qgam: Bayesian non-parametric quantile regression modelling in R

Matteo Fasiolo
University of Bristol

Simon N. Wood
University of Bristol

Margaux Zaffran
ENSTA Paris

Raphaël Nedellec
Électricité de France R&D

Yannig Goude
Électricité de France R&D

Abstract

Generalized additive models (GAMs) are flexible non-linear regression models, which can be fitted efficiently using the approximate Bayesian methods provided by the **mgcv** R package. While the GAM methods provided by **mgcv** are based on the assumption that the response distribution is modelled parametrically, here we discuss more flexible methods that do not entail any parametric assumption. In particular, this article introduces the **qgam** package, which is an extension of **mgcv** providing fast calibrated Bayesian methods for fitting quantile GAMs (QGAMs) in R. QGAMs are based on a smooth version of the pinball loss of Koenker (2005), rather than on a likelihood function, hence jointly achieving satisfactory accuracy of the quantile point estimates and coverage of the corresponding credible intervals requires adopting the specialized Bayesian fitting framework of Fasiolo, Goude, Nedellec, and Wood (2017). Here we detail how this framework is implemented in **qgam** and we provide examples illustrating how the package should be used in practice.

Keywords: Bayesian quantile regression, generalized additive models, regression splines, calibrated Bayes, fast Bayesian inference, R.

1. Introduction: additive quantile modelling in R

Generalized additive models (GAMs) (Hastie and Tibshirani 1986) are flexible regression models, where the relation between the response distribution and several covariates is modelled non-parametrically, typically via spline bases expansions. In standard GAMs only one parameter of the response distribution (typically the location) is modelled additively, but Rigby and Stasinopoulos (2005) developed methods for handling more flexible generalized additive

models for location, scale and shape (GAMLSS), where potentially all the parameters of the response distribution are allowed to depend on the covariates. The quantile GAM models (QGAMs) described in this work provide even more flexibility by modelling the quantiles of conditional response distribution individually, thus avoiding any parametric assumption on the distribution of response variable.

The purpose of this article is discussing methods and software for doing modelling with QGAMs in R. In particular, we focus on the **qgam** R package (Fasiolo, Wood, Zaffran, Goude, and Nedellec 2019b), which implements the fast calibrated Bayesian fitting methods proposed by Fasiolo *et al.* (2017). **qgam** is an extension of the recommended **mgcv** package (Wood 2019), which provides tools for building, fitting and visualizing GAM and GAMLSS models. A new package is required, because QGAMs can not be handled using the standard GAM methods implemented by **mgcv**. In particular, QGAMs are based on the pinball loss function (Koenker 2005), rather than on a probabilistic model of the response distribution, and the absence of a likelihood function impedes direct application of the Bayes' rule when updating the prior distribution on the regression coefficients given the observed responses. While this problem can be overcome by adopting the coherent Bayesian belief updating framework of Bissiri, Holmes, and Walker (2016), which effectively leads to the application of Bayes' rule using a loss-based pseudo-likelihood, naïvely plugging such pseudo-likelihood into standard Bayesian fitting methods can lead to inaccurate quantile estimates and inadequate coverage of the corresponding credible intervals. To avoid such issues, **qgam** implements the calibrated Bayesian methods of Fasiolo *et al.* (2017), which explicitly aim at selecting the 'learning rate' tuning parameter of the loss so as to achieve near-nominal frequentist coverage of the quantile credible intervals. Furthermore, **qgam** bases quantile regression on a smoothed version of the pinball loss, which enables the adoption of fast maximum a posterior (MAP) and empirical Bayes methods for, respectively, estimating the regression coefficient and selecting their prior variance hyper-parameters. The smoothness of the new loss is determined by minimizing the asymptotic mean squared error (MSE) of the estimated regression coefficients, approximated using a location-scale GAM model.

To our best knowledge and at the time of writing, the QGAM model fitting framework proposed by Fasiolo *et al.* (2017) is the only one able to estimate all the regression coefficients and prior hyper-parameters using fast direct optimization methods and to provide, at no extra computational cost, credible intervals which achieve adequate coverage for tail quantiles. An alternative to the calibrated Bayesian estimation methods provided by **qgam** is additive quantile regression via gradient boosting, available in R via the **mboost** package (Hothorn, Bühlmann, Kneib, Schmid, Hofner, Sobotka, Scheipl, and Mayr 2018). Fasiolo *et al.* (2017) shows that boosting can provide accurate point estimates, but selecting the optimal number of boosting steps requires running a computationally intensive cross-validation routine. Furthermore, the uncertainty of the estimated quantiles must be obtained by bootstrapping. Waldmann, Kneib, Yue, Lang, and Flexeder (2013) propose Bayesian methods for estimating QGAMs via the **BayesX** stand-alone software (Brezger, Kneib, and Lang 2005), with a specific quantile regression family being accessible from R using the **bamlss** package (Umlauf, Klein, Zeileis, Koehler, Simon, and Stadlmann 2018). However, their proposal is based on Markov chain Monte Carlo (MCMC) methods, which are slower than the direct optimization methods adopted here, and the resulting credible intervals struggle to achieve nominal frequentist coverage for tail quantiles, as detailed in Waldmann *et al.* (2013). Further alternatives are the **quantreg** (Koenker 2013) and the **VGAM** (Yee 2010) R packages, but in both cases the

prior smoothing hyper-parameters have to be selected manually. Lin, Bondell, Zhang, and Zou (2013) focus on variable selection, not smoothing, hence the corresponding software can not be considered an alternative to **qgam**.

The rest of the paper is structured as follows. In Section 2 we first present the basic structure of QGAM models, then we show how such models can be fitted using the fast calibrated Bayesian methods of Fasiolo *et al.* (2017) and how the fitting framework is implemented in the **qgam** package. In Section 3 we illustrate how the package can be used for quantile GAM modelling. In particular, the first example introduces the basic features of the package, while second considers a more realistic application, focused on electricity demand forecasting using functional effects. In Section 4 we outline promising directions for future work on additive quantile modelling with **qgam**.

2. Models, methods and software

2.1. General structure of additive quantile regression models

Let y be a continuous r.v. with conditional distribution $p(y|\boldsymbol{x})$, where $\boldsymbol{x}$ is a p-dimensional vector of covariates. Define also the conditional quantiles of $p(y|\boldsymbol{x})$ by $\mu_\tau(\boldsymbol{x}) = F^{-1}(\tau|\boldsymbol{x})$, where $\tau \in (0,1)$ and F^{-1} is the inverse conditional c.d.f. of y. In quantile regression (Koenker 2005) the conditional quantiles are modelled individually, without specifying a model for $p(y|\boldsymbol{x})$. Direct quantile estimation is generally achieved by exploiting the following alternative definition of a quantile

$$\mu_\tau(\boldsymbol{x}) = \operatorname*{argmin}_{\mu} \mathbb{E}\{\rho_\tau(y - \mu)|\boldsymbol{x}\}, \tag{1}$$

where

$$\rho_\tau(z) = (\tau - 1)\frac{z}{\sigma}\mathbb{1}(z < 0) + \tau\frac{z}{\sigma}\mathbb{1}(z \geq 0)$$

is the scaled version of the so-called 'pinball' or 'check' loss. $\sigma > 0$ is a scale parameter, which can potentially depend on the covariates $\boldsymbol{x}$. Its role will be clarified in Section 2.2. In **qgam** the pinball loss is replaced by the extended log-f (ELF) loss (Fasiolo *et al.* 2017), which is

$$\tilde{\rho}_\tau(z) = (\tau - 1)\frac{z}{\sigma} + \lambda \log(1 + e^{\frac{z}{\lambda\sigma}}), \tag{2}$$

where $\lambda > 0$, and the pinball loss is recovered as $\lambda \to 0$. The ELF loss is a smoothed version of the pinball loss which, as will be explained later, has the advantages of leading to more accurate quantile estimates (if λ is tuned adequately) and of enabling the use of efficient computational methods for model fitting.

In this work we assume that the conditional quantiles have an additive structure, that is

$$\mu(\boldsymbol{x}) = \sum_{j=1}^{J} f_j(\boldsymbol{x}),$$

where the f_j's are parametric, random or smooth effects. The latter are built using spline bases expansions, so the j-th effect can be written

$$f_j(\boldsymbol{x}) = \sum_{k=1}^{K_j} b_k^j(\boldsymbol{x})\beta_k^j,$$

where $b_1^j, \ldots, b_K^j$ are the spline basis function used to built the j-th effect and $\beta_1^j, \ldots, \beta_k^j$ are the corresponding regression coefficients. The basis functions are known and fixed, while the regression coefficients must be estimated. Notice that the quantile μ depends both on $\boldsymbol{x}$ and $\boldsymbol{\beta}$, but in the following we will refer to it using $\mu(\boldsymbol{x})$ or $\mu(\boldsymbol{\beta})$, depending on context.

In a Bayesian framework, the complexity of the smooth and random effects is controlled using a prior distribution on the regression coefficients, which we indicate with $p(\boldsymbol{\beta})$. In this work we assume that such prior is an improper multivariate Gaussian distribution, centred at $\boldsymbol{0}$ and with positive semi-definite precision matrix $\mathbf{S}^\gamma = \sum_{l=1}^m \gamma_l \mathbf{S}_l$. Here the $\mathbf{S}_l$'s are positive semi-definite matrices, scaled by the positive parameters $\boldsymbol{\gamma} = \{\gamma_1, \ldots, \gamma_l\}$. We assume that the $\mathbf{S}_j$'s are given, while the vector $\boldsymbol{\gamma}$ needs to be selected. The $\mathbf{S}_j$'s related to random effects are often simple diagonal matrices, while the $\mathbf{S}_j$'s related to smooth effects have more complex structures, aimed at penalizing departures from (some definition of) smoothness. Hence, increasing $\boldsymbol{\gamma}$ leads to a prior where the random effects coefficients are more concentrated around zero and the smooth effects less wiggly. In the following we refer to $\boldsymbol{\gamma}$ as the vector of smoothing parameters, with the understanding that some of its elements could instead be controlling the prior precision of the random effects. In general, there needs not to be a one-to-one correspondence between the smoothing parameters and the effects because, for example, a smoothing parameter might be determining the prior precision of several effects or the complexity of a single effect might controlled using multiple smoothing parameters. See Wood (2017) for detailed account on smooth and random effects model structures, in an additive modelling context.

This section outlined the basic QGAM modelling framework, the next two will detail how such models can be estimated within a fast calibrated Bayesian framework.

2.2. Performing the Bayesian update under the ELF loss

Let us indicate with $p(y|\boldsymbol{\beta})$ the true conditional distribution of the response, where the dependency on $\boldsymbol{x}$ has been temporarily suppressed to simplify the notation, and indicate with $p(\boldsymbol{\beta})$ the prior distribution, which is implicitly a function of the smoothing parameters $\boldsymbol{\gamma}$. Assume for the moment that the latter have been fixed to some value, so the prior is given, and recall that we are basing additive quantile regression on the ELF loss, rather than on a probabilistic model for $p(y|\boldsymbol{\beta})$. This is an impediment to performing Bayesian inference for QGAMs, as the lack of a likelihood function prevents us from applying Bayes' rule to update $p(\boldsymbol{\beta})$ to the corresponding posterior, $p(\boldsymbol{\beta}|y)$. We address the issue by adopting the belief updating (BU) framework of Bissiri *et al.* (2016), which allows to perform the Bayesian update using a general loss function, rather than a likelihood. In particular, as detailed in Fasiolo *et al.* (2017), applying the BU framework to the ELF loss leads to the following update formula

$$p(\boldsymbol{\beta}|y) \propto \tilde{p}_\tau\{y - \mu(\boldsymbol{\beta})\}p(\boldsymbol{\beta}), \tag{3}$$

where

$$\tilde{p}_\tau(y - \mu) = \frac{e^{-\tilde{\rho}_\tau\{y-\mu\}}}{\int e^{-\tilde{\rho}_\tau\{y-\mu\}}dy} = \frac{e^{(1-\tau)\frac{y-\mu}{\sigma}}(1 + e^{\frac{y-\mu}{\lambda\sigma}})^{-\lambda}}{\lambda\sigma \mathrm{Beta}[\lambda(1-\tau), \lambda\tau]}, \tag{4}$$

is the p.d.f. of the ELF distribution, which is an extension of the log-f distribution of Jones (2008). Here λ determines the smoothness of the loss, while $1/\sigma$ plays the role of a 'learning rate', determining the relative weights of the loss-based pseudo-likelihood and the prior. In contexts where the variance of y varies strongly with $\boldsymbol{x}$, it can be advantageous to let σ depend

on $\boldsymbol{x}$ via the decomposition $\sigma(\boldsymbol{x}) = \sigma_0 \tilde{\sigma}(\boldsymbol{x})$, where σ_0 is the baseline or average learning rate and $\tilde{\sigma}(\boldsymbol{x})$ is its $\boldsymbol{x}$-dependent component.

