exdqlm: An R Package for Estimation and Analysis of Flexible Dynamic Quantile Linear Models

Raquel Barata
University of California Santa Cruz

Raquel Prado
University of California Santa Cruz

Bruno Sansó
University of California Santa Cruz

Abstract

We present the R package **exdqlm** as a tool for dynamic quantile regression. The main focus of the package is to provide a framework for Bayesian inference and forecasting of flexible dynamic quantile linear models (exDQLMs). exDQLMs utilize a new family of error distributions for parametric quantile regression, the extended asymmetric Laplace (exAL), a generalization of the asymmetric Laplace (AL) which is commonly used in parametric quantile regression. Estimation of a dynamic quantile linear model, which utilizes the AL, is included in the package as a special case. Non-time-varying quantile regression models are also a special case of our methods. Further, routines for estimation of a nonlinear relationship between the response and a given input variable at a specified quantile via a transfer function model are available. The software provides the choice of two different algorithms for posterior inference: Markov chain Monte Carlo (MCMC), and importance sampling variational Bayes (ISVB). While MCMC provides an efficient sample-based exploration of the posterior distribution, the approximation obtained by the ISVB algorithm enables fast inference for long time series at a fraction of the memory and computational costs of the MCMC. A routine for forecasting the dynamic quantile is available in the package, as well as several quantitative and visual diagnostics for model evaluation. We illustrate the implementation of the functions and algorithms in the **exdqlm** package with a step-by-step guide for the analysis of several real data sets.

Keywords: dynamic quantile regression, dynamic models, asymmetric Laplace, variational Bayes.

1. Introduction

The purpose of the R (R Core Team 2013) package **exdqlm** is to provide an accessible and comprehensive tool for flexible Bayesian estimation of a single dynamic quantile. Quantile regression is an area that has gained popularity in the non-time-varying setting as a robust alternative to traditional mean-centric approaches. However, time-varying methods and software implementations remain limited. In a Bayesian setting, parametric quantile regression models are almost exclusively based on the asymmetric Laplace (AL) likelihood, as it corresponds to the check loss function that is minimized in traditional quantile regression problems (Yu and Moyeed 2001; Koenker 2005). A mixture representation of the AL provides compu-

tational advantages for the exploration of the posterior distribution of the model parameters. This has contributed to its popularity. However the AL is known to have several shortcomings in terms of flexibility. Very general error distributions for quantile regression have been considered extensively in the Bayesian non-parametric literature (Kottas and Krnjajić 2009; Reich, Bondell, and Wang 2009; Taddy and Kottas 2010), but the computational cost of these methods do not scale well to dynamic models for long times series. Yan and Kottas (2017) present a parametric extension of the AL, the extended asymmetric Laplace (exAL), which overcomes the restrictive aspects of the distribution while preserving its computational advantages. Barata, Prado, and Sansó (in press) utilize the new family of error distributions presented in Yan and Kottas (2017) to develop a flexible dynamic quantile linear model (exDQLM). This is the model implemented in the package **exdqlm**.

Dynamic quantile models typically consist of a set of variables which summarize the properties of the quantile of interest and a structure that defines the evolution of these variables, which together represent the behavior of a quantile over time. Our **exdqlm** package is a powerful and versatile tool for dynamic quantile modeling. The model form is robust enough to estimate any quantile regardless of non-standard features in the data distribution (i.e., skewness or extreme observations). Further, relevant features such as long-term trends or seasonality are easily incorporated into the evolution of the quantile. The package includes both, a Markov chain Monte Carlo (MCMC) algorithm for exact posterior inference, as well as an approximate importance sampling variational Bayes (ISVB) algorithm for fast, yet accurate estimation. The ISVB algorithm makes our methods accessible to applications with very large datasets, and enables quick model exploration and selection. The exDQLM framework facilitates k-step-ahead forecasting for the estimated quantile, also implemented in the package. Further, a transfer function extension is included in the software as a straight-forward method to quantify a non-linear relationship between a response and a given input variable at a specified quantile. Finally, **exdqlm** includes several visual and quantitative model diagnostics. The package is protected under the MIT License, and is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=exdqlm`.

Well-developed software implementations of a non-time-varying quantile regression models which utilize the AL are available in the Bayesian R package **bayesQR** (Benoit, Al-Hamzawi, Yu, and Van den Poel 2011) and, equivalently, a classical version, which utilizes the check-loss function (Koenker 2005), is available in the package **quantreg** (Koenker 2021). As a result of the AL being a special case of the exAL, the methods used in the Bayesian package **bayesQR** are also a special case of our exDQLM implemented in **exdqlm**. Further, there are many established packages for dynamic modeling including **dlm** (Petris and Gilks 2018) and **dynr** (Ou, Hunter, Chow, Ji, Chen, Hung, Lee, Li, and Park 2021), however, implementations of dynamic quantile modeling are much more limited. To our knowledge, the package **quantreg** includes a routine which allows for a time-evolving structure in the regression coefficients (a special case of the routines in our package), but this is the extent of the software available for dynamic quantile regression.

In the remainder of this paper we introduce the package as follows. In Section 2, we describe the modeling framework, including the algorithms used for efficient posterior inference and our methods for including nonlinear relationships. In Section 3 we detail the model diagnostics implemented in the package. Then, in Section 4, we walk through three step-by-step examples on how to use the package. Finally, in Section 5, we present a summary of the package capabilities.

2. Extended dynamic quantile linear models

Consider a set of scalar time-evolving responses, y_t, for times $t = 1, \ldots, T$. For each t, an extended dynamic quantile linear model (exDQLM) for inference on a single p_0-th quantile is defined by

$$\text{Observation equation:} \quad y_t = \mathbf{F}_t'\boldsymbol{\theta}_t + \epsilon_t, \qquad \epsilon_t \sim \text{exAL}_{p_0}(0, \sigma, \gamma) \tag{1}$$

$$\text{System equation:} \quad \boldsymbol{\theta}_t = \mathbf{G}_t\boldsymbol{\theta}_{t-1} + \boldsymbol{\omega}_t, \qquad \boldsymbol{\omega}_t \sim \text{N}(0, \mathbf{W}_t). \tag{2}$$

Here $\mathbf{F}_t$ is the $q \times 1$ regression vector of the covariates corresponding to the $q \times 1$ parameter vector $\boldsymbol{\theta}_t$ at time t, $\mathbf{G}_t$ is the $q \times q$-dimensional evolution matrix defining the structure of the parameter vector evolution in time, and the normal distribution according to which the system error $\boldsymbol{\omega}_t$ evolves is q-variate. The observational errors ϵ_t of the exDQLM follow an exAL distribution for fixed quantile p_0, whose density is denoted as $\text{exAL}_{p_0}(\cdot)$, and it is such that $\int_{-\infty}^{0} \text{exAL}_{p_0}(\epsilon_t|0, \sigma, \gamma)\text{d}\epsilon_t = p_0$. Thus, the observation equation in Equation (1) implies that $\mathbf{F}_t'\boldsymbol{\theta}_t$ corresponds to the p_0-quantile of y_t, even if this is not made explicit in the notation above.

The exAL distribution is constructed from an extension of the location-scale mixture representation of the AL (Kozumi and Kobayashi 2011), details of which can be found in Yan and Kottas (2017). This results in a mixture representation of $\text{exAL}_{p_0}(0, \sigma, \gamma)$ as follows:

$$\int \int_{\mathbb{R}^+ \times \mathbb{R}^+} \text{N}(y_t|C(p, \gamma)\sigma|\gamma|s_t + A(p)v_t, \sigma B(p)v_t)\text{Exp}(v_t|\sigma)\text{N}^+(s_t|0, 1)\text{d}v_t\text{d}s_t. \tag{3}$$

Here, $p = p(p_0, \gamma) = I(\gamma < 0) + \{[p_0 - I(\gamma < 0)]/g(\gamma)\}$ where $g(\gamma) = 2\Phi(-|\gamma|)\exp(\gamma^2/2)$, and $\Phi(\cdot)$ denotes the standard normal CDF. $A(p)$, $B(p)$, $C(p, \gamma)$ are the functions of p and γ: $A(p) = \frac{1-2p}{p(1-p)}$, $B(p) = \frac{2}{p(1-p)}$, $C(p, \gamma) = [I(\gamma > 0) - p]^{-1}$. Lastly, $\text{Exp}(v|\sigma)$ denotes the exponential distribution with mean σ, and $\text{N}^+(s_t|0, 1)$ denotes a normal distribution truncated to the positive reals with mean 0 and variance 1. This mixture representation of the exAL can be exploited to rewrite the exDQLM as the following hierarchical model for $t = 1, \ldots, T$:

$$y_t|\boldsymbol{\theta}_t, \sigma, \gamma, v_t, s_t \sim \text{N}(y_t|\mathbf{F}_t'\boldsymbol{\theta}_t + C(p, \gamma)\sigma|\gamma|s_t + A(p)v_t, \sigma B(p)v_t) \tag{4}$$

$$v_t, s_t|\sigma \sim \text{Exp}(v_t|\sigma)N^+(s_t|0, 1) \tag{5}$$

$$\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}, \mathbf{W}_t \sim \text{N}(\mathbf{G}_t\boldsymbol{\theta}_{t-1}, \mathbf{W}_t). \tag{6}$$

This hierarchical form is leveraged for the posterior inference implemented within the **exdqlm** package.

