients Selected
#                      (Defaults to F)
#           prior.choice = Choose Between Bayesian Horseshoe Prior ("hs") and
#                          Bayesian Horseshoe+ Prior ("hs+")
#           KL.threshold = Single value or vector of chosen thresholds
#                          for stopping rule based on Relative Efficiency
#                          Comparing Submodel to Full Model
#                          based on Kullback Leibler Divergence
#                          (Defaults to c(0.9,0.95,0.99))
#Source: Piironen and Vehtari (2015)
################################################ ##########################

#Creation of Function
pms123.func<-function(x,h=1,maxP,maxQ,KL.threshold=c(0.90,0.95,0.99),
                      prior.choice=c("hs","hs+"),updateMA=F){
  require(MCMCpack)
  require(bayesreg)

  prior.choice=match.arg(prior.choice)

  N=length(x)
  max.ar.order=ceiling(10*log10(N))

  init.modX=foreach(init.ar=1:max.ar.order,.combine=cbind)%do%{
    lag.func(x,k=(init.ar+h-1))
  }

  init.data=data.frame(y=x,init.modX)
  init.data=init.data[-(1:(max.ar.order+h-1)),]

  init.mod.est=MCMCregress(y~.,data=init.data,mcmc=2000,thin=10,burnin=10000)
  muBeta0=mean(init.mod.est[,1])
  muBeta=colMeans(init.mod.est[,-c(1,dim(init.mod.est)[2])])
  init.mod.error=init.data$y-(as.numeric(muBeta0) +
                 as.matrix(init.data[,-1])%*%as.vector(muBeta))

  dataP=foreach(p=1:maxP,.combine=cbind)%do%{
    lag.func(init.data$y,k=(p+h-1))
  }

  dataQ=foreach(q=1:maxQ,.combine=cbind)%do%{
    lag.func(init.mod.error,k=(q+h-1))
  }

  full.data=data.frame(y=init.data$y,dataP=dataP,dataQ=dataQ)
  full.data=full.data[-(1:(max(maxP,maxQ)+h-1)),]

  full.mod.est=bayesreg(y~.,data=full.data,prior=prior.choice,
                nsamples=2000,thin=10,burnin=10000)
```

```
full.mod.posterior=cbind(as.vector(full.mod.est$beta0),
                         t(as.matrix(full.mod.est$beta)),
                         as.vector(full.mod.est$sigma2))

full.mod.int=c(full.mod.est$muBeta0)
full.mod.coef=c(full.mod.est$muBeta)
full.mod.s2=full.mod.est$muSigma2

if(updateMA){
  full.mod.predict=rep(NA,length(x))
  full.mod.error=rep(0,length(x))
  for(k in (h+max(maxP,maxQ)):N){
    full.mod.predict[k]=full.mod.int+
      x[(k-h):(k-maxP-h+1)]%*%full.mod.coef[1:maxP]+
      full.mod.error[(k-h):(k-maxQ-h+1)]%*%full.mod.coef[-(1:maxP)]
    full.mod.error[k]=x[k]-full.mod.predict[k]
  }

  full.dataP=foreach(p=1:maxP,.combine=cbind)%do%{
    lag.func(x,k=(p+h-1))
  }

  update.dataQ=foreach(p=1:maxQ,.combine=cbind)%do%{
    lag.func(full.mod.error,k=(p+h-1))
  }

  full.mod.X=cbind(1,full.dataP,update.dataQ)
  full.mod.X=full.mod.X[-(1:(max(maxP,maxQ)+h-1)),]
  nMod=dim(full.mod.X)[2]
}else{
  full.mod.X=as.matrix(cbind(1,full.data[,-1]))
  nMod=dim(full.mod.X)[2]
}


KL=rep(NA,nMod)
chosen=1
notchosen=setdiff(1:nMod,chosen)
FIRST=pms.proj.func(fullpost=full.mod.posterior,X=full.mod.X,indproj=chosen)
KL[1]=FIRST$KL.MEAN

for(modnum in 2:nMod){
  nleft<-length(notchosen)
  val=foreach(j=1:nleft,.combine=c)%do%{
    ind<-sort(c(chosen,notchosen[j]))
    NEXT<-tryCatch({pms.proj.func(fullpost=full.mod.posterior,
                                  X=full.mod.X,indproj=ind)$KL.MEAN},
                   error=function(e){return(NA)})
    NEXT
  }
  minval<-which.min(val)
  chosen<-c(chosen,notchosen[minval])
  notchosen<-setdiff(1:nMod,chosen)
  KL[modnum]<-val[minval]
}