Having defined (3) which, following Syring and Martin (2015), we refer to as the Gibbs posterior, the next section outlines how the fast calibrated Bayesian methods proposed by Fasiolo *et al.* (2017), and implemented by **qgam**, can be used to fit ELF-based QGAMs.

2.3. Fast calibrated Bayesian model fitting methods

We fit quantile GAMs using three nested optimization routines, in particular:

1. in the outer iteration, the baseline learning rate $1/\sigma_0$ is selected by numerically minimizing a calibration loss function;

2. for fixed σ_0, the loss smoothness λ and the $\boldsymbol{x}$-dependent component of the learning-rate $\tilde{\sigma}(\boldsymbol{x})$ are determined using closed-form expressions, while the smoothing parameters $\boldsymbol{\gamma}$ are selected by numerically optimizing an intermediate criterion;

3. for fixed $\sigma(\boldsymbol{x}) = \sigma_0 \tilde{\sigma}(\boldsymbol{x})$, λ and $\boldsymbol{\gamma}$, the regression coefficient $\boldsymbol{\beta}$ are estimated by numerically optimizing an inner criterion.

Of course parameter $\tau \in (0, 1)$, which indicates the quantile of interest, is held fixed throughout. The fact that the three iterations are nested, not sequential, means that evaluating the outer objective function requires solving the intermediate optimization problem and, in turn, each step of the latter entails running the inner optimization to convergence. In the following we provide a methodological outline of the three iterations, going from the inner to the outer one, and in Section 2.4 we describe how the whole procedure is implemented in **qgam**.

Inner iteration: MAP estimation of the regression coefficients

The inner iteration estimates the regression coefficients using efficient maximum a posterior (MAP) methods, for fixed $\boldsymbol{\gamma}$, $\sigma(\boldsymbol{x})$ and λ. In particular, let $\mathbf{y} = \{y_1, \ldots, y_n\}$ be the vector of observed responses and notice that the conditional quantile is modelled by $\mu(\boldsymbol{x}_i) = \mathbf{x}_i^\mathsf{T} \boldsymbol{\beta}$, where $\mathbf{x}_i$ is the i-th row of the $n \times d$ design matrix $\mathbf{X}$, containing the spline basis functions evaluated at $\boldsymbol{x}_i$. It is easy to show that maximizing the logarithm of the ELF-based Gibbs posterior (3) is equivalent to minimizing the criterion

$$\tilde{V}_D\{\boldsymbol{\beta}, \boldsymbol{\gamma}, \sigma(\boldsymbol{x}), \lambda\} = \sum_{i=1}^{n} \mathrm{Dev}_i\{\boldsymbol{\beta}, \sigma(\boldsymbol{x}_i), \lambda\} + \sum_{l=1}^{m} \gamma_j \boldsymbol{\beta}^\mathsf{T} \mathbf{S}_j \boldsymbol{\beta}, \tag{5}$$

where $\mathrm{Dev}_i\{\boldsymbol{\beta}, \sigma(\boldsymbol{x}_i), \lambda\}$ is the i-th deviance component, based on the ELF density (4). Criterion (5) is smooth and can be minimized efficiently w.r.t. $\boldsymbol{\beta}$ using a penalized iteratively re-weighted least squares (PIRLS) algorithm, as detailed in Fasiolo *et al.* (2017).

Intermediate iteration: selection of the smoothing parameters and ELF loss smoothness

The intermediate iteration selects the smoothing parameters by maximizing a Laplace approximation to the ELF-based marginal likelihood. The latter is

$$\log p\{\mathbf{y}|\boldsymbol{\gamma}, \sigma(\boldsymbol{x}), \lambda\} = \int \tilde{p}_\tau(y_i - \mathbf{x}_i^\mathsf{T}\boldsymbol{\beta}) p(\boldsymbol{\beta}|\boldsymbol{\gamma}) d\boldsymbol{\beta}, \tag{6}$$

where we have made explicit the dependence of the prior on the smoothing parameters. Applying a Laplace approximation to the intractable integral (6) leads to the following Laplace approximate marginal likelihood (LAML) criterion

$$
\begin{aligned}
G\{\boldsymbol{\gamma}, \sigma(\boldsymbol{x}), \lambda\} = \; & -\frac{1}{2}\tilde{V}_D\{\hat{\boldsymbol{\beta}}, \boldsymbol{\gamma}, \sigma(\boldsymbol{x}), \lambda\} + \tilde{\mathrm{ll}}\{\sigma(\boldsymbol{x}), \lambda\} \\
& -\frac{1}{2}\Big[\log|\mathbf{X}^{\mathsf{T}}\mathbf{W}\mathbf{X} + \mathbf{S}^{\boldsymbol{\gamma}}| - \log|\mathbf{S}^{\boldsymbol{\gamma}}|_+\Big] + \frac{M_p}{2}\log(2\pi),
\end{aligned}
\tag{7}
$$

where $\hat{\boldsymbol{\beta}}$ is the minimizer of (5), estimated using the inner iteration, $\tilde{\mathrm{ll}}\{\sigma(\boldsymbol{x}), \lambda\}$ is the saturated log-likelihood corresponding to the ELF density (4), M_p is the dimension of the null-space of $\mathbf{S}^{\boldsymbol{\gamma}}$ and $|\mathbf{S}^{\boldsymbol{\gamma}}|_+$ is the product of its non-zero eigenvalues. Fasiolo *et al.* (2017) focuses on a marginal loss criterion, which is equal to the negative of (6) up to a normalization constant, to stress that σ_0 should not be selected by jointly maximizing (7) w.r.t. $\boldsymbol{\gamma}$ and σ_0, as for standard GAMs, but the outer calibration procedure for σ_0 should be used instead. Here we prefer to adopt a likelihood-based terminology to emphasize that (7) can be maximized w.r.t. $\boldsymbol{\gamma}$ using the numerically stable methods of Wood, Pya, and Säfken (2016), which are aimed at standard probabilistic GAMs.

Before describing the outer iteration, we need to specify how parameters $\tilde{\sigma}(\boldsymbol{x})$ and λ, which are held constant throughout the intermediate and inner iterations, are determined for fixed σ_0. Recall that $\sigma(\boldsymbol{x}) = \sigma_0\tilde{\sigma}(\boldsymbol{x})$ and define the scaled loss bandwidth $h(\boldsymbol{x}) = \sigma(\boldsymbol{x})\lambda$. Assume that the responses follow the location-scale model $y_i|\boldsymbol{x}_i \sim \alpha(\boldsymbol{x}_i) + \kappa(\boldsymbol{x}_i)z_i$, where the z_i's are i.i.d with $\mathbb{E}(z|\boldsymbol{x}) = 0$ and $\mathrm{var}(z|\boldsymbol{x}) = 1$. Fasiolo *et al.* (2017) shows that, under the above location-scale model and further assumptions specified therein, the asymptotic MSE of the regression coefficients is minimized by

$$
\tilde{h}^*(\boldsymbol{x}) = \left[\frac{d}{n}\frac{9f_z\{F_z^{-1}(\tau)\}}{\pi^4 f_z'\{F_z^{-1}(\tau)\}^2}\right]^{\frac{1}{3}}\kappa(\boldsymbol{x}),
\tag{8}
$$

where f_z, f_z' and F_z are the p.d.f. of z, its first derivative and its c.d.f., $F_z^{-1}(\tau)$ is the τ-th quantile of z and d the dimensionality of $\boldsymbol{\beta}$. Having determined $\tilde{h}^*(\boldsymbol{x})$, we impose $n^{-1}\sum_i \tilde{\sigma}(\boldsymbol{x}_i) = 1$ so that $\lambda = n^{-1}\sum_i \tilde{h}^*(\boldsymbol{x}_i)/\sigma_0$. The idea is to use $\tilde{\sigma}(\boldsymbol{x})$ to modulate the baseline scaled loss smoothness, $\lambda\sigma_0$, and learning rate, $1/\sigma_0$, to make them respectively directly and inversely proportional to the conditional variance of y. Hence, for fixed σ_0, we can obtain $\tilde{\sigma}(\boldsymbol{x})$ and λ using the formulas just mentioned. This, of course, requires estimates of $\alpha(\boldsymbol{x})$ and $\kappa(\boldsymbol{x})$ in the location-scale model, as well as of f_z and F_z. Such estimates need to be obtained only once, before initiating the nested QGAM fitting procedure described here, hence we delay discussing the specific approach we follow until Section 2.4, where we also describe its software implementation.

Outer iteration: selection of the learning rate by calibrated Bayes

Recall that $1/\sigma_0$ is the baseline learning rate, determining the relative weight of the loss and the prior in the Gibbs posterior (3). Increasing $1/\sigma_0$ leads to faster learning, that is wigglier fitted quantile curves and narrower posterior credible intervals. Fasiolo *et al.* (2017) selects σ_0 using a calibration procedure aimed at guaranteeing that the credible intervals for $\mu(\boldsymbol{x})$ approximately achieve the correct frequentist coverage. This is achieved by minimizing

$$
\hat{\mathrm{IKL}}(\sigma_0) = n^{-1}\sum_{i=1}^{n}\left[\frac{\hat{\tilde{v}}(\boldsymbol{x}_i)}{v(\boldsymbol{x}_i)} + \log\frac{v(\boldsymbol{x}_i)}{\hat{\tilde{v}}(\boldsymbol{x}_i)}\right]^{\zeta},
\tag{9}
$$

which is an estimate of the following integrated Kullback-Leibler (IKL) divergence

$$\text{IKL}(\sigma_0) = \int \text{KL}\left[\text{N}\{\mu(\boldsymbol{x}), \tilde{v}(\boldsymbol{x})\}, \text{N}\{\mu(\boldsymbol{x}), v(\boldsymbol{x})\}\right]^\zeta p(\boldsymbol{x}) d\boldsymbol{x} \propto \int \left\{\frac{\tilde{v}(\boldsymbol{x})}{v(\boldsymbol{x})} + \log \frac{v(\boldsymbol{x})}{\tilde{v}(\boldsymbol{x})}\right\}^\zeta p(\boldsymbol{x}) d\boldsymbol{x}.$$

Here $\text{KL}(\cdot, \cdot)$ and $\text{N}(\cdot, \cdot)$ indicate respectively the KL divergence and the univariate Gaussian distribution, ζ is a positive parameter which we fix to $1/2$, $v(\boldsymbol{x}) = \mathbf{x}^\mathsf{T}\mathbf{V}\mathbf{x}$ and $\tilde{v}(\boldsymbol{x}) = \mathbf{x}^\mathsf{T}\tilde{\mathbf{V}}\mathbf{x}$ are the posterior variance of $\mu(\boldsymbol{x})$ under two alternative posterior covariance matrices for $\boldsymbol{\beta}$. In particular $\mathbf{V} = (\mathcal{I} + \mathbf{S}^\gamma)^{-1}$ and $\tilde{\mathbf{V}} = (\mathcal{I}\boldsymbol{\Sigma}_\nabla^{-1}\mathcal{I} + \mathbf{S}^\gamma)^{-1}$, where $\mathcal{I}$ is the negative Hessian of the ELF-based log-likelihood and $\boldsymbol{\Sigma}_\nabla = \text{cov}[\nabla_{\boldsymbol{\beta}}\tilde{\rho}\{y - \mu(\boldsymbol{\beta})\}|_{\boldsymbol{\beta}=\hat{\boldsymbol{\beta}}}]$ is the covariance matrix of the gradient of the ELF loss, under the data generating process. $\hat{\tilde{v}}(\boldsymbol{x}_i)$ is an estimate of $\tilde{v}(\boldsymbol{x}_i)$, based on the regularized estimator of $\tilde{\mathbf{V}}$ described in Fasiolo *et al.* (2017). The IKL objective function is generally smooth and convex, hence we minimize it efficiently using Brent's method (Brent 2013).