2.1. Prior specification

Under a Bayesian formulation and given p_0, we specify priors for the initial state vector $\boldsymbol{\theta}_0$, scale parameter σ and skewness parameter γ. For $\boldsymbol{\theta}_0$, we assume a q-variate conjugate normal prior $\boldsymbol{\theta}_0 \sim \text{N}(\mathbf{m}_0, \mathbf{C}_0)$. For σ, we assume a conjugate inverse-gamma prior denoted as $\text{IG}(a_\sigma, b_\sigma)$. In many cases a strong prior on σ is necessary to guarantee fast posterior convergence and estimation. This is illustrated in the examples of Section 4. The parameter γ has bounded support over the interval (L, U) where L is the negative root of $g(\gamma) = 1 - p_0$ and U is the positive root of $g(\gamma) = p_0$ (Yan and Kottas 2017). In contrast to the flat prior suggested by Yan and Kottas (2017), we find that using a proper prior on the skewness

Input list	`model`				`PriorSigma`		`PriorGamma`		
Symbol	$\mathbf{F}_{1:T}$	$\mathbf{G}_{1:T}$	$\mathbf{m}_0$	$\mathbf{C}_0$	a_σ	b_σ	m_γ	s_γ	ν_γ
List component	`FF`	`GG`	`m0`	`C0`	`a_sig`	`b_sig`	`m_gam`	`s_gam`	`df_gam`
Default	-	-	-	-	2.1	1.1	0	1	1

Table 1: Translation from mathematical symbols to parameters used **exdqlm** functions, as well as default values where applicable.

parameter promotes reliable posterior inference. Thus, we place a Student-t distribution truncated to the interval (L, U) as the prior for γ, i.e. $\gamma \sim \mathrm{t}_{(L,U)}(m_\gamma, s_\gamma)$ with ν_γ degrees of freedom. Table 1 shows the inputs that define the structure of the dynamic model; $\mathbf{F}_t$ and $\mathbf{G}_t$ for $t = 1, \ldots, T$. The prior parameters used in the **exdqlm** functions, their corresponding mathematical symbols, and default values are also show in Table 1.

2.2. Posterior estimation

The hierarchical representation of the exDQLM facilitates posterior simulation using Markov Chain Monte Carlo (MCMC) algorithms. This is implemented in the function `exdqlmMCMC`. Gibbs sampling steps are used to update the sets of latent variables s_t and v_t for $t = 1, \ldots, T$. The dynamic regression coefficients are sampled using a forward filtering backwards sampling algorithm (Carter and Kohn 1994; Frühwirth-Schnatter 1994). Finally, parameters γ and σ are updated jointly with a random-walk Metropolis-Hastings (MH) step on the rescaled logit- and log-scale, respectively. The covariance matrix used to sample the parameters (Σ_{MH}) can be specified by the user and/or automatically computed from the burned sample. An example of this functionality can be found in Section 4.1. We perform a total of $N_{BURN} + N_{MCMC}$ MCMC iterations, discarding the first N_{BURN} resulting in N_{MCMC} posterior samples. The output of the `exdqlmMCMC` function includes the posterior samples of all parameters $(\boldsymbol{\theta}_{1:T}, v_{1:T}, s_{1:T}, \gamma, \sigma)$, a list containing the parameters of the filtered and smoothed distributions of the state vector $\boldsymbol{\theta}_{1:T}$, the covariance matrix Σ_{MH} and the acceptance rate of the MH step.

For purposes of model selection or large time series data sets, posterior sampling with the MCMC algorithm can be computationally intractable. To this end, we include in the package an importance sampling variational Bayes (ISVB) routine, `exdqlmISVB`, for fast and accurate approximate posterior estimation. The variational Bayes (VB) framework allows us to approximate the posterior distribution with partitioned variational distributions (Ostwald, Kirilina, Starke, and Blankenburg 2014; Beal 2003). For the exDQLM, these are recognizable closed-form distributions with the exception of the joint variational distribution of σ and γ. The variational distribution of σ and γ is approximated at each iteration using importance sampling (IS) with N_{IS} particles. The parameters of the remaining distributions are computed directly at each iteration, with the dynamic regression coefficients updated using a forward filtering backwards smoothing algorithm. After the algorithm reaches convergence with tolerance of ϵ_{tol}, N_{SAMP} samples are drawn from the variational distributions, resulting in an approximate sample of the posterior distribution. The output of the `exdqlmISVB` function includes the approximate posterior samples as well as lists containing the parameters of the variational distributions for all parameters $(\boldsymbol{\theta}_{1:T}, v_{1:T}, s_{1:T}, \gamma, \sigma)$. For γ and σ the list also contains the IS weights and samples.

exdqlm function	`exdqlmMCMC`			`exdqlmISVB`		
Symbol	Σ_{MH}	N_{BURN}	N_{MCMC}	ϵ_{tol}	N_{IS}	N_{SAMP}
Input	`Sig.mh`	`n.burn`	`n.mcmc`	`tol`	`n.IS`	`n.samp`
Default	$0.05\mathbf{I}_2$	2000	1500	0.1	500	200

Table 2: Translation from mathematical symbols to parameters used in functions `exdqlmMCMC` and `exdqlmISVB`, as well as default values. Here $\mathbf{I}_2$ denotes the identity matrix of dimension 2.

Symbol	$\tilde{t}$	k	$\mathbf{F}_{(\tilde{t}+1):(\tilde{t}+k)}$	$\mathbf{G}_{(\tilde{t}+1):(\tilde{t}+k)}$
`exdqlmForecast` input	`start.t`	`k`	`fFF`	`fGG`

Table 3: Translation from mathematical symbols to additional parameters used in function `exdqlmForecast`.

Table 2 shows the input parameters used in the functions discussed in this section, their mathematical symbols and default values.

2.3. Specification of the evolution covariance via discount factors

Both estimation algorithms, `exdqlmMCMC` and `exdqlmISVB`, utilize discount factors to specify the time-evolving covariance matrices $\mathbf{W}_t$ introduced in Equation (2) (West, Harrison, and Migon 1985). More precisely, we define the evolution variance matrix $\mathbf{W}_t = \frac{1-\delta}{\delta}\mathbf{G}_t\mathbf{C}_{t-1}\mathbf{G}_t'$ where $\mathbf{C}_{t-1}$ denotes the posterior variance of the state vector $\boldsymbol{\theta}_{t-1}$ at time $t-1$ and δ is a discount factor such that $0 < \delta \leq 1$. The discount factor δ can be specified by the user via function parameter `df`. Selection of discount factors is typically done by optimizing some model checking criterion. We suggest using the Kullback-Leibler (KL) divergence of the one-step-ahead predictive distribution functions discussed in the following Section 3. A brief example of this method for discount factor selection can be found in Section 4.2.

Discounting by component of the state-space model is also a functionality of our estimation algorithms. Suppose the state vector is comprised of h components $\boldsymbol{\theta}_{it}$ of dimension q_i for $i = 1, \ldots, h$ such that $\boldsymbol{\theta}_t' = (\boldsymbol{\theta}_{1t}', \ldots, \boldsymbol{\theta}_{ht})'$ and $\sum_{i=1}^{h} q_i = q$. If $\mathbf{W}_{it}$ denotes the i^{th} component of the evolution covariance matrix such that $\mathbf{W}_t = \text{blockdiag}\{\mathbf{W}_{1t}, \ldots, \mathbf{W}_{ht}\}$ and δ_i denotes a discount factor for the i^{th} component, we can define the evolution variance matrix by component as $\mathbf{W}_{it} = \frac{1-\delta_i}{\delta_i}\mathbf{G}_{it}\mathbf{C}_{i,t-1}\mathbf{G}_{it}'$. Here $\mathbf{G}_{it}$ and $\mathbf{C}_{i,t-1}$ denote the i^{th} components of $\mathbf{G}_t$ and $\mathbf{C}_{t-1}$, respectively. The function parameters `df` and `dim.df` allow users to specify a set of component discount factors $(\delta_1, \ldots, \delta_h)$ and their dimensions $(q_1, \ldots, q_h)$, respectively, within the state-space model. Examples of this feature can be found in Section 4. For a full review of discount factors including discounting by component, see West *et al.* (1985).

2.4. k-step-ahead forecast

For an arbitrary time $\tilde{t}$, the k-step-ahead future marginal distribution of the quantile is

$$\mathbf{F}_{\tilde{t}+k}'\boldsymbol{\theta}_{\tilde{t}+k}|y_1, \ldots, y_{\tilde{t}} \sim \text{N}(\mathbf{F}_{\tilde{t}+k}'a_{\tilde{t}}(k), \mathbf{F}_{\tilde{t}+k}'\mathbf{R}_{\tilde{t}}(k)\mathbf{F}_{\tilde{t}+k}) \tag{7}$$

where $\mathbf{a}_{\tilde{t}}(k) = \mathbf{G}_{\tilde{t}+k}\mathbf{a}_{\tilde{t}}(k-1)$, $\mathbf{R}_{\tilde{t}}(k) = \mathbf{G}_{\tilde{t}+k}\mathbf{R}_{\tilde{t}}(k-1)\mathbf{G}_{\tilde{t}+k}' + \mathbf{W}_{\tilde{t}+k}$, $\mathbf{a}_{\tilde{t}}(0) = \mathbf{m}_{\tilde{t}}$, and $\mathbf{R}_{\tilde{t}}(0) = \mathbf{C}_{\tilde{t}}$, with $\mathbf{m}_{\tilde{t}}$ and $\mathbf{C}_{\tilde{t}}$ denoting the filtered mean and covariance of $\boldsymbol{\theta}_{\tilde{t}}$, respectively. This

Symbol	X_t	$\tilde{\lambda}$	$(\delta_\zeta, \delta_\psi)$	$\mathbf{m}_0^{tf}$	$\mathbf{C}_0^{tf}$
`transfn_exdqlmISVB` input	`X`	`lam`	`tf.df`	`tf.m0`	`tf.C0`

Table 4: Translation from mathematical symbols to additional parameters used in function `transfn_exdqlmISVB`.

distribution is implemented in the function `exdqlmForecast` which is compatible with the outputs from both the MCMC and ISVB algorithms. Table 3 shows the input parameters used in the function, as well as their mathematical symbols seen in this section. The output of `exdqlmForecast` includes the parameters of the forecasted distribution as well as $\mathbf{a}_{\tilde{t}}(k)$ and $\mathbf{R}_{\tilde{t}}(k)$ for the k steps.