nKL.threshold=length(KL.threshold)
KL.select=matrix(rep(chosen,nKL.threshold),ncol=nKL.threshold)
final.mod.posterior=list()
final.mod.int=list()
final.mod.coef=list()
final.mod.s2=list()

for(k in 1:nKL.threshold){
  TEMP1=min(which((1-KL/KL[1])>KL.threshold[k]))
```

```
    TEMP2=pms.proj.func(fullpost=full.mod.posterior,
                        X=full.mod.X, indproj=chosen[1:TEMP1])
    KL.select[-(1:TEMP1),k]=NA
    final.mod.posterior[[k]]=cbind(TEMP2$COEF.PROJ,TEMP2$VAR.PROJ)
    final.mod.int[[k]]=TEMP2$COEF.MEAN[1]
    final.mod.coef[[k]]=TEMP2$COEF.MEAN[-1]
    final.mod.s2[[k]]=TEMP2$VAR.MEAN
  }

  return(list(CHOICE=chosen,  #Full Path of Variables in Order Of Selection
                              # In Forward Search Algorithm
              KL=KL,  #Full Path of Kullback Leibler Divergences in Order
                      # Of Selection in Forward Algorithm
              KL.threshold=KL.threshold, #Reoutputs the Thresholds Considered

              #Matrix where Each Column Identifies the Parameters Selected
              #Based on a Specific Threshold
              KL.select=KL.select,

              #Estimated Intercept Before Selection
              full.mod.int=full.mod.int,
              #Estimated Coefficients Before Selection
              full.mod.coef=full.mod.coef,
              #Estimated Noise Variance Before Selection
              full.mod.s2=full.mod.s2,

              #Lists Where Each Element Corresponds to a
              #Specific Threshold in KL.threshold

              #Final Estimated Intercept for each Threshold
              final.mod.int=final.mod.int,
              #Final Selection of Coefficients for each Threshold
              final.mod.coef=final.mod.coef,
              #Final Estimated Noise Variance for each Threshold
              final.mod.s2=final.mod.s2))
}

#Illustration of Function for BHS Estimation
bhs123.out=pms123.func(x=maunaloa.co2.train,h=1,maxP=14,maxQ=14,
                       KL.threshold=c(0.90,0.95,0.98),
                       prior.choice="hs",updateMA=F)
bhs1=list(final.mod.coef=bhs123.out$final.mod.coef[[1]],
          final.mod.int=bhs123.out$final.mod.int[[1]],
          final.mod.s2=bhs123.out$final.mod.s2[[1]],
          nonzero.select=which(bhs123.out$final.mod.coef[[1]]!=0))
bhs2=list(final.mod.coef=bhs123.out$final.mod.coef[[2]],
          final.mod.int=bhs123.out$final.mod.int[[2]],
          final.mod.s2=bhs123.out$final.mod.s2[[2]],
          nonzero.select=which(bhs123.out$final.mod.coef[[2]]!=0))
bhs3=list(final.mod.coef=bhs123.out$final.mod.coef[[3]],
          final.mod.int=bhs123.out$final.mod.int[[3]],
          final.mod.s2=bhs123.out$final.mod.s2[[3]],
          nonzero.select=which(bhs123.out$final.mod.coef[[3]]!=0))

#Illustration of Function for BHS+ Estimation
bhsp123.out=pms123.func(x=maunaloa.co2.train,h=1,maxP=14,maxQ=14,
                        KL.threshold=c(0.90,0.95,0.98),
                        prior.choice="hs+",updateMA=F)
bhsp1=list(final.mod.coef=bhsp123.out$final.mod.coef[[1]],
           final.mod.int=bhsp123.out$final.mod.int[[1]],
           final.mod.s2=bhsp123.out$final.mod.s2[[1]],
           nonzero.select=which(bhsp123.out$final.mod.coef[[1]]!=0))
bhsp2=list(final.mod.coef=bhsp123.out$final.mod.coef[[2]],
           final.mod.int=bhsp123.out$final.mod.int[[2]],
           final.mod.s2=bhsp123.out$final.mod.s2[[2]],
           nonzero.select=which(bhsp123.out$final.mod.coef[[2]]!=0))
bhsp3=list(final.mod.coef=bhsp123.out$final.mod.coef[[3]],
```

```
        final.mod.int=bhsp123.out$final.mod.int[[3]],
        final.mod.s2=bhsp123.out$final.mod.s2[[3]],
        nonzero.select=which(bhsp123.out$final.mod.coef[[3]]!=0))
##########################################################################