Fasiolo *et al.* (2017) propose also an alternative calibration procedure, based on another version of the IKL loss where $\tilde{v}(\boldsymbol{x})$ is substituted by the sample pointwise variance of the estimated quantile $\hat{\mu}(\boldsymbol{x})$, estimated by bootstrapping. Such procedure require re-fitting the model to several bootstrap samples, hence it is more computationally expensive than minimization of the IKL loss defined above, but it can lead to better coverage in small samples. We refer to Fasiolo *et al.* (2017) for explanations regarding how minimizing either version of the IKL loss w.r.t. σ_0 leads to better frequentist coverage of the posterior credible intervals, relative to joint maximization of LAML (7) w.r.t. σ_0 and $\boldsymbol{\gamma}$.

This section outlined the Bayesian QGAM fitting framework of Fasiolo *et al.* (2017) from a methodological point of view. The next section explains how the fitting framework just described is implemented in the **qgam** package.

2.4. Software implementation of the fitting framework

The **qgam** package is an extension of **mgcv** providing additional tools for handling QGAMs. Here we focus mainly on how the package implements the fitting framework of Fasiolo *et al.* (2017), while in Section 3 we provide usage examples.

The main function provided by **qgam** is itself called `qgam` and has the following arguments

```
qgam(form, data, qu, lsig = NULL, err = NULL,
  multicore = !is.null(cluster), cluster = NULL, ncores = detectCores() - 1,
  paropts = list(), control = list(), argGam = NULL)
```

Arguments `form` and `data` have the same meaning as in `mgcv::gam` (henceforth just `gam`), `qu` is the quantile of interest τ, `lsig` is $\log\sigma_0$ and `err` is a positive parameter allowing to set the ELF loss smoothness manually. By default `qgam` determines the loss smoothness automatically, using the methods described in Section 2.3, hence most users will not need to use argument `err`. However, Appendix A provides some guidelines for users who decide to select the loss smoothness manually. In `qgam` version `1.3.1`, arguments `multicore`, `cluster`, `ncores` and `paropts` are relevant only when the learning rate is calibrated by bootstrapping, and can be used to perform bootstrap re-fitting in parallel. `control` is a list of control parameters useful, for instance, to choose the type of calibration used, and `argGam` is a list of arguments to be passed to `gam`.

To describe how **qgam** implements the fitting framework of Fasiolo *et al.* (2017), assume that we want to fit a QGAM for quantile $\tau = 0.5$, with model formula `myForm = list(y ~ s(x1) + s(x2), ~ s(x3))` and using data from a `data.frame` called `myDat`. Here `y ~ s(x1) + s(x2)` is the model used for the median, that is $\mu_\tau(\boldsymbol{x}) = f_1(x_1) + f_2(x_2)$, where f_1 and f_2 are two smooth effects constructed, by default, using ten thin plate splines basis function. `~ s(x3)` is the model used for the variance $\kappa(\boldsymbol{x})$ in the preliminary location-scale GAM model fit. The first thing to point out is that, if σ_0 and the bandwidth $h(\boldsymbol{x})$ were known, QGAM models could be fitted using `gam` directly. In fact, we could fit a median quantile model using

```
gam(myForm[[1]], data = myDat,
  family = elf(theta = 1, qu = 0.5, co = rep(1, nrow(myDat))))
```

where `qgam::elf` is a family providing the ELF log-likelihood function and its derivatives. Its argument `theta` and `co` correspond respectively to $\log \sigma_0$ and $h(\boldsymbol{x})$, which have been arbitrarily fixed to 1 here. Hence, given σ_0 and $h(\boldsymbol{x})$, the intermediate and inner iterations for selecting $\boldsymbol{\gamma}$ and estimating $\boldsymbol{\beta}$ take place inside `gam`. But these parameters are generally unknown, so **qgam** determines σ_0 using the outer iteration and $h(\boldsymbol{x})$ via the preliminary location-scale GAM fit. In particular, upon calling

```
qgam(myForm, data = myDat, qu = 0.5)
```

the `qgam` function executes the following pseudo-code:

1. estimate $\alpha(\boldsymbol{x})$ and $\kappa(\boldsymbol{x})$ by fitting a Gaussian location-scale GAM

   ```
   gam(myForm, data = myDat, family = gaulss)
   ```

   where `mgcv::gaulss` is a Gaussian location-scale family.

2. numerically fit the sinh-arch density of Jones and Pewsey (2009) to the standardized Gaussian GAM residuals, to provide estimates of f_z, f_z' and $F_z^{-1}(\tau)$.

3. estimate $\tilde{h}^*(\boldsymbol{x})$ by plugging the estimates obtained in previous two steps in (8), where d is set to the number of effective degrees of freedom used to model $\alpha(\boldsymbol{x})$ in step 1.

4. call the `tuneLearnFast` function, which minimizes the IKL loss w.r.t. $\log \sigma_0$ using an outer Brent algorithm. For each trial value of $\log \sigma_0$:

   (a) call `gam` with arguments `formula = myForm[[1]]` and `family = elf`, the latter having parameter `theta` fixed to the current value of $\log \sigma_0$ and `co` fixed to $\tilde{h}^*(\boldsymbol{x})$. ELF parameters λ and $\sigma(\boldsymbol{x})$ are obtained within `elf` by doing $\lambda = n^{-1} \sum_i \tilde{h}^*(\boldsymbol{x}_i)/\sigma_0$ and then $\sigma(\boldsymbol{x}_i) = \tilde{h}^*(\boldsymbol{x}_i)/\lambda$. The `gam` call performs the intermediate iteration for selecting $\boldsymbol{\gamma}$ and the inner iteration for estimating $\boldsymbol{\beta}$.

   (b) Estimate the IKL loss using (9), where $v(x)$ is easily obtained from the `gam` output, while $\tilde{v}(\boldsymbol{x})$ is estimated using the regularized estimator of Fasiolo *et al.* (2017).

   Upon convergence, return the value of $\log \sigma_0$ minimizing the IKL loss.

5. Call `gam` as in step 4(a), with `elf` parameter `theta` fixed to the value of $\log \sigma_0$ returned by `tuneLearnFast`, and `co` set to $\tilde{h}^*(\boldsymbol{x})$.

6. Return the output of the last `gam` call, with the output of `tuneLearnFast` call stored in its `$calibr` slot.

Notice that a variety of different density estimators could be used in step 2 above. We use the sinh-arch distribution because it provides a good balance between flexibility and stability. However, future versions of `qgam` might allow user to provide their own density estimator. Similar considerations hold for the Gaussian GAM used in step 1. The `tuneLearnFast` function of step 4 has exactly the same arguments as `qgam`, and it returns a list containing diagnostic information regarding the outer iteration and the value of σ_0 which minimizes the IKL loss. The output of `qgam` is an object of class `c("qgam", "gam", "glm", "lm")`, which can be manipulated using all the methods available for objects of class `"gam"`.

3. Examples

3.1. A basic example: the motorcycle data set

Here we consider the classic motorcycle accident data set of Silverman (1985), available in the MASS package (Ripley, Venables, Bates, Hornik, Gebhardt, and Firth 2013). We start by loading the data and fitting a QGAM for quantile $\tau = 0.9$ as follows

```
R> library("MASS")
R> library("qgam")
R> fitCycle1 <- qgam(list(form = accel ~ s(times, k = 20, bs = "ad"),
+    ~ s(times)), data = mcycle, qu = 0.9)
```

We are using the model $\mu_\tau(\text{time}) = f(\text{time})$, where $f(\cdot)$ is an adaptive smooth effect built using 20 P-splines basis functions (Eilers and Marx 1996). We use an adaptive smooth because, as shown in Figure 1, the shape of the conditional quantile varies sharply between roughly 10 and 40 milliseconds, but it is otherwise quite flat. Recall that the second part of the formula, `~ s(times)`, is not used to model the quantile, but the variance $\kappa(\boldsymbol{x})$ of the response within the location-scale GAM model.

Given that `fitCycle1` inherits from class `"gam"`, generic function calls such as

```
R> predict(fitCycle1)[1:5]
```

```
          1           2           3           4           5
0.460372275 0.421737850 0.271621363 0.146774904 0.005606131
```

dispatch to the relevant method (here `predict.gam`). Similarly, the fitted effects in the quantile model can be plotted using

```
R> plot(fitCycle1)
```

which dispatches to `plot.gam`, and produces the plot shown in Figure 1. To save memory, no component of the Gaussian location-scale fit is stored in the output of `qgam`, but the selected values of the parameters $\log \sigma_0$ and $h(\boldsymbol{x})$ can be extracted by doing
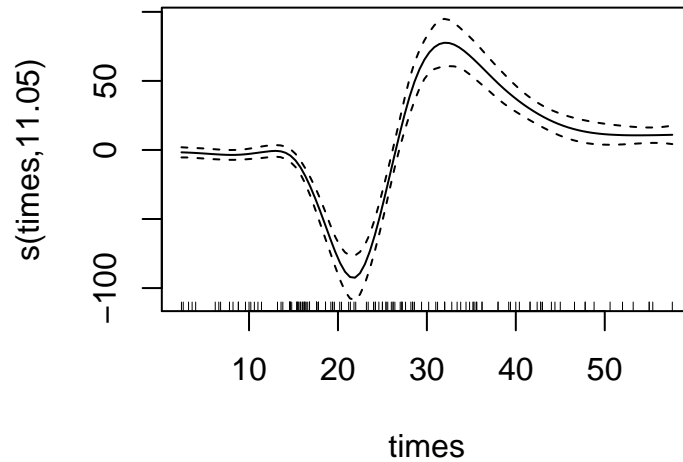
Figure 1:   Fitted smooth effect of time on quantile 0.9 of the `mcycle` data set, with 95% credible intervals obtained using a Gaussian approximation to the posterior.

```
R> fitCycle1$family$getTheta()

     0.9
1.237221


R> fitCycle1$family$getCo()[1:5]


[1] 0.2791352 0.2728345 0.2549963 0.2442300 0.2346063
```

Above we fitted a single quantile, but multiple quantiles can be estimated at once using the following function call

```
R> fitCycleM <- mqgam(list(form = accel ~ s(times, k = 20, bs = "ad"),
+     ~ s(times)), data = mcycle, qu = c(0.1, 0.25, 0.5, 0.75, 0.9))
```

where the `mqgam` function fits a QGAM to each of the five quantiles. The main advantage of using `mqgam`, rather than several calls to `qgam`, is that it saves memory by storing a single version of potentially large objects, such as the model matrix, for all fitted QGAMs. The five fitted QGAM models are in the list `fitCycleM$fit`, but calling fuctions such as `plot(fitCycleM$fit[[1]])` would trigger an error because some object elements are missing. The output of `mqgam` should instead be manipulated using the `qdo` function, for example

```
R> qdo(fitCycleM, qu = 0.25, fun = summary)



Family: elf
Link function: identity


Formula:
```

```
accel ~ s(times, k = 20, bs = "ad")

Parametric coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -42.739      2.016   -21.2   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Approximate significance of smooth terms:
           edf Ref.df Chi.sq p-value
s(times) 10.55  11.83  837.3  <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

R-sq.(adj) =  0.735   Deviance explained = 89.2%
-REML = 570.51  Scale est. = 1          n = 133
```

shows the output of `summary.gam` for the QGAM fitted to quantile $\tau = 0.25$. Similarly, the fitted effect for $\tau = 0.1$ can be plotted by doing `qdo(fitCycleM, qu = 0.1, fun = plot)`.