2.5. Transfer function model

Quantifying the relationship between an input and a quantile of a response is a non-trivial task, particularly when the relationship is nonlinear. In mean-centric dynamic modeling, transfer functions are a straight-forward method to incorporate variables which measure the combined effect of current and past regression effects (West *et al.* 1985). An extension of the transfer function approach to the dynamic quantile modeling setting is implemented within the **exdqlm** package, particularly for the ISVB algorithm.

For $t = 1, \ldots, T$ and univariate input, X_t, a transfer function exDQLM with exponential decay is defined by

$$y_t|\boldsymbol{\theta}_t, \gamma, \sigma \sim \text{exAL}_{p_0}(\mathbf{F}_t'\boldsymbol{\theta}_t + \zeta_t, \sigma, \gamma) \tag{8}$$
$$\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}, \mathbf{W}_t \sim \text{N}(\mathbf{G}_t\boldsymbol{\theta}_{t-1}, \mathbf{W}_t) \tag{9}$$
$$\zeta_t|\zeta_{t-1}, \psi_{t-1}, \omega_t \sim \text{N}(\lambda\zeta_{t-1} + X_t\psi_{t-1}, \omega_t) \tag{10}$$
$$\psi_t|\psi_{t-1}, \nu_t \sim \text{N}(\psi_{t-1}, \nu_t). \tag{11}$$

Here ζ_t captures the effect of the current and past regression effects on the quantile at time t. Alternatively, the parameter ψ_t determines the instantaneous effect X_t has on the quantile. The parameter $\lambda \in [0, 1]$ represents the memory of the regression effect up to time t. This effect decays at an exponential rate, reducing by a factor of λ at every time step. More explicitly, the effect of X_t on the quantile at time $t + k$ is $\lambda^k\psi_{t-1}X_t$. Thus, the series $k_t \geq \frac{\log(\epsilon)-\log(|\psi_{t-1}X_t|)}{\log(\lambda)}$ represents a lower bound for the number of time steps until the effect of X_t is less than or equal to a fixed ϵ. The median value of k_t corresponding to $\epsilon = 1\text{e}{-3}$ is included as an output of our transfer function routine.

Conditional on a fixed λ, say $\tilde{\lambda}$, we augment the ISVB algorithm to incorporate the transfer function structure by simply replacing $\mathbf{F}_t, \boldsymbol{\theta}_t, \mathbf{G}_t$, and $\mathbf{W}_t$ in Equations (1)-(2) with $\tilde{\mathbf{F}}_t' = (\mathbf{F}_t', 1, 0)$, $\tilde{\boldsymbol{\theta}}_t' = (\boldsymbol{\theta}_t', \zeta_t, \psi_t)$, $\tilde{\mathbf{G}}_t = \text{blockdiag}\left\{\mathbf{G}_t, \begin{pmatrix} \tilde{\lambda} & X_t \\ 0 & 1 \end{pmatrix}\right\}$, and $\tilde{\mathbf{W}}_t = \text{blockdiag}\left\{\mathbf{W}_t, \begin{pmatrix} \omega_t & 0 \\ 0 & \nu_t \end{pmatrix}\right\}$, respectively. This augmentation is implemented in the function `transfn_exdqlmISVB`, an extension of the routine found in `exdqlmISVB`.

Similar to the specification of $\mathbf{W_t}$ discussed in Section 2.3, discount factors $(\delta_\zeta, \delta_\psi)$ are utilized to specify $\begin{pmatrix} \omega_t & 0 \\ 0 & \nu_t \end{pmatrix}$. A bivariate normal conjugate prior on $(\zeta_0, \psi_0)' \sim N(\mathbf{m}_0^{tf}, \mathbf{C}_0^{tf})$ completes the transfer function model extension. Table 4 shows the additional parameters used in `transfn_exdqlmISVB`, as well as their mathematical symbols discussed in this section.

2.6. Special cases

As mentioned previously, non-time-varying quantile regression is a special case of our methodology. More specifically, when discount factors are set to 1, the evolution covariance $\mathbf{W}_t$ reduces to $\mathbf{0}$ resulting in non-time-varying parameters (West *et al.* 1985). Thus, the methodology presented in Yan and Kottas (2017) which utilizes the exAL in a static model can also be implemented using our package with parameter `df = 1`.

The next special case of our exDQLM arises from the fact that the exAL is an extension of the AL (Yan and Kottas 2017). When $\gamma = 0$, the observational error ϵ_t in Equation (1) reduces such that $\epsilon_t \sim \text{exAL}_{p_0}(\epsilon_t|0, \sigma, \gamma) = \text{AL}_{p_0}(\epsilon_t|0, \sigma)$. This special case with observational errors distributed independently from an AL is the dynamic quantile linear model (DQLM) presented in Gonçalves, Migon, and Bastos (2017). The DQLM can be implemented with our package by fixing $\gamma = 0$ using the parameter `dqlm.ind = TRUE`.

The final special case is the intersection of the previous two. A static quantile regression model with observational errors distributed according to an AL is the well-known Bayesian parametric method for quantile regression first presented in Yu and Moyeed (2001). This methodology, utilized to create the Bayesian package for quantile regression **bayesQR**, can also be implemented with our package using parameters `df = 1` and `dqlm.ind = TRUE`.

For further details on the methodology outlined in this section, see Barata *et al.* (in press).

3. Model Diagnostics

To evaluate the quantile inference and predictive performance of the exDQLM, several quantitative and visual model diagnostics are included in the package.

The one-step-ahead predictive distribution function introduced by Rosenblatt (1952) is a useful diagnostic for models in which the marginal forecast distributions are unrecognizable or difficult to compute (Huerta, Jiang, and Tanner 2003; Prado, Molina, and Huerta 2006). For the exDQLM, this distribution is defined as

$$u_t = \Pr(Y_t \leq y_t | y_{1:t-1}, v_{1:T}, s_{1:T}, \sigma, \gamma), \tag{12}$$

with $y_{1:t-1} = y_1, \ldots, y_{t-1}$, $v_{1:T} = (v_1, \ldots, v_T)$ and $s_{1:T} = (s_1, \ldots, s_T)$. Here u_t defines an independent sequence which is uniformly distributed on the interval $(0, 1)$ (Rosenblatt 1952). Conditional on $\{v_{1:T}, s_{1:T}, \sigma, \gamma\}$, the forecast distribution of y_t within the exDQLM is normally distributed. Therefore, conditional on MAP estimates $\{\hat{v}_{1:T}, \hat{s}_{1:T}, \hat{\sigma}, \hat{\gamma}\}$ we estimate the sequence as

$$\hat{u}_t = \Phi(y_t | y_{1:t-1}, \hat{v}_{1:T}, \hat{s}_{1:T}, \hat{\sigma}, \hat{\gamma}) \tag{13}$$

where Φ denotes the normal CDF.

Using this estimated sequence $\{\hat{u}_t\}$, we can assess the model performance in several ways. First, we visually examine the correlation of the estimated sequence via an ACF plot. Second, by transforming the values with a standard normal inverse CDF, we compare their distributional shape to that of a standard normal with a normal QQ-plot. To quantify the divergence of this transformed sequence from normality we compute the KL divergence (Kullback and Leibler 1951), $\text{KL}(h, \phi) = \int_{-\infty}^{\infty} h(x) \log \frac{h(x)}{\phi(x)} \text{d}x$, where h denotes the numerically approximated density of our transformed sample and ϕ is the standard normal density. A smaller

KL divergence suggests a superior model fit. Lastly, the MAP standardized forecast errors,

$$\frac{y_t - \mathsf{E}[y_t|y_{1:t-1}, \hat{v}_{1:T}, \hat{s}_{1:T}, \hat{\sigma}, \hat{\gamma}]}{\sqrt{\text{Var}(y_t|y_{1:t-1}, \hat{v}_{1:T}, \hat{s}_{1:T}, \hat{\sigma}, \hat{\gamma})}} \tag{14}$$

can also be used as an additional graphical aid in examining the distribution at specific time points. These MAP standardized forecast errors are included in the output of the `exdqlmMCMC` and `exdqlmISVB` functions.

For a final diagnostic, we include Gelfand and Ghosh (1998) posterior predictive loss criterion (pplc) with check loss function $\rho_{p_0}(x) = x[p_0 - I(x < 0)]$. That is,

$$\text{pplc} = \sum_t \mathsf{E}[\rho_{p_0}(y_t^{obs} - y_t^{rep})|y_{1:T}], \tag{15}$$

where the expectation is with respect to the posterior predictive distribution of y_t, $p(y_t^{rep}|y_{1:T})$. This distribution is used to generate a replicate dataset and is the expected value of the observation equation (1) with respect to the posterior distributions of the parameters. Again, a smaller pplc value suggest a superior model fit. Samples from the posterior replicate distributions are included in the output of the `exdqlmMCMC` and `exdqlmISVB` functions.

The function `exdqlmChecks` implements the model checking criteria discussed in this section. When an output from either `exdqlmMCMC` and `exdqlmISVB` is passed to `exdqlmChecks`, the function output includes: the estimated sequence $\{\hat{u}_t\}$, the KL divergence, pplc, the ordered pairs of the QQ-plot, and autocorrelations by lag. By default, the function also produces the QQ-plot, ACF plot and MAP standard forecast error plot. Examples of the utility of `exdqlmChecks` can be found in the following Section 4.

4. Examples

The following examples are implemented on a personal computer with a 2.5 GHz Intel Core i5 processor. We begin by loading the **exdqlm** package used for all examples.

```
R> library(exdqlm)
```

4.1. Lake Huron

In this example, we consider the Lake Huron time series from the package **data sets**. The data are annual measurements of the level (ft) of Lake Huron from 1875 to 1972, shown in Figure 2. With this example we show how to: build a basic state-space structure, specify a prior on γ, run the MCMC algorithm, plot the estimated quantiles and forecast distributions.