##########################################################################
#Function to Conduct ARMA Selection via
#   Bayesian Projection Posterior Predictive Distribution
#   Implementing Out-of-Sample Based Final Model Selection
#
#Arguments: x = Time Series to Be Modeled Using ARMA Process
#           h = Horizon Specific Model (Defaults to 1)
#           maxP = Maximum Autoregressive Order
#           maxQ = Maximum Moving Average Order
#           updateMA = Indicator Determining if Moving Average Terms Is
#                      Updated After Initial Coefficients Selected
#                      (Defaults to F)
#           KL.threshold = Single value or vector of chosen thresholds
#                          for stopping rule based on Relative Efficiency
#                          Comparing Submodel to Full Model
#                          based on Kullback Leibler Divergence
#                          (Defaults to c(0.9,0.95,0.99))
#           KL.stop = Stopping Rule
#    (Select from Considered Models Where Relative Efficiency < KL.stop)
#Source: Piironen and Vehtari (2015)
##########################################################################

#Creation of Function
pms4.func<-function(x,h=1,maxP,maxQ,KL.stop=0.98,test.per=0.2,
                    prior.choice=c("hs","hs+"),updateMA=F){
  require(MCMCpack)
  require(bayesreg)

  prior.choice=match.arg(prior.choice)

  cv.vector=OOS.IndepCV.func(x,test.per=test.per)

  x.train=x[cv.vector==0]
  x.test=x[cv.vector==1]

  Nt=length(x.train)
  max.ar.order=ceiling(10*log10(Nt))

  init.modX=foreach(init.ar=1:max.ar.order,.combine=cbind)%do%{
    lag.func(x.train,k=(init.ar+h-1))
  }

  init.data=data.frame(y=x.train,init.modX)
  init.data=init.data[-(1:(max.ar.order+h-1)),]

  init.mod.est=MCMCregress(y~.,data=init.data,mcmc=2000,thin=10,burnin=10000)
  muBeta0=mean(init.mod.est[,1])
  muBeta=colMeans(init.mod.est[,-c(1,dim(init.mod.est)[2])])
  init.mod.error=init.data$y-(as.numeric(muBeta0) +
                 as.matrix(init.data[,-1])%*%as.vector(muBeta))

  dataP=foreach(p=1:maxP,.combine=cbind)%do%{
    lag.func(init.data$y,k=(p+h-1))
  }

  dataQ=foreach(q=1:maxQ,.combine=cbind)%do%{
    lag.func(init.mod.error,k=(q+h-1))
  }

  full.data=data.frame(y=init.data$y,dataP=dataP,dataQ=dataQ)
  full.data=full.data[-(1:(max(maxP,maxQ)+h-1)),]
```

```
##############################################
#xc.mean=as.numeric(colMeans(full.data))
#xc.sd=as.numeric(apply(full.data,2,sd))
#full.data=as.data.frame(scale(full.data))
###############################################

full.mod.est=bayesreg(y~.,data=full.data,prior=prior.choice,nsamples=2000,
                      thin=10,burnin=10000)
full.mod.posterior=cbind(as.vector(full.mod.est$beta0),
                         t(as.matrix(full.mod.est$beta)),
                         as.vector(full.mod.est$sigma2))

full.mod.int=c(full.mod.est$muBeta0)
full.mod.coef=c(full.mod.est$muBeta)
full.mod.s2=full.mod.est$muSigma2

if(updateMA){
  full.mod.predict=rep(NA,length(x.train))
  full.mod.error=rep(0,length(x.train))
  for(k in (h+max(maxP,maxQ)):Nt){
    full.mod.predict[k]=full.mod.int+
      x.train[(k-h):(k-maxP-h+1)]%*%full.mod.coef[1:maxP]+
      full.mod.error[(k-h):(k-maxQ-h+1)]%*%full.mod.coef[-(1:maxP)]
    full.mod.error[k]=x.train[k]-full.mod.predict[k]
  }

  full.dataP=foreach(p=1:maxP,.combine=cbind)%do%{
    lag.func(x.train,k=(p+h-1))
  }

  update.dataQ=foreach(p=1:maxQ,.combine=cbind)%do%{
    lag.func(full.mod.error,k=(p+h-1))
  }

  full.mod.X=cbind(1,full.dataP,update.dataQ)
  full.mod.X=full.mod.X[-(1:(max(maxP,maxQ)+h-1)),]
  nMod=dim(full.mod.X)[2]
}else{
  full.mod.X=as.matrix(cbind(1,full.data[,-1]))
  nMod=dim(full.mod.X)[2]
}