More convenient methods for plotting multiple QGAMs jointly are provided by the **mgcViz** package Fasiolo, Nedellec, Goude, and Wood (2019a). In the following we will only use the **mgcViz** tools that are specific to quantile GAMs, and we refer to Fasiolo, Nedellec, Goude, and Wood (2018) and to the relevant package documentation for more details. To exploit the visualizations offered by **mgcViz**, it is necessary to transform the output of `mqgam` as follows

```
R> library("mgcViz")
R> fitCycleM <- getViz(fitCycleM)
```

which produces an object of class `c("mqgamViz", "mgamViz")`. This is simply a list of objects of class `"gamViz"`, which inherits from the `"qgam"` class. Given that the elements of `fitCycleM` are full objects, they can be manipulated directly rather than via `qdo`. For example, we can use `plot(fitCycleM[[1]])` to plot the fitted effect for quantile $\tau = 0.1$. Of course, transforming `fitCycleM` using `getViz` has some memory cost, because `fitCycleM` now stores a copy of the model matrix for each quantile. Having transformed `fitCycleM`, the effect of time on all quantiles can now be visualized jointly by doing

```
R> plot(fitCycleM)
```

which produces the plot shown in Figure 2, by dispatching to `mgcViz::plot.mgamViz`. The latter produces an object of class `c("plotGam", "gg")`, which is a wrapper around one or several objects of class `ggplot`, defined in the **ggplot2** package (Wickham, Chang, Henry, Pedersen, Takahashi, Wilke, Woo, and Yutani 2019). Notice that Figure 2 shows the fitted effects for each τ which, in this case, are equal to the fitted quantiles up to some vertical shifts (the intercepts). The effects cross each other because they are all centered (see Wood (2017) for details on identifiability constraints in GAMs), hence this does not imply that the corresponding quantile estimates cross within the range of the data (time $\in (0, 60)$). Quantiles fitted with **qgam** typically cross somewhere (outside the data range in this example), because
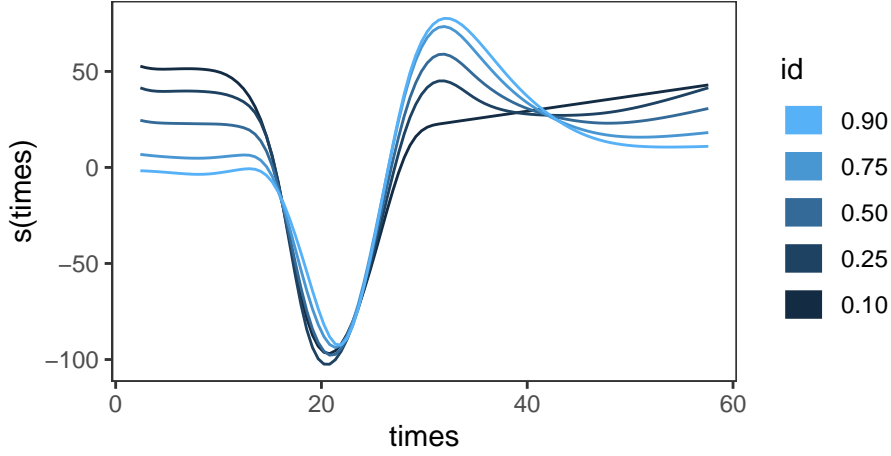
Figure 2: Plot showing the fitted additive effects of time on all five quantiles, obtained using the **mgcViz** package.

they are fitted independently and without any non-crossing constraint. This is a well known drawback of quantile regression (Koenker 2005).

In the `mcycle` data set, the variance of the response (`accel`) varies strongly with the independent variable (`times`), hence it is natural to let the learning rate and the ELF loss smoothness vary along the latter. This is achieved by providing the `~ s(times)` formula, which is used by `qgam` to model the conditional variance $\kappa(\boldsymbol{x})$ in the location-scale GAM model. However, it is possible to avoid modelling the variance and to assume that it is constant along the covariates. For example, in the following code

```
R> fitCycleConst <- qgam(accel ~ s(times, k = 20, bs = "ad"),
+    data = mcycle, qu = 0.9)
```

we are providing only the model for the quantile $\tau = 0.9$, rather than a list of two formulas, so `qgam` uses a constant learning rate $1/\sigma(\boldsymbol{x}) = 1/\sigma_0$ and loss bandwidth $\tilde{h}^*(\boldsymbol{x}) = \tilde{h}^*$. For the `mcycle` data set, ignoring the heteroscedasticity leads to the poor results shown in Figure 3. One problem with the quantile fit shown on the right is that it lies far above all the responses, for time < 10ms. But we are fitting quantile 0.9, hence we should expect around 10% of the responses to lay above the fit. The issue is that the bias of the ELF loss used by `qgam` is inversely proportional to the conditional variance of the response (see Fasiolo *et al.* (2017) for details), hence not modelling the response variance leads to high bias for time < 10ms. Ignoring heteroscedasticity can also lead to credible intervals that have non-constant coverage. In fact, the width of the intervals should be proportional to the variance of the response, but on the right plot in Figure 3 they have nearly constant width, thus providing an incorrect representation of the uncertainty of the fit. Of course, in most data set, heteroscedasticity is not as dramatic as in `mcycle` and using a location-only model often leads to satisfactory results. However, it is important to be aware of the issues highlighted by this example.
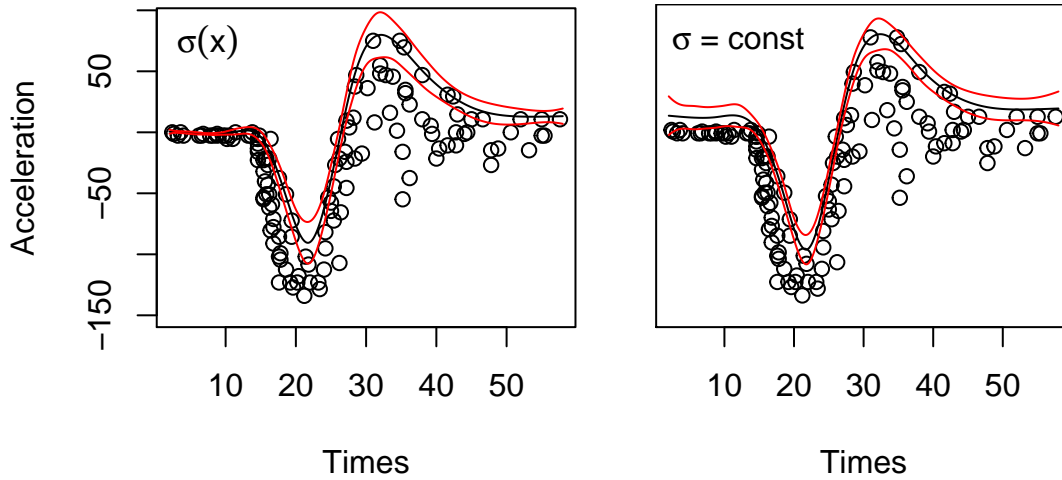
Figure 3: Fitted quantile for $\tau = 0.9$, with 95% credible intervals, when learning rate and loss smoothness are allowed to vary with the independent variable (left) or not (right).

3.2. Functional QGAM modelling of electricity demand data

In this example we consider smart meter data from $n_c = 247$ anonymized residential customers from Sidney, Australia, covering the period between the 3rd of July 2010 and the 30th of June 2011. The data has been downloaded from `https://www.ausgrid.com.au`, and it originally contained electricity demand from 300 customers, at 30min resolution. We discarded 53 customers because their demand was too irregular (e.g., they were absent from home for over 30 consecutive days), and we integrated the demand data with temperature data from the National Climatic Data Center (`www.ncdc.noaa.gov`), covering the same period. The aim of this example is illustrating how to use quantile GAMs in **qgam**, in the context of an electricity demand forecasting application. We start by loading the following data set:

```
R> data("AUDem", package = "qgam")
R> meanDem <- AUDem$meanDem
R> head(meanDem, 3)

  doy  tod       dem dow      temp                date     dem48
1 184 18.0 0.8248777 Sat 3.357407 2010-07-03 17:30:00 0.8978636
2 184 18.5 0.8686110 Sat 2.517073 2010-07-03 18:00:00 0.9417633
3 184 19.0 0.8519471 Sat 1.898399 2010-07-03 18:30:00 0.9148921
```

which contains the variables:

- `doy` the day of the year, from 1 to 365;

- `tod` the time of day, ranging from 18 to 22, where 18 indicates the period from 17:00 to 17:30, 18.5 the period from 17:30 to 18:00 and so on;

- `dem` the demand (in KW) during a 30min period, averaged over the n_c households;

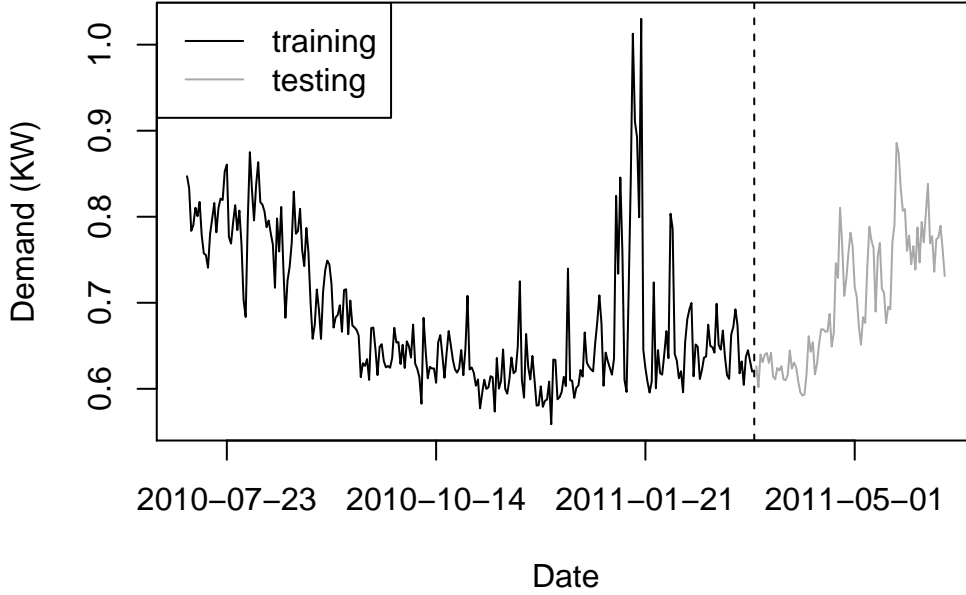- `dow` factor variable indicating the day of the week;

Figure 4:   Electricity demand between 19:30 and 20:00, averaged across the n_c customers.