To estimate the dynamic distribution of the data, we consider three quantiles, $p_0 = 0.95, 0.50,$ and 0.05. We begin by creating the state-space structure and prior used to estimate the quantiles (i.e. $\mathbf{F}_t, \mathbf{G}_t, \mathbf{m}_0$, and $\mathbf{C}_0$ discussed in Section 2). We choose to model the quantiles with a second order polynomial trend, which we construct with the `polytrendMod` function.

```
R> model = polytrendMod(order = 2, m0 = c(mean(LakeHuron),0), C0 = 10*diag(2))
R> model
```

```
$FF
[1] 1 0

$GG
     [,1] [,2]
[1,]    1    1
[2,]    0    1

$m0
[1] 579.0041   0.0000

$C0
     [,1] [,2]
[1,]   10    0
[2,]    0   10
```

To allow variability in the dynamic quantile, we select a single discount factor of 0.9 to define the evolution covariance matrix $\mathbf{W}_t$. Discount factors can also be optimized, an example of which we will show in Section 4.2. We use the parameter `fix.sigma = TRUE` to place a point-mass prior on σ to facilitate convergence, with point-mass at 0.07 for $p_0 = 0.95, 0.05$ and 0.4 for $p_0 = 0.5$. A discussion on how to choose the location of the point-masses can be found in Section 4.2. Further, we place truncated Student-t priors on the skewness parameters via `PriorGamma` as discussed in Section 2.1. Lastly, we limit the number of iterations with `n.burn = 700` and `n.mcmc = 300`. The R output is shown for only the last quantile using the input `verbose = TRUE`.

```
R> M95 = exdqlmMCMC(y = LakeHuron, p0 = 0.95, model = model,
+                   df = 0.9, dim.df = 2,
+                   fix.sigma = TRUE, sig.init = 0.07,
+                   PriorGamma = list(m_gam = -1, s_gam = 0.1, df_gam = 1),
+                   n.burn = 700, n.mcmc = 300, verbose = FALSE)
R> M50 = exdqlmMCMC(y = LakeHuron, p0 = 0.50, model = model,
+                    df = 0.9, dim.df = 2,
+                    fix.sigma = TRUE, sig.init = 0.4,
+                    PriorGamma = list(m_gam = 0, s_gam = 0.1, df_gam = 1),
+                    n.burn = 700, n.mcmc = 300, verbose = FALSE)
R> M5 = exdqlmMCMC(y = LakeHuron, p0 = 0.05, model = model,
+                   df = 0.9, dim.df = 2,
+                   fix.sigma = TRUE, sig.init = 0.07,
+                   PriorGamma = list(m_gam = 1, s_gam = 0.1, df_gam = 1),
+                   n.burn = 700, n.mcmc = 300, verbose = TRUE)

[1] "running isvb algorithm to initialize mcmc"
[1] "ISVB iteration 5: 2021-06-14 14:02:30"
[1] "ISVB iteration 10: 2021-06-14 14:02:31"
[1] "ISVB iteration 15: 2021-06-14 14:02:33"
[1] "ISVB converged: 15 iterations, 4.584 seconds"
```

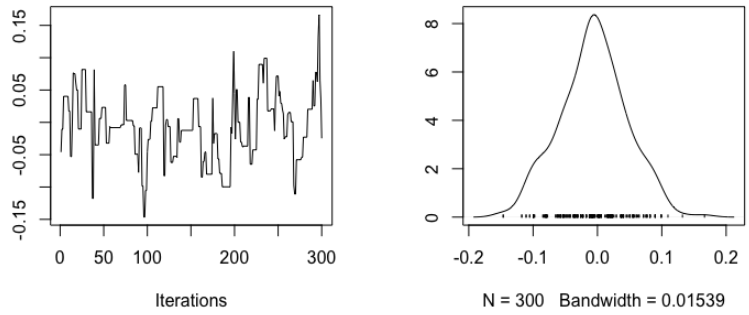

Figure 1: Example 1: Lake Huron. Trace plot (left) and density (right) of the MCMC samples of the skewness parameter γ estimated at the median.

```
[1] "burn-in iteration 500, acceptance rate 0.032: 2021-06-14 14:03:11"
[1] "MCMC iteration 1000, acceptance rate 0.127: 2021-06-14 14:03:52"
[1] "MCMC complete: 1000 iterations, 78.617 seconds"
```

For this time series of length 98 it takes approximately 80 seconds for the MCMC algorithm sample 1000 iterations, as seen in the output above. By default `exdqlmMCMC` runs the ISVB algorithm to initialize the MCMC, which prints progress every 5 iterations. When the ISVB is used for initialization, the covariance matrix used to sample σ and γ in the MH step is automatically calculated from the ISVB samples of σ and γ. This covariance can be specified manually by using the setting `init.from.isvb = FALSE`, and providing the covariance via input `mh_Sig`. Further, using the setting `joint.sample = TRUE`, the covariance matrix can be recalculated from burned samples.

After the ISVB algorithm converges, the `exdqlmMCMC` prints progress after each 500 MCMC iterations, along with the current acceptance rate of the MH step and system time. It is not uncommon for low acceptance rates to facilitate convergence, particularly for extreme quantiles (i.e. 0.05), as is seen in this example. The data are relatively symmetric, therefore we do not expect the skewness parameter γ to be far from zero for $p_0 = 0.5$. We can examine this with the samples from the posterior distribution of γ. All MCMC samples in the output of `exdqlmMCMC` are `mcmc` objects. Thus we can use the package **coda** to examine the trace plot and density of γ, seen in Figure 1.

```
R> library(coda)
R> traceplot(M50$samp.gamma, main = "")
R> densplot(M50$samp.gamma, main = "")
```

Clearly γ is indistinct from zero, thus the added flexibility of the skewness parameter is not necessary for the median. We re-run the model with a point-mass prior on γ at 0 using the settings `gam.init = 0` and `fix.gamma = 0`, equivalent to the setting `dqlm.ind = TRUE`.

```
R> M50 = exdqlmMCMC(y = LakeHuron, p0 = 0.50, model = model,
+                   df = 0.9, dim.df = 2,
```

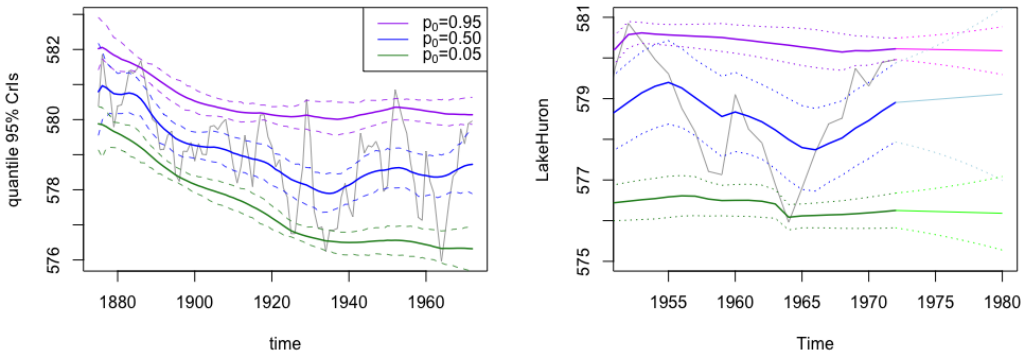

Figure 2: Example 1: Lake Huron. Left: MAP estimates and 95% CrIs of the estimated quantiles, plotted with the data (grey). Right: Forecasted quantile estimates and 95% credible intervals (seen after 1972), along with the filtered quantile estimates and 95% credible intervals (seen from 1952 to 1972).

```
+                  fix.sigma = TRUE, sig.init = 0.4,
+                  gam.init = 0, fix.gamma = TRUE,
+                  n.burn = 700, n.mcmc = 300)

[1] "running isvb algorithm to initialize mcmc"
[1] "ISVB iteration 5: 2021-07-18 12:42:47"
[1] "ISVB iteration 10: 2021-07-18 12:42:48"
[1] "ISVB iteration 15: 2021-07-18 12:42:50"
[1] "ISVB converged: 15 iterations, 4.305 seconds"
[1] "burn-in iteration 500, acceptance rate 0.212: 2021-07-18 12:43:35"
[1] "MCMC iteration 1000, acceptance rate 0.265: 2021-07-18 12:44:18"
[1] "MCMC complete: 1000 iterations, 87.92 seconds"
```

The results are `exdqlm` objects that can be used for analysis and forecasting. First we examine the estimated quantiles with the function `exdqlmPlot`, seen in Figure 2. This function generates a plot of the data with the MAP estimates and the 95% CrIs of the dynamic quantile. Subsequent quantiles (i.e. $p_0 = 0.50, 0.05$ in this example) can be added to the plot using the setting `add = TRUE`.

```
R> exdqlmPlot(y = LakeHuron, M95)
R> exdqlmPlot(y = LakeHuron, M50, add = TRUE, col = "blue")
R> exdqlmPlot(y = LakeHuron, M5, add = TRUE, col = "forest green")
R> legend("topright", lty = 1, col = c("purple","blue","forest green"),
+        legend = c(expression('p'[0]*'=0.95'),expression('p'[0]*'=0.50'),
+                   expression('p'[0]*'=0.05')))
```

Next, we forecast the $k = 8$ step-ahead distributions starting in 1972. To do this, we first create the observational vector (`fFF`) and evolution matrix (`fGG`) that will be used in the

forecast updates. In this case both are non-time-varying and identical to those used to estimate the quantile.

```
R> fFF = model$FF
R> fGG =  model$GG
```

We plot the time series data in a narrower range for closer examination and add the forecast estimates using `exdqlmForecast` with setting `add = TRUE`. Notice we start the forecast at the last time index of the data, i.e. `start.t = length(LakeHuron)`.

```
R> plot.ts(LakeHuron,  xlim = c(1952,1980), ylim = c(575,581),
+        col = "dark grey")
R> exdqlmForecast(y = LakeHuron, start.t = length(LakeHuron), k = 8, M95,
+                fFF = fFF, fGG = fGG, plot = TRUE, add = TRUE)
R> exdqlmForecast(y = LakeHuron, start.t = length(LakeHuron), k = 8, M50,
+                fFF = fFF, fGG = fGG, plot = TRUE, add = TRUE,
+                cols = c("blue","light blue"))
R> exdqlmForecast(y = LakeHuron, start.t = length(LakeHuron), k = 8, M5,
+                fFF = fFF, fGG = fGG, plot = TRUE, add = TRUE,
+                cols = c("forest green","green"))
```

This generates a plot of the data with the forecasted quantile estimates and 95% credible intervals (seen after 1972), along with the filtered quantile estimates and 95% credible intervals (before 1972), for reference. Results can be seen in Figure 2. The percentage in the CrIs can be modified with the parameter `cr.percent`.