KL=rep(NA,nMod)
chosen=1
notchosen=setdiff(1:nMod,chosen)
FIRST=pms.proj.func(fullpost=full.mod.posterior,X=full.mod.X,indproj=chosen)
KL[1]=FIRST$KL.MEAN
modnum=2
check=0

while(check<KL.stop | modnum==nMod){
  nleft<-length(notchosen)

  val=foreach(j=1:nleft,.combine=c)%do%{
    ind<-sort(c(chosen,notchosen[j]))
    NEXT<-tryCatch({pms.proj.func(fullpost=full.mod.posterior,
                    X=full.mod.X,indproj=ind)$KL.MEAN},
                    error=function(e){return(NA)})
    NEXT
  }
  minval<-which.min(val)
  chosen<-c(chosen,notchosen[minval])
  notchosen<-setdiff(1:nMod,chosen)
  KL[modnum]<-val[minval]
  check=1-KL[modnum]/KL[1]
```

```
    modnum=modnum+1
  }

  KL=KL[!is.na(KL)]
  chosen=chosen[!is.na(chosen)]

  nMod2=length(KL)
  MSE=rep(NA,nMod2)

  for(modnum in 1:nMod2){
    out=pms.proj.func(fullpost=full.mod.posterior,X=full.mod.X,
                      indproj=chosen[1:modnum])
    coef=out$COEF.MEAN
    int=coef[1]
    coef.ar=coef[2:(1+maxP)]
    coef.ma=coef[-(1:(maxP+1))]

    predictx.test=rep(NA,length(x.test))
    errorx.test=rep(0,length(x.test))
    for(k in (h+max(maxP,maxQ)):length(x.test)){
      predictx.test[k]=int+x.test[(k-h):(k-maxP-h+1)]%*%coef.ar+
        errorx.test[(k-h):(k-maxQ-h+1)]%*%coef.ma
      errorx.test[k]=x.test[k]-predictx.test[k]
    }

    MSE[modnum]=mean((x.test-predictx.test)^2,na.rm=T)
  }
  best.mod=chosen[1:which.min(MSE)]
  out.mod=pms.proj.func(fullpost=full.mod.posterior,
                        X=full.mod.X,indproj=best.mod)

  final.mod.int=out.mod$COEF.MEAN[1]
  final.mod.coef=out.mod$COEF.MEAN[-1]
  final.mod.s2=out.mod$VAR.MEAN

  nonzero.select=which(final.mod.coef!=0)

  return(list(CHOICE=chosen,  #Full Path of Variables in Order Of Selection
              #  In Forward Search Algorithm
              KL=KL,  #Full Path of Kullback Leibler Divergences in Order
              #  Of Selection in Forward Algorithm
              KL.threshold=KL.threshold, #Reoutputs the Thresholds Considered

              #Matrix where Each Column Identifies the Parameters Selected
              #Based on a Specific Threshold
              KL.select=KL.select,

              #Estimated Intercept Before Selection
              full.mod.int=full.mod.int,
              #Estimated Coefficients Before Selection
              full.mod.coef=full.mod.coef,
              #Estimated Noise Variance Before Selection
              full.mod.s2=full.mod.s2,

              #Lists Where Each Element Corresponds to a
              #Specific Threshold in KL.threshold

              #Final Estimated Intercept for each Threshold
              final.mod.int=final.mod.int,
              #Final Selection of Coefficients for each Threshold
              final.mod.coef=final.mod.coef,
              #Final Estimated Noise Variance for each Threshold
              final.mod.s2=final.mod.s2))
}

#Illustration of Function for BHS Estimation
bhs4=pms4.func(x=maunaloa.co2.train,h=1,maxP=14,maxQ=14,KL.stop=0.98,
               test.per=0.2,prior.choice="hs",updateMA=F)
```

```
#Illustration of Function for BHS+ Estimation
bhsp4=pms4.func(x=maunaloa.co2.train,h=1,maxP=14,maxQ=14,KL.stop=0.98,
                                test.per=0.2,prior.choice="hs+",updateMA=F)
################################################################################


################################################################################
#Obtain Forecasts for All Models
#
#List of All Models

MODELS=list(adlasso1,adlasso2,adlasso3,adlasso4,adlasso5,
             adlasso6,adlasso7,adlasso8,
             adlasso9,adlasso10,adlasso11,
             adenet1,adenet2,adenet3,adenet4,adenet5,
             adenet6,adenet7,adenet8,adenet9,
             adenet10,adenet11,
             bhs1,bhs2,bhs3,bhs4,
             bhsp1,bhsp2,bhsp3,bhsp4)

#
################################################################################

#Total Number of Models Estimated
nMODELS=length(MODELS)