- `temp` the external temperature at Sidney airport, in degrees Celsius;

- `date` local date and time;

- `dem48` the lagged mean demand, that is the average demand (`dem`) during the same 30min period of the previous day;

Assume that we aim at producing a probabilistic forecast of the average demand one day ahead, using additive quantile regression. We start by dividing the data into a learning and a testing set:

```
R> library("dplyr")
R> cutDate <- as.Date("2011-04-01 00:00:00")
R> meanDemLearn <- filter(meanDem, as.Date(date) < cutDate)    # Learning
R> meanDemTest <- filter(meanDem, as.Date(date) >= cutDate)    # Testing
```

which leaves the last three months for testing. Figure 4 shows the average demand between 19:30 and 20:00, over the whole period. We focus only on the period between 17:30 and 21:30, because the demand dynamics change considerably before and after this time slot, hence modelling demand across the whole day would require a much more sophisticated and computationally expensive model than any of those considered below.

We start by modelling the τ-th quantile of the average demand using the quantile model

$$\mu_\tau(\boldsymbol{x}_t) = \sum_{j=1}^{7} \beta_j \mathbb{I}(\mathrm{dow}_t = j) + \beta_8 \mathrm{dem}_{t-48} + f_1(\mathrm{tod}_t) + f_2(\mathrm{temp}_t) + f_3(\mathrm{doy}_t), \qquad (10)$$

where $\mathbb{I}(\mathrm{dow}_t = j) = 1$ when dow_t is the j-th day of the week and zero otherwise, f_1 and f_2 are smooth effects built using thin plate splines, while f_3 is a cyclical effect constructed using
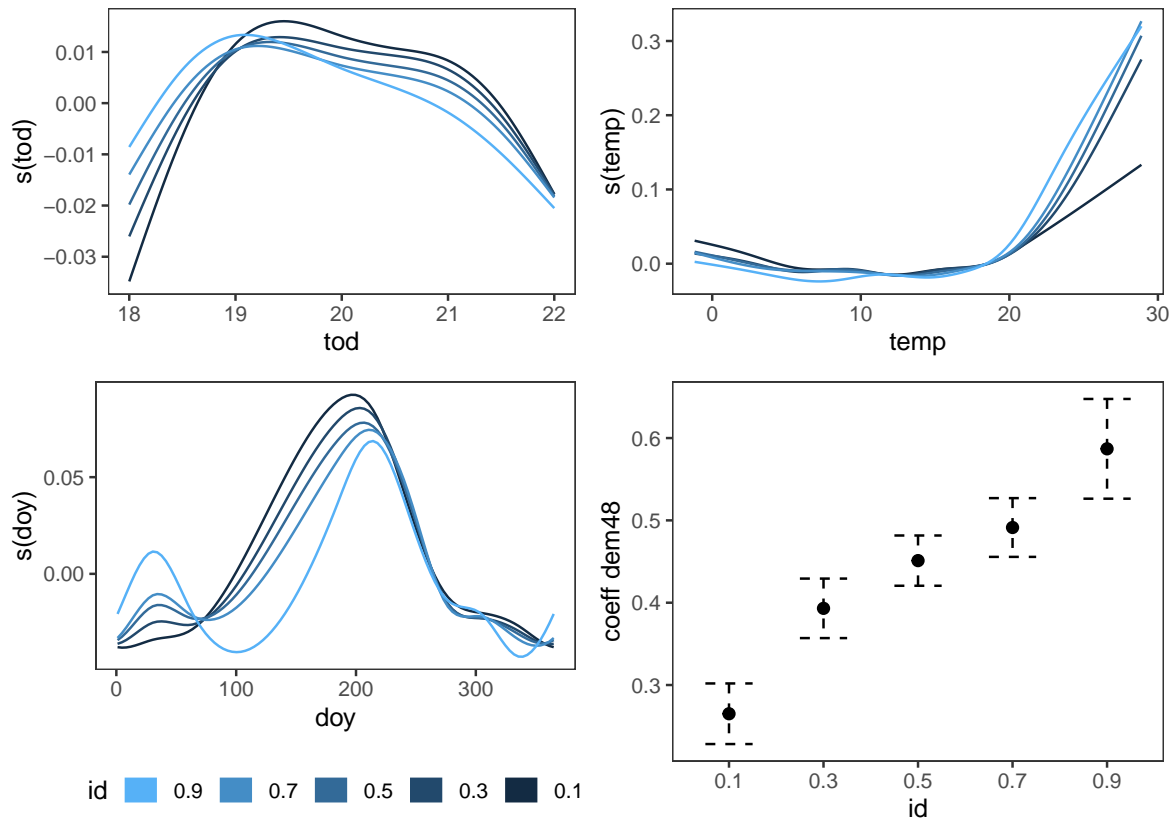
Figure 5: Estimated effects of time of day (`tod`), temperature (`temp`), day of year (`doy`) and lagged demand (`dem48`) from model (10), for the quantiles labelled using the `id` variable.

cubic regression splines. We use a cyclic effect for `doy` to make so that the seasonal effect has the same value on the 31st of December and on the 1st of January. Given that demand variability seems to spike during the austral summer (see Figure 4), we model the variance $\kappa(\boldsymbol{x})$ using a cyclic effect for `doy` in the location-scale Gaussian GAM.

We fit the model just described to quantiles $\tau = 0.1, 0.3, 0.5, 0.7, 0.9$ using the following code

```
R> library("qgam")
R> qusObj <- seq(0.1, 0.9, length.out = 5)
R> fitQ <- mqgam(list(dem ~ dow + dem48 +
+    s(tod, k = 6) + s(temp) + s(doy, bs = "cc"), ~ s(doy, bs = "cc")),
+    qu = qusObj, data = meanDemLearn)
```

We then plot the estimated smooth effects and the linear coefficients of lagged demand using

```
R> library("mgcViz")
R> fitQ <- getViz(fitQ)
R> print(plot(fitQ, allTerms = TRUE, select = c(1:3, 5)), pages = 1)
```

The output, up to some aesthetic manipulation, is shown in Figure 5. Notice that we used the plotting tools provided by `mgcViz`, which required us to convert the output of `mqgam` using

`getViz`. Then we used `plot` to plot all model terms (`allTerms = TRUE` made so that also the parametric terms were plotted), we selected a subset of the effects using the `select` argument and we used `print` to arrange all the plots on a single page.

The shapes of the estimated effects seem quite reasonable: the `tod` effect increases as people get home and decreases later in the evening, the temperature plot shows a strong cooling effect, but very little heating effect, and the effect of `doy` shows a main mode during the winter and a lower, narrower, mode during the summer. The discrepancy between the effects estimated on different quantiles can be interpreted in terms of the corresponding effect on the shape of the demand distribution. For example, at `tod` = 18 the effects f_1 are more spread out than at `tod` = 19 implying that, all else being equal, the variance of the demand is higher. Similarly, at `doy` ≈ 25 the effect f_3 is more positive for quantile $\tau = 0.9$ than for $\tau = 0.1$, while the positions are reversed at `doy` ≈ 120. Hence, the demand distribution is more skewed to the right in the first case. The linear coefficient of the lagged demand increases with τ hence, all else being equal, the demand variance one day ahead increases with the lagged demand.

So far we considered only the mean demand over the portfolio of customers, however the original data set contains demand for each of the n_c households. Average demand is typically much easier to forecast than individual demand, because the highly irregular consumption of individual customers is averaged out. However, the by-customer data should carry more information, hence it is interesting to verify whether it can be used to improve the quantile forecasts for the average demand. We capture part of the information contained in the by-customer demand data by considering a model which, for storage reasons, uses a functional summary of the full demand data, as explained in the following.

We consider the functional quantile model

$$\mu_\tau(\boldsymbol{x}_t) = \sum_{j=1}^{7} \beta_j \mathbb{I}(\mathrm{dow}_t = j) + f_1(\mathrm{tod}_t) + f_2(\mathrm{temp}_t) + f_3(\mathrm{doy}_t) + \int_0^1 f_4(p) G_{t-48}^{-1}(p) dp, \quad (11)$$

where $\mu_\tau(\boldsymbol{x}_t)$ is, as in (10), the τ-th quantile of the average demand, while we indicate with G_t the c.d.f. of the demand distribution across customers at time t and with $G_{t-48}^{-1}(p)$ the corresponding p-th quantile, at the same instant of the previous day. Function f_4 is a smooth effect along $p \in [0,1]$, constructed using a thin plate spline basis. Hence, the last term in (11) is a functional effect which takes as input the whole quantile function of the across-customers demand distribution at time $t - 48$, and integrates it over p using weight function $f_4(p)$. The aim is verifying whether the shape of the distribution of the individual demand at time $t - 48$ can be used to improve the forecast of the average demand at time t. The regression coefficient of $f_4(p)$ are unknown, but they can be estimated using the methods presented in Section 2.3. In particular, we have that

$$\int_0^1 f_4(p) G_{t-48}^{-1}(p) dp = \int_0^1 \sum_{k=1}^{K_4} b_k^4(p) \beta_k^4 G_{t-48}^{-1}(p) dp = \sum_{k=1}^{K_4} \left\{ \int_0^1 b_k^4(p) G_{t-48}^{-1}(p) dp \right\} \beta_k^4 = \sum_{k=1}^{K_4} \tilde{b}_k^4 \beta_k^4,$$

where transformed basis $\tilde{b}_1^4, \ldots, \tilde{b}_{K_4}^4$ implicitly depends on time t. Hence, once the transformed basis is obtained, estimating the coefficients of the functional term presents no extra difficulty, relative to standard smooth effects. The across-customers demand c.d.f. G_{t-48} is unknown, but we have a sample of size n_c from it, consisting of the by-customer demand data. Hence,
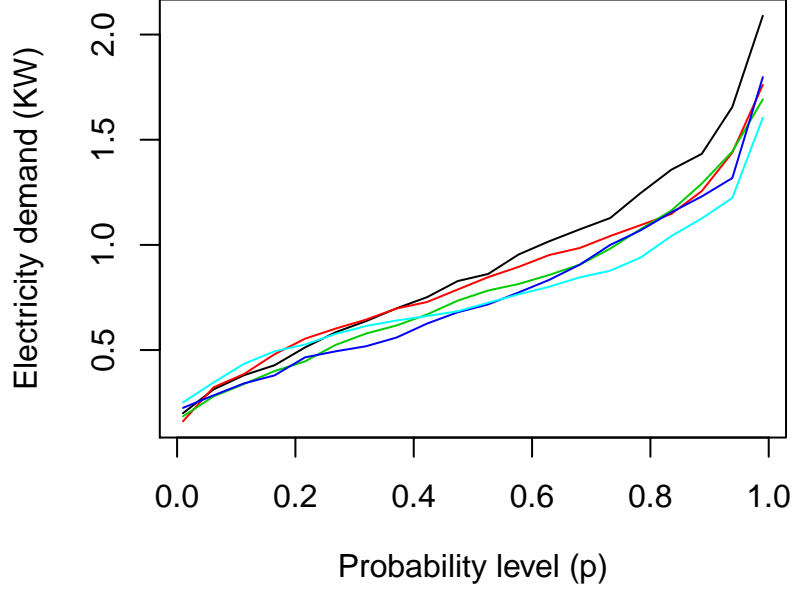
Figure 6: Estimates of the individual demand quantile function G_{t-48}^{-1}, for five values of t.

we can approximate the transformed basis using

$$\tilde{b}_k^4 = \int_0^1 b_k^4(p) G_{t-48}^{-1}(p) dp \approx \frac{1}{n_q} \sum_{l=1}^{n_q} b_k^4(p_l) \hat{G}_{t-48}^{-1}(p_l), \tag{12}$$

where $\hat{G}_{t-48}^{-1}(p_1), \ldots, \hat{G}_{t-48}^{-1}(p_{n_q})$ is a set of empirical quantiles of the n_c by-customer demand observations at time $t-48$, corresponding to probabily level $0 < p_1 < p_2 < \cdots < p_{n_q} < 1$.