4.2. Sunspots

For our next example, we will use the yearly Sunspot time series from the package **datasets**. The data are yearly counts for sunspots from 1700 to 1988, shown in Figure 3. With this example we will show how to: use the **dlm** package to create the state-space model, combine blocks of a state-space model, apply the ISVB algorithm, choose a prior for σ to facilitate convergence, perform visual model diagnostics to compare the exDQLM with the DQLM, and use those diagnostics for model selection.

The solar cycle impacts astronauts in space as well as lives and technology on earth. Of particular interest to scientists is the solar max, or period in which the sun is most active Fox (2012), therefore we will explore the 0.85 quantile in this example. Again, we begin by building the state-space structure used to estimate the quantile. Here we choose to model the quantile with a first order polynomial trend component as well as a Fourier form seasonal component that incorporates a fundamental period every 11 observations (years), as well as some of its corresponding harmonics. This results in a dynamic model with a total of 9 state parameters, 1 for the first order polynomial component and 8 for the seasonal component. The **exdqlm** functions are compatible with time-invariant `dlm` objects from the **dlm** package. For example, we create the first order trend component with the function `dlmModPoly` from the **dlm** package.

```
R> library(dlm)
R> dlm.trend.comp = dlmModPoly(1, m0 = mean(sunspot.year), C0 = 10)
```

Next, we construct the seasonal component with our `seasMod` function. The duration of sunspot cycles typically follow a period of 11 years, of which we include the first 4 harmonics in the component.

```
R> seas.comp = seasMod(p = 11, h = 1:4, C0 = 10*diag(8))
```

To combine the two components into a single state-space structure, we use our function `combineMods`.

```
R> model = combineMods(dlm.trend.comp,seas.comp)
```

```
Warning message:
In combineMods(dlm.trend.comp, seas.comp) :
  m1 converted from a dlm object using 'dlmMod(m1)'
```

This produces a warning that the first input `dlm.trend.comp` was converted from a `dlm` object using our function `dlmMod`. This function can also be used directly on the `dlm` object to eliminate the warning. The resulting evolution matrix of the combined models is printed for illustration.

```
R> trend.comp = dlmMod(dlm.trend.comp)
R> model = combineMods(trend.comp,seas.comp)
R> model$GG
```

```
     [,1]    [,2]    [,3]    [,4]   [,5]    [,6]    [,7]    [,8]    [,9]
 [1,]    1  0.0000  0.0000  0.0000 0.0000  0.0000  0.0000  0.0000  0.0000
 [2,]    0  0.8413  0.5406  0.0000 0.0000  0.0000  0.0000  0.0000  0.0000
 [3,]    0 -0.5406  0.8413  0.0000 0.0000  0.0000  0.0000  0.0000  0.0000
 [4,]    0  0.0000  0.0000  0.4154 0.9096  0.0000  0.0000  0.0000  0.0000
 [5,]    0  0.0000  0.0000 -0.9096 0.4154  0.0000  0.0000  0.0000  0.0000
 [6,]    0  0.0000  0.0000  0.0000 0.0000 -0.1423  0.9898  0.0000  0.0000
 [7,]    0  0.0000  0.0000  0.0000 0.0000 -0.9898 -0.1423  0.0000  0.0000
 [8,]    0  0.0000  0.0000  0.0000 0.0000  0.0000  0.0000 -0.6549  0.7557
 [9,]    0  0.0000  0.0000  0.0000 0.0000  0.0000  0.0000 -0.7557 -0.6549
```

We will apply discount factors by component, i.e. a discount factor of 0.9 for the trend component of dimension 1 and a discount factor of 0.85 for the seasonal component of dimension 8. This discounting strategy can be applied with parameters `df = c(0.9,0.85)` and `dim.df = c(1,8)`.

Next we illustrate how to fix σ at a reasonable value. To do this, we investigate the scale of the data using the DQLM, a special case of the exDQLM described in Section 2.6. We call the function `exdqlmISVB` to apply the ISVB algorithm with `dqlm.ind = TRUE` and `fix.sigma = FALSE` to obtain estimates of the scale parameter σ. For this dataset of length 298 the ISVB algorithm for the DQLM takes 29 iterations and approximately 25 seconds to converge, seen in the output below. This output is suppressed for the remainder of this section using input `verbose = FALSE`.

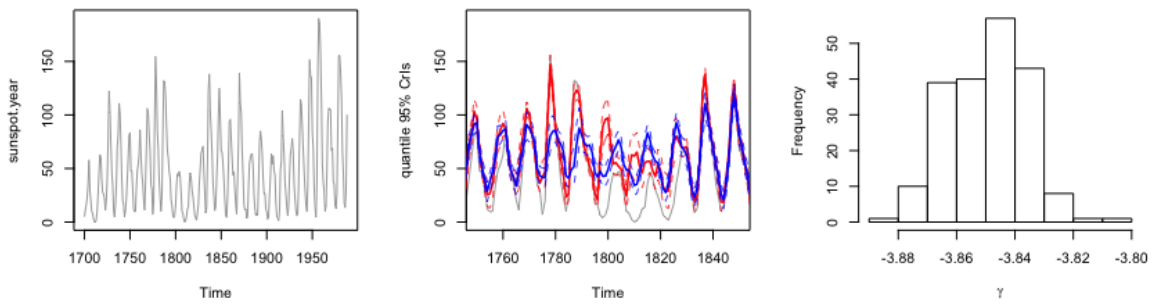

Figure 3: Example 2: Sunspots. Left: The sunspot time series from 1700 to 1988. Center: MAP estimates and 95% CrIs of the estimated quantiles from the DQLM (red) and exDQLM (blue), plotted with the data (grey) from 1750 to 1850. Right: Histogram of samples from the approximated posterior distribution of γ.

```
R> M1 = exdqlmISVB(y = sunspot.year, p0 = 0.85, model = model,
+                  df = c(0.9,0.85), dim.df = c(1,8),
+                  dqlm.ind = TRUE, fix.sigma = FALSE)

[1] "ISVB iteration 5: 2021-07-18 13:01:04"
[1] "ISVB iteration 10: 2021-07-18 13:01:08"
[1] "ISVB iteration 15: 2021-07-18 13:01:12"
[1] "ISVB iteration 20: 2021-07-18 13:01:16"
[1] "ISVB iteration 25: 2021-07-18 13:01:21"
[1] "ISVB converged: 29 iterations, 24.772 seconds"
```

The σ estimates from the DQLM give a general idea of the scale of the data and are a good starting point for fixing σ (or setting a strong prior) in the exDQLM. For example, the MAP estimate of σ from the DQLM is approximately 4.

```
R> summary(M1$samp.sigma)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  3.470   3.806   3.935   3.934   4.054   4.513
```

Reducing this value, we place a point-mass prior on σ at 2. By default, `exdqlmISVB` places a point-mass prior on σ at the value of `sig.init`. We re-run the algorithm with `dqlm.ind = TRUE` to estimate the DQLM, as well as with the default settings to estimate the exDQLM.

```
R> M1 = exdqlmISVB(y = sunspot.year, p0 = 0.85, model = model,
+                  df = c(0.9,0.85), dim.df = c(1,8),
+                  dqlm.ind = TRUE, sig.init = 2,
+                  verbose = FALSE)
R> M2 = exdqlmISVB(y = sunspot.year, p0 = 0.85, model = model,
+                  df = c(0.9,0.85), dim.df = c(1,8),
+                  sig.init = 2, verbose = FALSE)
```

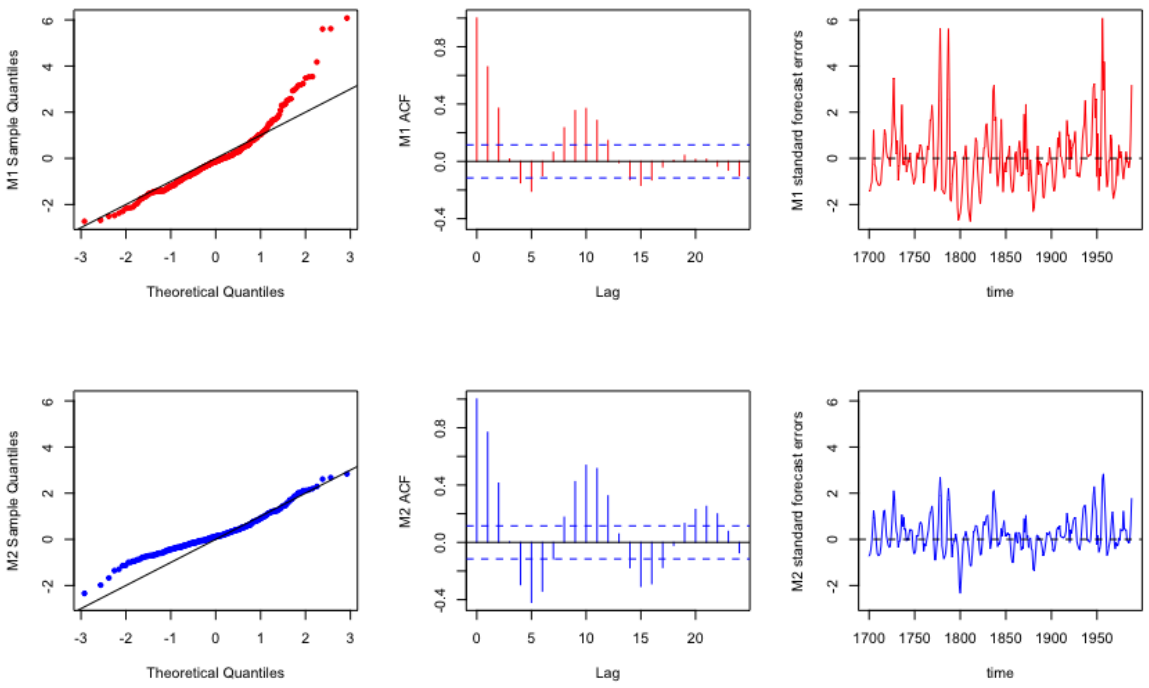

Figure 4: Example 2: Sunspots. The QQ-plots (left column), ACF plots (center column) and standard forecast errors (right column) from the DQLM (red) and exDQLM (blue).