#Each Row of the Following Matrices Corresponds to a Different Final Model

#Matrix of Binary Variables Indicated Selection
SELECT=matrix(0,nMODELS,ncol=28)
#Matrix of Coefficients Estimated from All Models
COEF=matrix(0,nMODELS,ncol=28)

#1-step Ahead Forecasting Results
RMSFE=rep(NA,nMODELS) #Matrix of Root mean squared Forecast Error
MAPFE=rep(NA,nMODELS) #Matrix of Mean Absolute Percentage Forecast Error
MFB=rep(NA,nMODELS)   #Matrix of Mean Forecast Bias
MDFB=rep(NA,nMODELS)  #Matrix of Mean Directional Forecast Bias

#Matrices of Lower, Upper, and Mean Forecasts
#   (Monte Carlo 1-step Ahead Forecast Distribution)
#   Columns Correspond for each of the models estimated

#5% Quantile of Monte Carlo Distribution
FC.LOWER=matrix(NA,sum(VALIDATION.PERIOD),nMODELS)
#Mean of Monte Carlo Distribution
FC.MEAN=matrix(NA,sum(VALIDATION.PERIOD),nMODELS)
#95% Quantile of Monte Carlo Distribution
FC.UPPER=matrix(NA,sum(VALIDATION.PERIOD),nMODELS)

for(k in 1:nMODELS){
  temp.mod=MODELS[[k]]
  SELECT[k,temp.mod$nonzero.select]=1
  COEF[k,]=temp.mod$final.mod.coef

  fc.lower=rep(NA,length(maunaloa.co2.final))
  fc.mean=rep(NA,length(maunaloa.co2.final))
  fc.upper=rep(NA,length(maunaloa.co2.final))

  error=rep(0,length(maunaloa.co2.final))

  for(j in 30:length(maunaloa.co2.final)){
    fc=as.numeric(maunaloa.co2.final[(j-1):(j-14)]%*%
                  temp.mod$final.mod.coef[1:14]+
                  error[(j-1):(j-14)]%*%temp.mod$final.mod.coef[-(1:14)]) +
      rnorm(100000,mean=temp.mod$final.mod.int,sd=sqrt(temp.mod$final.mod.s2))
```

```
    fc.lower[j]=quantile(fc,0.05)
    fc.mean[j]=mean(fc)
    fc.upper[j]=quantile(fc,0.95)
    error[j]=maunaloa.co2.final[j]-fc.mean[j]
  }

  RMSFE[k]=sqrt(mean((maunaloa.co2.final[VALIDATION.PERIOD]-
             fc.mean[VALIDATION.PERIOD])^2))
  MAPFE[k]=100*mean(abs((maunaloa.co2.final[VALIDATION.PERIOD]-
             fc.mean[VALIDATION.PERIOD])/
             maunaloa.co2.final[VALIDATION.PERIOD]))
  MFB[k]=mean(maunaloa.co2.final[VALIDATION.PERIOD]-
           fc.mean[VALIDATION.PERIOD])
  MDFB[k]=(sum((maunaloa.co2.final[VALIDATION.PERIOD]-
             fc.mean[VALIDATION.PERIOD])>0)-
             sum((maunaloa.co2.final[VALIDATION.PERIOD]-
             fc.mean[VALIDATION.PERIOD])<0))/
             sum(VALIDATION.PERIOD)

  FC.LOWER[,k]=fc.lower[VALIDATION.PERIOD]
  FC.MEAN[,k]=fc.mean[VALIDATION.PERIOD]
  FC.UPPER[,k]=fc.upper[VALIDATION.PERIOD]

}
################################################################################


################################################################################
#Check Stationarity and Invertibility of Estimates From All Different Methods
#Based on the COEF matrix above
################################################################################

#Check Stationarity and Invertibility of Estimates
stationarity=rep(NA,nMODELS)
invertibility=rep(NA,nMODELS)
for(k in 1:nMODELS){
  ar.coef=COEF[k,1:14]
  st.poly=c(1,-ar.coef)
  st.root=polyroot(st.poly)
  st.check=sum(abs(st.root)>1)==length(st.root)
  stationarity[k]=st.check

  ma.coef=COEF[k,-(1:14)]
  ma.poly=c(1,ma.coef)
  ma.root=polyroot(ma.poly)
  ma.check=sum(abs(ma.root)>1)==length(ma.root)
  invertibility[k]=ma.check
}

print(stationarity) #All True
print(invertibility) #Models (adlasso4,adlasso5,adenet4,adenet5) Fail
######################## ######################################################
```