To reduce the amount of data stored in the **qgam** package, we use $n_q = 20 < n_c$ quantiles equally spaced between $p_1 = 0.01$ and $p_{n_q} = 0.99$. The quantiles are stored in

```
R> qDem48 <- AUDem$qDem48
R> qDem48[1:3, c(1, 2, 5, 15, 19, 20)]

         1% 6.157895% 21.63158% 73.21053% 93.84211%       99%
[1,] 0.1995373 0.3128564 0.5125927  1.127776  1.655850 2.089360
[2,] 0.1595543 0.3368127 0.5569682  1.206294  1.587732 2.084527
[3,] 0.2091480 0.3571971 0.5652339  1.141480  1.632144 2.044097
```

where each row contains $\hat{G}_{t-48}^{-1}(p_1), \ldots, \hat{G}_{t-48}^{-1}(p_{n_q})$, for some t. Figure 6 shows five selected rows of qDem48, each line being an estimate of the quantile function G_{t-48}^{-1}, for a different t. To fit model (11) with **qgam**, we need also a matrix where each row of probLev contains the probability levels $p_1, p_2, \ldots, p_{n_q}$. This is computed as follows

```
R> probLev <- matrix(seq(0.01, 0.99, length.out = 20),
+    nrow = nrow(meanDem), ncol = 20, byrow = TRUE)
R> probLev[1:3, c(1, 2, 5, 15, 19, 20)]
```

```
      [,1]       [,2]      [,3]      [,4]       [,5] [,6]
[1,] 0.01 0.06157895 0.2163158 0.7321053 0.9384211 0.99
[2,] 0.01 0.06157895 0.2163158 0.7321053 0.9384211 0.99
[3,] 0.01 0.06157895 0.2163158 0.7321053 0.9384211 0.99
```

We add the matrices just defined to the learning and testing sets by doing

```
R> ntrain <- nrow(meanDemLearn)
R> ntest <- nrow(meanDemTest)
R> meanDemLearn$qDem48 <- qDem48[1:ntrain, ]
R> meanDemTest$qDem48  <- qDem48[-(1:ntrain), ]
R> meanDemLearn$probLev <- probLev[1:ntrain, ]
R> meanDemTest$probLev <- probLev[-(1:ntrain), ]
```

We are now ready to fit model (11) using

```
R> fitFunQ <- mqgamV(list(dem ~ dow + s(temp) + s(doy, bs = "cc") +
+    s(tod, k = 6) + s(probLev, by = qDem48), ~ s(doy, bs = "cc")),
+    qu = qusObj, data = meanDemLearn)
```

where we are using the `mgcViz::mqgamV` function, which is simply a shortcut for fitting the model using `mqgam` and then transforming it using `getViz`. Here the functional effect is constructed using `s(probLev, by = orDem48)`, which is the constructor typically used for by-variable or varying-coefficient effects. The corresponding transformed spline basis is constructed by `gam`, which performs the summation in (12) using the rows of the `probLev` and `orDem48` matrices.

We now have a look at the fitted functional smooth effects $f_4(p)$ using

```
R> plot(fitFunQ, select = 4) + labs(color = expression(tau))
```

which produces the plot in Figure 7. It is interesting to notice that the effects corresponding to low and high probabilities diverge, suggesting that the extremes of the quantile function $F_{t-48}^{-1}(p)$ have a stronger linear effect on the high average demand quantiles (e.g., $\mu_\tau(\boldsymbol{x})$ with $\tau = 0.9$) than on the low ones. However, the `plot.mgamViz` method does not include credible intervals when plotting the same effect for multiple quantiles, to avoid cluttering the plot. In Figure 8 we compare the effects for $\tau = 0.1$ and 0.9, and we include 95% credible intervals. This shows that the discrepancy between the effects is statistically stronger for $p \approx 1$, than for $p \approx 0$.

To check whether model (11) leads to a better fit than model (10), we can compare the values of the Bayesian information criterion (BIC) corresponding to each quantile by doing

```
R> round(rbind(QGAM = sapply(fitQ, BIC), FuncQGAM = sapply(fitFunQ, BIC)))
```

```
           0.1   0.3   0.5   0.7   0.9
QGAM     -8255 -9513 -9700 -9137 -6651
FuncQGAM -8458 -9617 -9698 -9152 -6680
```
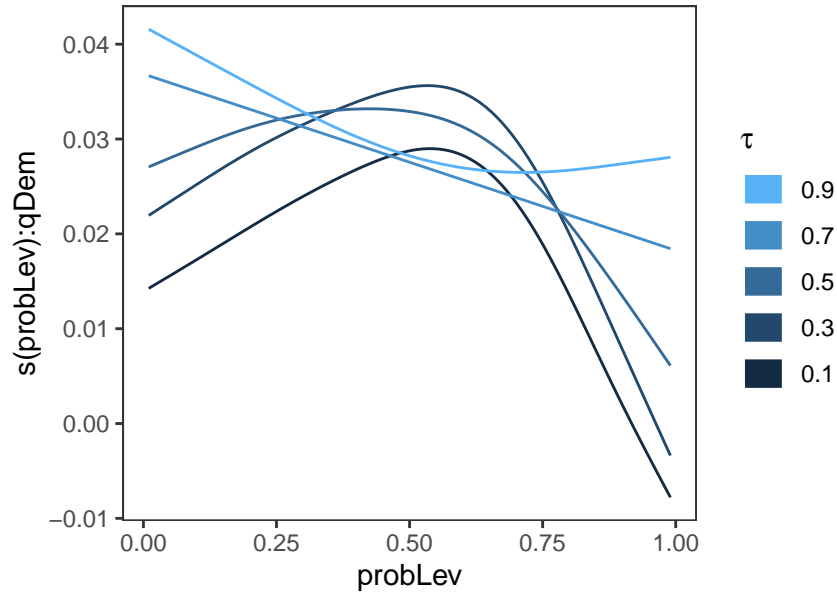
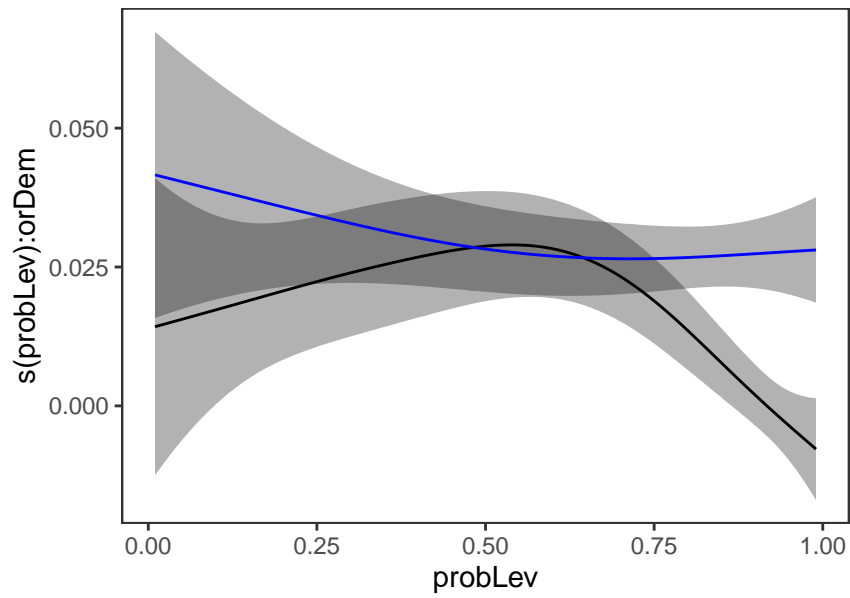Figure 7: Estimated effect $f_4(p)$ from model (11), for each quantile.

Figure 8: Estimated effect $f_4(p)$ from model (11) with 95% credible intervals, for quantile $\tau = 0.1$ (black) and 0.9 (blue).

BIC favours slightly model (11), but the simpler model is preferred for one of the quantiles. Instead the Akaike information criterion (AIC), which penalizes model complexity less strongly than BIC, favours model (11) for all quantiles

```
R> round(rbind(QGAM = sapply(fitQ, AIC), FuncQGAM = sapply(fitFunQ, AIC)))


            0.1   0.3   0.5   0.7   0.9
QGAM      -8416 -9679 -9867 -9303 -6813
FuncQGAM  -8637 -9800 -9881 -9325 -6858
```

We can also compare the predictive performance of the two models on the test set, which comprises the last three months of data set (1st of April to 30th of June, see Figure 4). For example, we can calculate the pinball loss for each model and quantile on such data as follows

```
R> predQ <- sapply(fitQ, predict, newdata = meanDemTest)
R> predFunQ <- sapply(fitFunQ, predict, newdata = meanDemTest)
R> round(rbind(QGAM = pinLoss(meanDemTest$dem, predQ, qusObj),
+     FuncQGAM = pinLoss(meanDemTest$dem, predFunQ, qusObj)), 1)


          0.1 0.3  0.5  0.7  0.9
QGAM      4.7 8.8 10.4 10.6 10.7
FuncQGAM  4.6 8.6 10.3 10.2 10.2
```

where `qgam::pinLoss` evaluates the pinball loss. The functional model does better on the test set, but it is important to point out that the size of the test set is rather limited, especially for the purpose of evaluating the performance of the extreme quantiles estimates.

Of course, model (11) could be improved further. For example, a call to

```
R> check(fitFunQ[[3]])
```

```
Theor. proportion of neg. resid.: 0.5   Actual proportion: 0.5155229
Integrated absolute bias |F(mu) - F(mu0)| = 0.02719397
Method: REML   Optimizer: outer newton
full convergence after 7 iterations.
Gradient range [-3.421018e-08,4.412375e-07]
(score -4857.326 & scale 1).
Hessian positive definite, eigenvalue range [0.5027064,3.678087].
Model rank =  39 / 39


Basis dimension (k) check: if edf is close too k' (maximum possible edf)
it might be worth increasing k.


                 k'  edf
s(temp)           9 8.21
s(doy)            8 7.65
s(tod)            5 4.69
s(probLev):qDem48 10 3.15
```

shows that, for $\tau = 0.5$, the effective degrees of freedom (`edf`) are quite close to their maximum possible value (`k'`) for the temperature effect. This means that the fit is using all the degrees of freedom available for that effect, hence we might want to consider increasing the number of basis functions by using `s(temp, k = 20)` in the model formula. However, one might be reluctant to increase the number of basis functions used to fit more extreme quantiles (e.g., $\tau = 0.9$), where the data is sparser. This highlights a disadvantage of using `mqgam`, rather than repeated calls to `qgam`, when fitting several QGAMs: the same model formula is used for all the quantiles. The model could also be improved by, for example, including the effect of lagged temperatures, often useful for capturing thermal inertia in buildings.

4. Conclusions

The **qgam** R package provides tools for building, fitting and checking quantile additive models. The main advantage of such models, relative to standard probabilistic GAMs, is that they do not make any parametric assumption on the distribution of the response variable. This is achieved by modelling and estimating the conditional quantiles directly, using the pinball loss. Model fitting is based on the fast calibrated Bayesian framework of Fasiolo *et al.* (2017), which aims at providing calibrated credible intervals, via IKL selection of the learning rate, and computational efficiency, by smoothing the pinball loss and by exploiting the marginal likelihood methods of Wood *et al.* (2016). Model building is handled by **mgcv**, hence the quantile GAMs built using **qgam** can include any of the effect types made available by **mgcv**, among others: cyclic, adaptive and tensor-product smooths, functional and random effects.

At the time of writing, **qgam** version `1.3.1` can handle data sets and models of moderate size. In particular, the current fitting methods require explicit formation of the $n \times d$ model matrix $\mathbf{X}$, which leads to high memory usage for large n. Wood, Goude, and Shaw (2015) proposed a GAM fitting framework where explicit formation of $\mathbf{X}$ is avoided by adopting block-oriented iterations for smoothing parameters selection and regression coefficients estimation. The `bam` function in **mgcv** implements such methods, as well as the marginal discretization approach of Wood, Li, Shaddick, and Augustin (2017), which allows it to fit models with up to 10^4 coefficients and 10^8 data points. Future work will aim at improving the scalability of the Bayesian QGAM fitting framework currently implemented in **qgam**, by exploiting the block-oriented discretized fitting methods provided by **mgcv** for the inner and intermediate iterations, and developing a similarly scalable version of the IKL minimization routine currently used for learning rate selection.