We can visualize the differences between the two resulting dynamic quantiles using the function `exdqlmPlot`, seen in Figure 3. For clarity we limit the figure to years 1750 to 1850.

```
R> plot(sunspot.year, xlim = c(1750,1850), col = "dark grey",
+       ylab = "quantile 95% CrIs")
R> exdqlmPlot(y = sunspot.year, M1, add = TRUE, col = "red")
R> exdqlmPlot(y = sunspot.year, M2, add = TRUE, col = "blue")
```

To examine whether the added flexibility of the exDQLM is significant, we can also visualize the approximate posterior distribution of the skewness parameter γ by plotting the samples from the corresponding variational distribution, also seen in Figure 3.

```
R> hist(M2$samp.gamma,xlab=expression(gamma),main="")
```

The approximate distribution is clearly distinct from zero, thus we expect a better model fit from the exDQLM.

To further examine the two models, we use the function `exdqlmChecks` to plot the model diagnostics discussed in Section 3. The results are seen in Figure 4. It is clear the QQ-plot and the scale of the standard forecast errors favor the exDQLM.

```
R> exdqlmChecks(y = sunspot.year, M1, M2, cols = c("red","blue"))
```

Finally, the function `exdqlmChecks` can also be used for model selection. Say we want to check whether the discount factor we used for the seasonal component was optimal with respect to the KL divergence of the one-step-ahead predictive distributions. We can create a list of possible discount factors, quickly run the ISVB for each possible model, and select the set of discount factors which results in the lowest KL divergence. A simple example is seen below, which takes approximately 6 minutes to run.

```
R> possible.dfs = cbind(0.9,seq(0.85,1,0.05))
R> possible.dfs

     [,1] [,2]
[1,]  0.9 0.85
[2,]  0.9 0.90
[3,]  0.9 0.95
[4,]  0.9 1.00

R> KLs <- vector("numeric")
R> ref.samp = rnorm(length(sunspot.year))
R> for(i in 1:nrow(possible.dfs)){
+   temp.M2 = exdqlmISVB(y = sunspot.year, p0 = 0.85, model = model,
+                   df = possible.dfs[i,], dim.df = c(1,8),
+                   sig.init = 2, verbose = FALSE)
+   temp.check = exdqlmChecks(y = sunspot.year, temp.M2,
+                               plot = FALSE, ref = ref.samp)
+   KLs = c(KLs,temp.check$m1.KL)
+ }
R> # optimal dfs based off KL divergence
R> possible.dfs[which.min(KLs),]

[1] 0.90 0.85
```

The results suggest the discount factor we used for the seasonal component is optimal.

4.3. Big Tree water flows

For our final example, we consider the average monthly natural water flow (cubic feet per second) at the Big Tree gauge of the San Lorenzo river in Santa Cruz County, CA from January 1937 through April 2021 (U.S. Geological Survey 2016), for a total of 1012 observations. The dataset is included in our package as `BTflow`. With this example we show how to build a state-space structure manually (particularly for a dynamic quantile regression), apply our transfer function exDQLM, examine individual components of the estimated quantiles, and perform quantitative model diagnostics to compare models.

For this particular dataset, the interest is in low quantiles as a possible indicator of drought, thus we will consider the $p_0 = 0.15$ quantile. To investigate any climatological influence on this low quantile, we include the Niño 3.4 index (Rayner, Parker, Horton, Folland, Alexander, Rowell, Kent, and Kaplan 2003) as in input of our exDQLM. The Niño 3.4 index is a measure

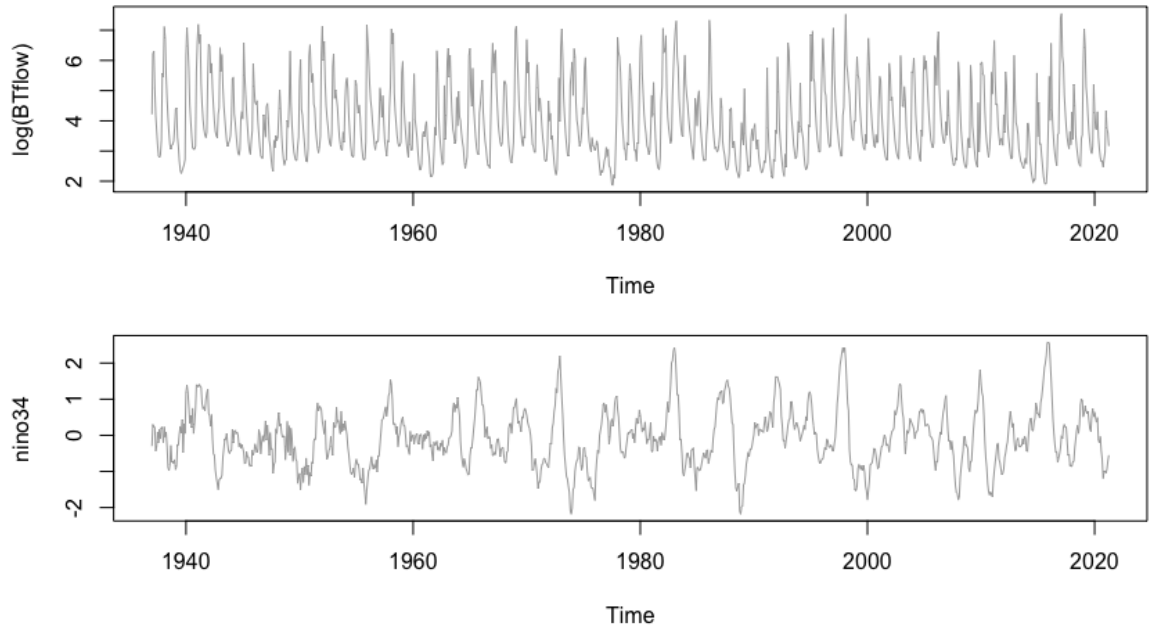

Figure 5: Example 3: Big Tree water flow. Top: Average monthly water flow at the Big Tree gauge of the San Lorenzo river in Santa Cruz County, CA, plotted on the log-scale. Bottom: Monthly values of Niño 3.4. Both time series span January 1937 through April 2021.

of how anomalously warm or cool the average sea surface temperature is in the Niño 3.4 region (5°S-5°N, 170°W-120°W) of the Pacific Ocean. It is one of the most commonly used indices to define El Niño and La Niña events. The monthly Niño 3.4 dataset used in our analysis is also included in the package as `nino34`. We model the relationship between Niño 3.4 and the quantile with two model structures: a dynamic quantile regression and a transfer function as seen in Section 2.5. To avoid modeling problems caused by the scale and extremes in the Big Tree dataset, we fit the following models to the time series on the log scale seen in Figure 5.

In addition to the components that capture the relationship with the Niño 3.4 index, we also include a first order polynomial trend component as well as a Fourier form seasonal component that includes a single periodic component with period 12. The Big Tree water flow time series follows a period of twelve months, of which we include only the first (annual) harmonic.

```
R> trend.comp = polytrendMod(1, m0 = 3, C0 = 0.1)
R> seas.comp = seasMod(p = 12, h = 1, C0 = diag(1,2))
R> model = combineMods(trend.comp, seas.comp)
```

Next, we create the dynamic regression component for our first model. A regression effect is easily written in a state-space format by setting the observational vector $F_t = X_t$ and $G_t = 1$, where X_t denotes the input at time t. Thus we create this regression component for the state-space model manually as follows.

```
R> reg.comp <- NULL
R> reg.comp$m0 = 0
R> reg.comp$C0 = 1
```

```
R> reg.comp$FF = matrix(nino34,nrow = 1)
R> reg.comp$GG = 1
```

Combining this with the trend and seasonal component we have our complete state-space structure for the dynamic quantile regression model.

```
R> model.w.reg = combineMods(model,reg.comp)
```

To allow the Niño 3.4 input components of both models to vary over time, we set the respective discount factors to be 0.95 (this is done via the parameter `df` in the `exdqlmISVB` function and the parameter `tf.df` in the `transfn_exdqlmISVB` function). Alternatively, we set the discount factors to 1 and 0.9 for the trend and seasonal components, respectively. This allows the seasonal amplitudes to evolve over time, while any further variation in the 0.15 quantile from year to year will solely be attributed to the effects of the Niño 3.4 input in our models.

The transfer function model is created within the function `transfn_exdqlmISVB` and follows the structure detailed in Section 2.5. Optimization using the KL divergence suggests the effects of the Niño 3.4 index decay at an optimal transfer function rate of $\lambda = 0.85$.

```
R> possible.lams = seq(0.1,0.9,0.05)
R> KLs <- vector("numeric")
R> ref.samp = rnorm(length(BTflow))
R> for(i in 1:length(possible.lams)){
R>   temp.M2 = transfn_exdqlmISVB(y = log(BTflow), p0 = 0.15, model = model,
+                               df = c(1,0.9), dim.df = c(1,2),
+                               X = nino34, tf.df = c(0.95), lam = possible.lams[i],
+                               tf.m0 = c(0,0), tf.C0 = diag(c(0.1,0.005),2),
+                               sig.init = 0.1, gam.init = - 0.1,
+                               tol = 0.05, verbose = FALSE)
R>   temp.check = exdqlmChecks(y = log(BTflow), temp.M2,
+                            plot = FALSE, ref = ref.samp)
R>   KLs = c(KLs,temp.check$m1.KL)
R> }
R> # optimal lambda based off KL divergence
R> possible.lams[which.min(KLs)]

[1] 0.85
```

With these discount factors and optimal λ, we estimate the dynamic quantile regression model (`M1`) and transfer function model (`M2`).

```
R> M1 = exdqlmISVB(y = log(BTflow), p0 = 0.15, model = model.w.reg,
+                 df = c(1,0.9,0.95), dim.df = c(1,2,1),
+                 sig.init = 0.1, gam.init = - 0.1,
+                 tol = 0.05, verbose = FALSE)
R> M2 = transfn_exdqlmISVB(y = log(BTflow), p0 = 0.15, model = model,
+                         df = c(1,0.9), dim.df = c(1,2),
```

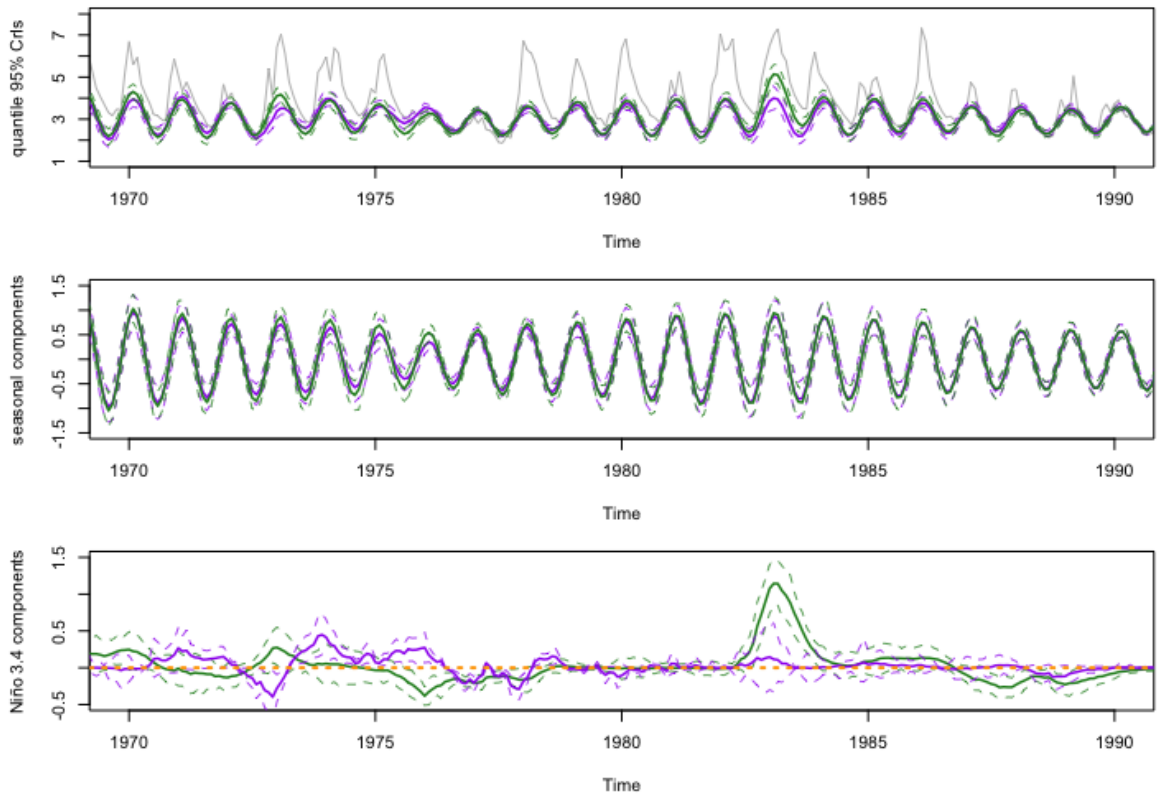

Figure 6: Example 3: Big Tree water flow. Top: MAP estimates and 95% CrIs of the estimated quantiles from the models with the regression component (purple) and transfer function component (green), plotted with the data (grey) from 1970 to 1990. Middle: MAP estimates and 95% CrIs of the annual seasonal components. Bottom: MAP estimates and 95% CrIs of the components which model the quantile association with Niño 3.4. The estimates of the dynamic regression component $X_t\boldsymbol{\theta}_{4,t}$ of M_1 where $\boldsymbol{\theta}_{4,t}$ denotes the fourth element of the state vector, are seen in purple. The estimates of the transfer function component ζ_t of M_2 are in green. The horizontal orange dotted line is at zero for reference.

```
+                         X = nino34, tf.df = c(0.95), lam = 0.85,
+                         tf.m0 = c(0,0), tf.C0 = diag(c(0.1,0.005),2),
+                         sig.init = 0.1, gam.init = - 0.1,
+                         tol = 0.05, verbose = FALSE)
```

To visualize the differences between the two resulting dynamic quantiles we use the function `exdqlmPlot`, results of which are found in Figure 6. For clarity we limit the figure to view 1970 to 1990.

```
R> plot(log(BTflow), col = "grey", ylim = c(1,8), xlim=c(1970,1990),
+       ylab = "quantile 95% CrIs")
R> exdqlmPlot(y = BTflow, M1, add = TRUE)
R> exdqlmPlot(y = BTflow, M2, add = TRUE, col = "forest green")
```

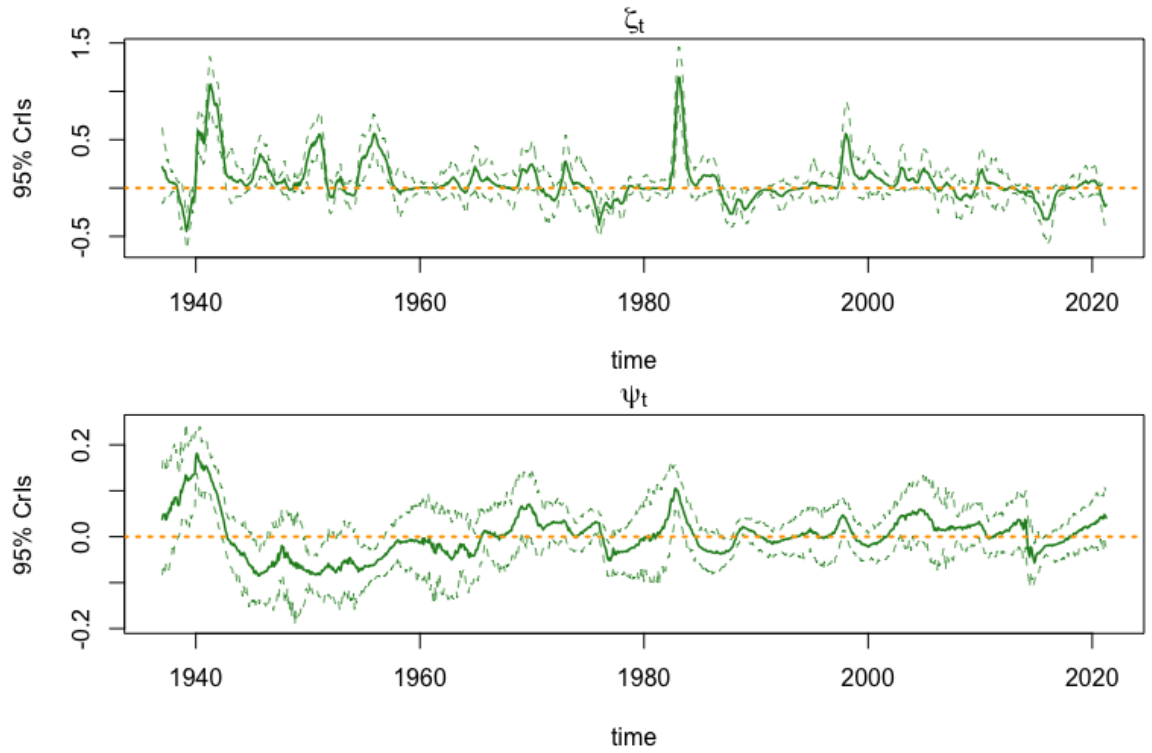

Figure 7: Example 3: Big Tree water flow. MAP estimates and 95% CrIs of parameters ζ_t and ψ_t from the transfer function component. Horizontal dotted orange lines are at zero for reference.