Acknowledgments

References

Bissiri PG, Holmes CC, Walker SG (2016). "A General Framework for Updating Belief

Distributions." *Journal of the Royal Statistical Society B*, **78**(5), 1103–1130. `doi:doi.org/10.1111/rssb.12158`.

Brent RP (2013). *Algorithms for Minimization without Derivatives*. Courier Corporation.

Brezger A, Kneib T, Lang S (2005). "**BayesX**: Analysing Bayesian Structured Additive Regression Models." *Journal of Statistical Software*, **14**(11). `doi:10.18637/jss.v014.i11`.

Eilers PH, Marx BD (1996). "Flexible Smoothing with B-splines and Penalties." *Statistical Science*, **11**(2), 89–102.

Fasiolo M, Goude Y, Nedellec R, Wood SN (2017). "Fast Calibrated Additive Quantile Regression." ArXiv:1707.03307 [stat.ME], URL `https://arxiv.org/abs/1707.03307`.

Fasiolo M, Nedellec R, Goude Y, Wood SN (2018). "Scalable Visualisation Methods for Modern Generalized Additive Models." ArXiv:1809.10632 [stat.ME], URL `https://arxiv.org/abs/1809.10632`.

Fasiolo M, Nedellec R, Goude Y, Wood SN (2019a). **mgcViz***: Visualisations for Generalized Additive Models*. R package version 0.1.4, URL `https://CRAN.R-project.org/package=mgcViz`.

Fasiolo M, Wood SN, Zaffran M, Goude Y, Nedellec R (2019b). **qgam***: Smooth Additive Quantile Regression Models*. R package version 1.3.0, URL `https://CRAN.R-project.org/package=qgam`.

Hastie T, Tibshirani R (1986). "Generalized Additive Models." *Statistical Science*, **1**(3), 297–318.

Hothorn T, Bühlmann P, Kneib T, Schmid M, Hofner B, Sobotka F, Scheipl F, Mayr A (2018). **mboost***: Model-Based Boosting*. R package version 2.9-1, URL `https://CRAN.R-project.org/package=mboost`.

Jones M (2008). "On a Class of Distributions with Simple Exponential Tails." *Statistica Sinica*, **18**(3), 1101–1110.

Jones M, Pewsey A (2009). "Sinh-arcsinh Distributions." *Biometrika*, **96**(4), 761–780. `doi:10.1093/biomet/asp054`.

Koenker R (2005). *Quantile Regression*. Cambridge University Press, Cambridge.

Koenker R (2013). **quantreg***: Quantile Regression*. **R** package version 5.38, URL `https://CRAN.R-project.org/package=quantreg`.

Lin CY, Bondell H, Zhang HH, Zou H (2013). "Variable Selection for Non-parametric Quantile Regression via Smoothing Spline Analysis of Variance." *Stat*, **2**(1), 255–268. `doi:10.1002/sta4.33`.

Rigby RA, Stasinopoulos DM (2005). "Generalized Additive Models for Location, Scale and Shape." *Journal of the Royal Statistical Society C*, **54**(3), 507–554. `doi:10.1111/j.1467-9876.2005.00510.x`.

Ripley B, Venables B, Bates DM, Hornik K, Gebhardt A, Firth D (2013). **MASS***: Support Functions and Datasets for Venables and Ripley's MASS*. R package version 7.3-51.4, URL `https://CRAN.R-project.org/package=MASS`.

Silverman BW (1985). "Some Aspects of the Spline Smoothing Approach to Non-parametric Regression Curve Fitting." *Journal of the Royal Statistical Society B*, **47**(1), 1–52. `doi:10.1111/j.2517-6161.1985.tb01327.x`.

Syring N, Martin R (2015). "Scaling the Gibbs Posterior Credible Regions." ArXiv:1509.00922 [stat.ME], URL `https://arxiv.org/pdf/1509.00922`.

Umlauf N, Klein N, Zeileis A, Koehler M, Simon T, Stadlmann S (2018). **bamlss***: Bayesian Additive Models for Location Scale and Shape (and Beyond)*. R package version 1.0-1, URL `https://CRAN.R-project.org/package=bamlss`.

Waldmann E, Kneib T, Yue YR, Lang S, Flexeder C (2013). "Bayesian Semiparametric Additive Quantile Regression." *Statistical Modelling*, **13**(3), 223–252. `doi:10.1177/1471082X13480650`.

Wickham H, Chang W, Henry L, Pedersen TL, Takahashi K, Wilke C, Woo K, Yutani H (2019). **ggplot2***: Create Elegant Data Visualisations Using the Grammar of Graphics*. R package version 3.2.0, URL `https://CRAN.R-project.org/package=ggplot2`.

Wood SN (2017). *Generalized Additive Models: an Introduction with R.* Chapman & Hall/CRC, Boca Raton.

Wood SN (2019). **mgcv***: Mixed GAM Computation Vehicle with Automatic Smoothness Estimation*. R package version 1.8.28, URL `https://CRAN.R-project.org/package=mgcv`.

Wood SN, Goude Y, Shaw S (2015). "Generalized Additive Models for Large Data Sets." *Journal of the Royal Statistical Society C*, **64**(1), 139–155. `doi:10.1111/rssc.12068`.

Wood SN, Li Z, Shaddick G, Augustin NH (2017). "Generalized Additive Models for Gigadata: Modeling the UK Black Smoke Network Daily Data." *Journal of the American Statistical Association*, **112**(519), 1199–1210. `doi:10.1080/01621459.2016.1195744`.

Wood SN, Pya N, Säfken B (2016). "Smoothing Parameter and Model Selection for General Smooth Models." *Journal of the American Statistical Association*, **111**(516), 1548–1575. `doi:10.1080/01621459.2016.1180986`.

Yee TW (2010). "The **VGAM** Package for Categorical Data Analysis." *Journal of Statistical Software*, **32**(10), 1–34. `doi:10.18637/jss.v032.i10`.

A. Choosing the loss bandwidth manually

By default the `qgam` function determines the ELF loss bandwidth $h(\boldsymbol{x})$ automatically, but such choice can be overwritten by specifying the `err` argument, which is set to `NULL` by default. Here we provide some guidelines for users who wish to choose this parameter manually. In particular, the examples in Section A.1 are meant to illustrate how parameter `err` affects the statistical and computational performance of the fitting methods provided by **qgam**.

A.1. Illustrating the effect of the loss bandwidth and some diagnostics

To clarify the interpretation of parameter `err`, let us consider a univariate context where we aim at estimating the τ-th quantile of response y and there are no covariates. Define $\epsilon = |F(\mu^*) - F(\mu_0)|$, where F is the c.d.f. of y, μ_0 is the true quantile at probability level τ and μ^* is the minimizer of $\mathbb{E}\{\tilde{\rho}_\tau(y - \mu)\}$. Given that μ_0 is the minimizer of the expected pinball loss (see (1)), ϵ is the asymptotic absolute bias induced by using a smoothed loss, with bandwidth $h = \lambda\sigma$. In Appendix A.2 we prove that

$$\epsilon = |F(\mu^*) - F(\mu_0)| \leq 2\log 2\, h \sup_y f(y) = \frac{2\log 2}{\sqrt{2\pi\kappa}}\, h, \tag{13}$$

where f is the p.d.f. of y and the last equality hold if y is Gaussian with variance κ. Via argument `err`, `qgam` allows users to set the maximum tolerable bias ϵ, which is then used to determine the loss bandwidth by doing

$$h = \epsilon\frac{\sqrt{2\pi\kappa}}{2\log 2}. \tag{14}$$

Here κ is estimated via the location-scale GAM models, and can depend on $\boldsymbol{x}$ as usual. Parameter $h = h(\boldsymbol{x})$, obtained via (14), is then used to determine λ and $\sigma(\boldsymbol{x})$ as in Section 2.3. Of course ϵ (`err` in the software) is an approximate bound, because the response is not normally distributed in practice.

In **qgam** we let users choose ϵ, rather than λ, because it does not depend on the scale of the response y. Indeed, considering the interpretation of ϵ, reasonable values of this parameter fall in $(0, 1)$ (`qgam` checks only the positivity of `err`). To illustrate how `err` influences the accuracy of the quantile estimates and the computational performance of `qgam`, we consider a simple data set simulated as follows

```
R> set.seed(5523)
R> x <- seq(-3, 3, length.out = 1e3)
R> X <- data.frame("x" = x, "y" = x + x^2 + rgamma(1e3, 4, 1))
```

Hence $y_i = x_i + x_i^2 + z_i$, where $z_i \sim \text{Gamma}(4, 1)$. Assuming that we want to estimate quantiles 0.05, 0.5 and 0.95, the following code estimates them using different values of `err`

```
R> fitGrid <- lapply(c(0.01, 0.05, 0.1, 0.3, 0.5),
+    function(.errVal){
+    mqgam(y ~ s(x), data = X, qu = c(0.05, 0.5, 0.95), err = .errVal)
+    })
```
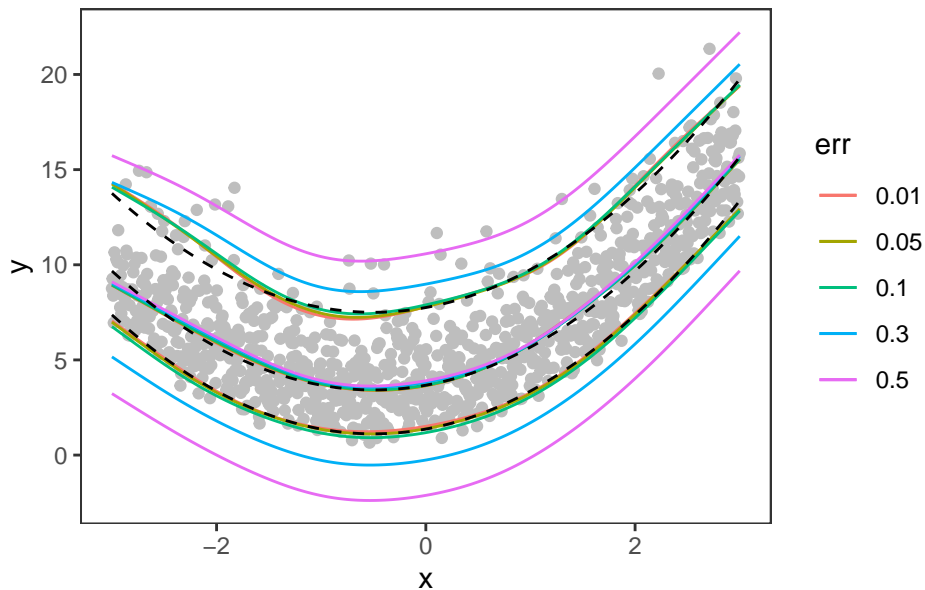
Figure 9: Quantile fits for $\tau = 0.05, 0.5$ and 0.95 for different values of `err`. The dashed lines are the true quantiles.

The resulting quantile fits are shown in Figure 9. It is clear that the bias is positive for $\tau = 0.95$, negative for $\tau = 0.05$ and that its absolute value increases with `err`, becoming clearly visible for `err` > 0.1. However, increasing `err` affects the median estimate only slightly. Heuristically, this is because the bias induced by the smoothed loss is greater when the response distribution is more asymmetric around the quantile of interest.