Clearly the resulting quantiles have significant differences. To identify from where these differences arise, we can utilize the function `compPlot` to examine the components of the models more closely. We examine the annual component of the parameter vector first. The annual component is comprised of the second and third elements of the parameter vector $\boldsymbol{\theta}_t$, thus we set the parameter `index = c(2,3)` to visualize this component. The result is seen in Figure 6, again limited to the years 1970 to 1990.

```
R> plot(NA, ylim = c(-1.5,1.5), xlim = c(1970,1990),
+         ylab = "seasonal components")
R> compPlot(y = log(BTflow), M1, index = c(2,3), add = TRUE)
R> compPlot(y = log(BTflow), M2, index = c(2,3), add = TRUE,
+            col = "forest green")
```

Although the seasonal components of the two models exhibit differences in amplitude, we expect more significant differences to appear in the Niño 3.4 component of each model. Again, we use the function `compPlot` to visualize these components seen in Figure 6 from 1970 to 1990.

```
R> plot(NA, ylim = c(-0.5,1.5), xlim = c(1970,1990),
+          ylab = "Niño 3.4 components")
R> compPlot(y = log(BTflow), M1, index = c(4), add = TRUE)
```

```
R> compPlot(y = log(BTflow), M2, index = c(4,5), add = TRUE,
+            col = "forest green")
R> abline(h = 0, col = "orange", lty = 3, lwd = 2)
```

For the transfer function model, `M2`, it is also of interest to look at the estimates of parameters ζ_t and ψ_t directly. Because these parameters augmented the state-space model as discussed in Section 2.5, the function `compPlot` can also be used to look at these estimates directly using the parameter `just.theta = TRUE`. This setting returns the dynamic posterior estimates of a single index of the state vector $\boldsymbol{\theta}_t$ (or $\tilde{\boldsymbol{\theta}}_t$ in the case of the transfer function model). Parameter `index` of `4` and `5` correspond to ζ_t and ψ_t, respectively. Results can be found in Figure 7.

```
R> compPlot(y = log(BTflow), M2, index = 4, col= "forest green",
+             add = FALSE, just.theta = TRUE)
R> abline(h = 0, col = "orange", lty = 3, lwd = 2)
R> title(expression(eta[t]))
R> compPlot(y = log(BTflow), M2, index = 5, col = "forest green",
+             add = FALSE, just.theta = TRUE)
R> abline(h = 0, col = "orange", lty = 3, lwd = 2)
R> title(expression(psi[t]))
```

To further understand these transfer function parameters, we examine the median number of time steps until the effect of Niño 3.4 on the 0.15 quantile is less than or equal to 1e−3, as discussed in Section 2.5. A median value 20.76 suggests the effect of the Niño 3.4 index value at time t last for around 21 months.

```
R> M2$median.kt
```

```
[1] 20.76305
```

This suggests a long-term forecast could be of interest. Thus we forecast the $k = 18$ step ahead distributions for both models starting in October 2019, results of which are seen in Figure 8.

```
R> exdqlmForecast(y = log(BTflow), start.t = length(BTflow)-18,
+                   k = 18, M1)
R> exdqlmForecast(y = log(BTflow), start.t = length(BTflow)-18,
+                   k = 18, M2, add = TRUE, cols = c("forest green","green"))
```

Finally, we use the function `exdqlmChecks` to compute the KL divergence and pplc discussed in Section 3, results of which are seen in Table 5.

```
R> checks.out = exdqlmChecks(y = log(BTflow), M1, M2,
+                              col = c("purple","orange"), plot = FALSE)
```

The KL divergence and pplc both clearly favor the nonlinear relationship captured by the transfer function model.

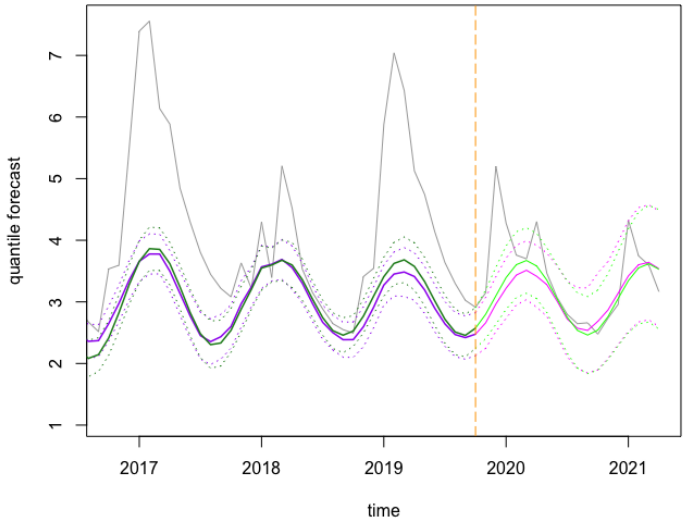

Figure 8: Example 3: Big Tree water flow. 18-step-ahead quantile forecast beginning October 2019 through April 2021 overlaid on the log Big Tree water flow time series. The vertical dot-dashed line is at the beginning of the forecast period, October 2019, for reference. Solid lines indicate mean estimates and dotted lines 95% CrI, with the filtered estimates seen leading up to October 2019 and the forecast estimates beyond October 2019. Estimates from M_1, the quantile regression model, are seen in purple/pink. Estimates from M_2, the transfer function model, are seen in green.

	`M1`		`M2`	
	output	value	output	value
pplc	`m1.pplc`	957.72	`m2.pplc`	897.22
KL	`m1.kl`	1.135	`m2.kl`	0.880

Table 5: Example 3: Big Tree water flow. Model diagnostic output from `exdqlmChecks` and resulting values.

5. Conclusion

Our exDQLM framework provides a robust and versatile tool for dynamic quantile modeling, implemented in the presented **exdqlm** package. Relevant features of the data can easily be incorporated in the evolution of the quantile. The MCMC algorithm provides a method for exact posterior inference. Alternatively, the ISVB algorithm provides a method for fast approximate inference enabling estimation for large data sets, quick examination relevant model features and model selection. Further, our transfer function model provides a straight-forward method for quantifying a nonlinear relationship between a quantile and response. Finally, our forecast and diagnostic routines facilitate further model exploration. These capabilities have been discussed in detail and demonstrated with a set of examples. The **exdqlm** package makes these methods, and more generally, dynamic quantile modeling, accessible to all levels of users.

References

Barata R, Prado R, Sansó B (in press). "Fast inference for time-varying quantiles via flexible dynamic models with application to the characterization of atmospheric rivers." *The annals of applied statistics*.

Beal MJ (2003). *Variational algorithms for approximate Bayesian inference*. Ph.D. thesis, UCL (University College London).

Benoit D, Al-Hamzawi R, Yu K, Van den Poel D (2011). ***bayesQR**: A Bayesian approach to quantile regression*. R package version 2.3, URL https://CRAN.R-project.org/package=bayesQR.

Carter CK, Kohn R (1994). "On Gibbs sampling for state space models." *Biometrika*, **81**(3), 541–553.

Fox K (2012). "Solar Minimum; Solar Maximum." URL https://www.nasa.gov/mission_pages/sunearth/news/solarmin-max.html.

Frühwirth-Schnatter S (1994). "Data augmentation and dynamic linear models." *Journal of time series analysis*, **15**(2), 183–202.

Gelfand AE, Ghosh SK (1998). "Model choice: a minimum posterior predictive loss approach." *Biometrika*, **85**(1), 1–11.

Gonçalves K, Migon HS, Bastos LS (2017). "Dynamic quantile linear model: a Bayesian approach." *arXiv preprint arXiv:1711.00162*.

Huerta G, Jiang W, Tanner MA (2003). "Time series modeling via hierarchical mixtures." *Statistica Sinica*, pp. 1097–1118.

Koenker R (2005). *Quantile regression*. Cambridge University Press, New York.

Koenker R (2021). ***quantreg**: Quantile Regression*. R package version 5.85, URL https://CRAN.R-project.org/package=quantreg.

Kottas A, Krnjajić M (2009). "Bayesian semiparametric modelling in quantile regression." *Scandinavian Journal of Statistics*, **36**(2), 297–319.

Kozumi H, Kobayashi G (2011). "Gibbs sampling methods for Bayesian quantile regression." *Journal of statistical computation and simulation*, **81**(11), 1565–1578.

Kullback S, Leibler RA (1951). "On information and sufficiency." *The annals of mathematical statistics*, **22**(1), 79–86.

Ostwald D, Kirilina E, Starke L, Blankenburg F (2014). "A tutorial on variational Bayes for latent linear stochastic time-series models." *Journal of Mathematical Psychology*, **60**, 1–19.

Ou L, Hunter M, Chow SM, Ji L, Chen M, Hung HJ, Lee J, Li Y, Park J (2021). ***dynr**: Dynamic Models with Regime-Switching*. R package version 0.1.16-2, URL https://CRAN.R-project.org/package=dynr.

Petris G, Gilks W (2018). ***dlm**: Bayesian and Likelihood Analysis of Dynamic Linear Models*. R package version 1.1-5, URL https://CRAN.R-project.org/package=dlm.

Prado R, Molina F, Huerta G (2006). "Multivariate time series modeling and classification via hierarchical VAR mixtures." *Computational Statistics & Data Analysis*, **51**(3), 1445–1462.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/.

Rayner N, Parker DE, Horton E, Folland CK, Alexander LV, Rowell D, Kent EC, Kaplan A (2003). "Global analyses of sea surface temperature, sea ice, and night marine air temperature since the late nineteenth century." *Journal of Geophysical Research: Atmospheres*, **108**(D14).

Reich BJ, Bondell HD, Wang HJ (2009). "Flexible Bayesian quantile regression for independent and clustered data." *Biostatistics*, **11**(2), 337–352.

Rosenblatt M (1952). "Remarks on a multivariate transformation." *The annals of mathematical statistics*, **23**(3), 470–472.

Taddy MA, Kottas A (2010). "A Bayesian nonparametric approach to inference for quantile regression." *Journal of Business & Economic Statistics*, **28**(3), 357–369.

US Geological Survey (2016). "National Water Information System data available on the World Wide Web (USGS Water Data for the Nation)." URL http://waterdata.usgs.gov/nwis/.

West M, Harrison PJ, Migon HS (1985). "Dynamic generalized linear models and Bayesian forecasting." *Journal of the American Statistical Association*, **80**(389), 73–83.

Yan Y, Kottas A (2017). "A new family of error distributions for Bayesian quantile regression." *arXiv preprint arXiv:1701.05666*.

Yu K, Moyeed RA (2001). "Bayesian quantile regression." *Statistics & Probability Letters*, **54**(4), 437–447.

Affiliation:

Raquel Barata
PhD Candidate, Department of Statistics
University of California Santa Cruz
Santa Cruz, CA 95064
E-mail: rbarata@ucsc.edu