One might conclude that the loss smoothness should be kept as small as possible, but this leads to computational and numerical stability issues. In fact, decreasing `err` tends to slow down computation, and in this example using a very small tolerance increases the computing time dramatically

```
R> system.time(fitBigErr <- qgam(y ~ s(x), data = X,
+    qu = 0.95, err = 0.05))[[3]]
```

```
Estimating learning rate. Each dot corresponds to a loss evaluation.
qu = 0.95.............done
```

```
[1] 0.345
```

```
R> system.time(fitSmallErr <- qgam(y ~ s(x), data = X,
+    qu = 0.95, err = 0.001))[[3]]
```

```
Estimating learning rate. Each dot corresponds to a loss evaluation.
qu = 0.95....................done
```
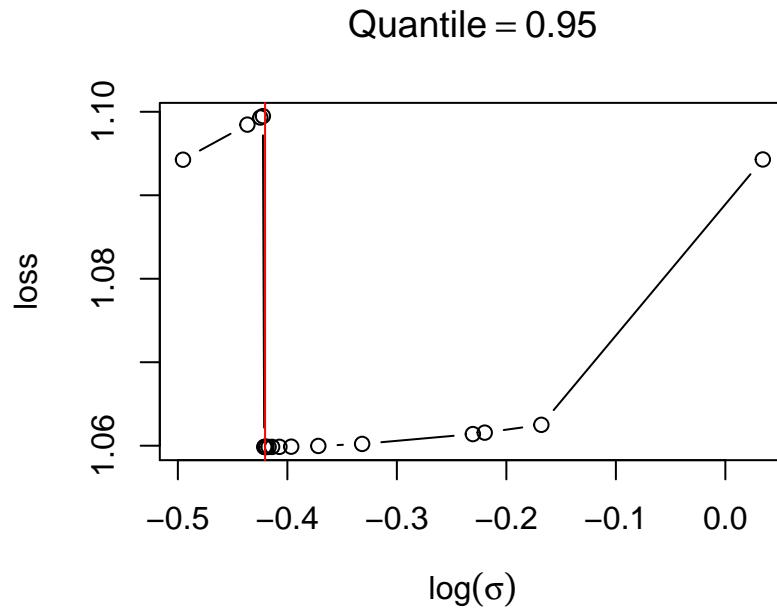
```
[1] 35.236
```

Figure 10:   Plot produced by `check.learnFast`, showing the estimated IKL loss at several values of $\log \sigma_0$, when `err` $= 0.001$. The vertical jump in the loss indicates numerical problems.

More importantly, using a very low value of `err` can lead to numerical problems. In fact, the following code

```
R> check(fitSmallErr$calibr, sel = 2)
```

produces the plot in Figure 10, which shows how the estimated IKL loss minimized by the outer iteration looks like. The dots indicate the points at which the loss has been evaluated during the outer Brent optimization, and the vertical red line indicates the value of $\log \sigma_0$ which minimizes the loss. Here the loss seems to be discontinuous, which is due to numerical instabilities. In fact, the analogous plot for `fitBigErr` (not shown) looks smooth and convex. To produce the diagnostic plot in Figure 10 we used the generic `qgam::check` function, which dispatched to the `check.learnFast` method. This is because the output of `qgam` contains the results of a call to `tuneLearnFast` in the `$calibr` slot. The call to `check` generates two plots and here we are showing only the second one by choosing `sel = 2`.

We can use `check` to have also an estimate of the bias attributable to the smoothed loss and information regarding the convergence of the smoothing parameter estimation routine. This is achieved by doing

```
R> check(fitBigErr)
```

```
Theor. proportion of neg. resid.: 0.95    Actual proportion: 0.952
Integrated absolute bias |F(mu) - F(mu0)| = 0.004319621
Method: REML   Optimizer: outer newton
```

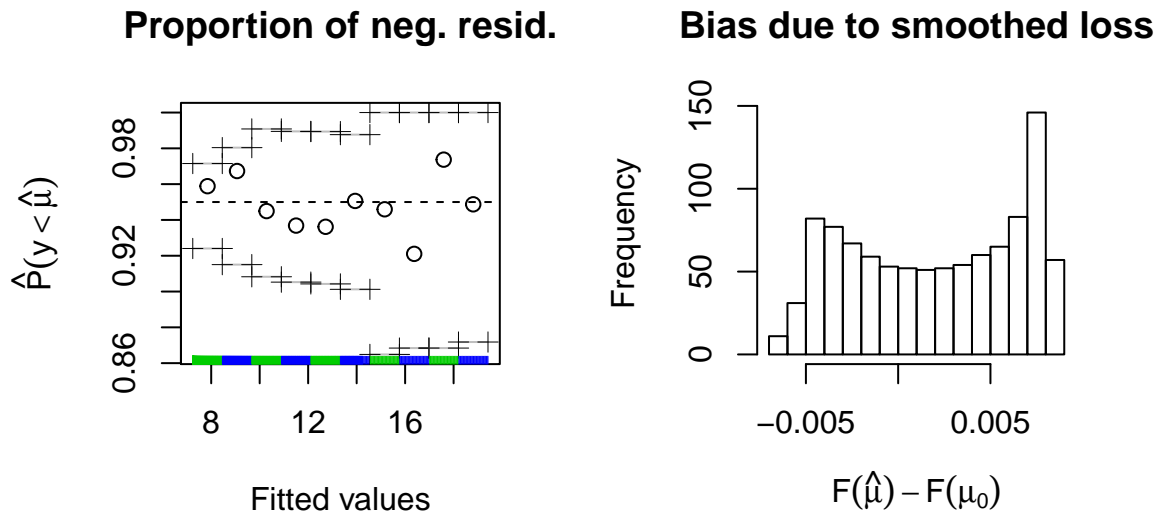**Proportion of neg. resid.**   **Bias due to smoothed loss**

Figure 11: Diagnostic plots produced by `check.qgam`. Left: observed proportion of negative residuals in ten bins, with 95% reference intervals. Right: histogram of estimated bias attributable to using a smooth loss.

```
full convergence after 5 iterations.
Gradient range [2.626207e-05,2.626207e-05]
(score 3187.994 & scale 1).
Hessian positive definite, eigenvalue range [1.874658,1.874658].
Model rank =  10 / 10

Basis dimension (k) check: if edf is close too k' (maximum possible edf)
it might be worth increasing k.

     k'  edf
s(x)  9 5.27
```

which produces the preceding text output and the diagnostic plots shown in Figure 11. The text states that the intermediate optimization for selecting $\boldsymbol{\gamma}$ achieved full convergence in few iterations. It reports the range of the entries of the LAML (7) gradient w.r.t. $\boldsymbol{\gamma}$ at convergence, which we expect to be small, and confirms that the Hessian is positive definite, so that LAML is (at least locally) convex. The text also states that the model has full rank, hence all the regression coefficients $\boldsymbol{\beta}$ are identifiable. The first two lines of the printed text are concerned with the bias due to the use of the smooth ELF loss. The first line says that nearly 95% of the observations fall below the fitted quantile, which is the percentage we expect for $\tau = 0.95$. The second line reports an estimate of the absolute bias $|F\{\mu^*(\boldsymbol{x}_i)\} - F\{\mu_0(\boldsymbol{x}_i)\}|$, averaged over the observed $\boldsymbol{x}_i$'s. This is much lower than the bound `err = 0.05` used to fit this model. Figure 11 gives more details, with the left plot showing the proportion of negative residuals in a sequence of bins, ordered according to increasing value of $\mu_\tau(\boldsymbol{x})$. The crosses define 95% reference intervals for the proportions, obtained using binomial quantiles. The plot on the

right is a histogram of all the estimated values of the bias $|F\{\mu^*(\boldsymbol{x}_i)\} - F(\boldsymbol{x}_i)|$, and it shows that the bias is low for all observations. Hence, the text and visual output of `check.qgam` confirm that the LAML optimization achieved full convergence and that we should not be too concerned about the bias. The last few lines of the text output indicate that the effective degrees of freedom (`edf`) used to model the effect of `x` are well below the degrees of freedom available for this smooth effect (`k'`). This suggests that the number of basis functions used to model the effect of `x` was sufficiently large, while `edf` $\approx$ `k'` would indicate that the fit is using all the available degrees of freedom, and that we might want to increase the number of basis functions used.

The results presented in this section and our practical experience on loss smoothness selection and convergence checking with **qgam** can be summarized in the following set of suggestions:

- the automatic procedure for selecting the loss smoothness generally offers a good compromise between statistical bias, variance and numerical stability;

- the old default (used in **qgam** versions lower than `1.3.0`) was `err = 0.05`, which generally does not lead to unacceptable levels of bias;

- if the calibration loss plotted by `check(fittedQGAM$learn)` is irregular, or the text printed by `check(fittedQGAM)` does not confirm that `full convergence` was achieved, try to increase `err`;

- if you have to increase `err` to 0.2 or higher to avoid convergence issues, then there might be something wrong with your model (for example, it is missing an important effect);

- the bias attributable to the adoption of a smoothed ELF loss can be estimated using `check(fittedQGAM)`;

- you might get messages saying that `outer Newton did not converge fully` during estimation. These are generated by `mgcv::gam` during LAML maximization, and should not be problematic as long as the calibration loss is smooth, which you can check using `check(fittedQGAM$calibr)`, and `check(fittedQGAM)` states that `full convergence` was achieved;

- in preliminary studies, that is when you are exploring different model structures and you are not yet interested in getting the most accurate estimates, do not decrease `err` too much as it considerably slows down computation;

- setting `err` too low is generally not a good idea. In fact, `err` is an approximate upper bound on the bias, the latter being generally much lower than this parameter suggests, and it is arguably better to have a small amount of bias than numerical problems. As stated above, the default loss smoothness selection procedure, used by default in `qgam` and `mqgam`, typically provides stable fits and a near optimal bias-variance tradeoff.

A.2. Derivation of the asymptotic bias

To simplify the notation, indicate $\tilde{p}_\tau(y - \mu)$ with $\tilde{p}_\tau(y)$. We start from

$$F(\mu^*) - F(\mu_0) = \int \mathbb{1}(y \le \mu^*) f(y) dy - \tau = \int \left\{ \frac{\partial \log \tilde{p}_\tau(y)}{\partial \mu} \bigg|_{\mu=\mu^*} - \frac{\partial \rho_\tau(y)}{\partial \mu} \bigg|_{\mu=\mu^*} \right\} f(y) \, dy,$$

where $f(y)$ is the p.d.f. of y and we used the fact that $\int \partial \log \tilde{p}_\tau(y)/\partial\mu|_{\mu=\mu^*} f(y) dy = 0$, by definition of μ^*. We proceed to bound the r.h.s. from above. For any μ, simple manipulations lead to

$$\int \left\{ \frac{\partial \log \tilde{p}_\tau(y)}{\partial \mu} - \frac{\partial \rho_\tau(y)}{\partial \mu} \right\} f(y) \, dy = \int \left\{ \Phi(y|\mu, \lambda\sigma) - \mathbb{1}(y > \mu) \right\} f(y) \, dy, \qquad (15)$$

where $\mathbb{1}(\cdot)$ is the indicator function and $\Phi(y|\mu, \lambda\sigma)$ is the c.d.f. of a logistic r.v. with mean μ and scale $\lambda\sigma$. Then we have

$$\begin{aligned}
|F(\mu^*) - F(\mu_0)| &\le \int \left| \Phi(y|\mu^*, \lambda\sigma) - \mathbb{1}(y > \mu^*) \right| \sup_y f(y) \, dy \\
&= 2 \sup_y f(y) \int_{-\infty}^{\mu^*} \Phi(y|\mu^*, \lambda\sigma) \, dy,
\end{aligned}$$

where the second equality holds due to the symmetry of the integrand around μ^*. Using the substitution $z = (y - \mu^*)/\lambda\sigma$ leads to

$$\begin{aligned}
|F(\mu^*) - F(\mu_0)| &\le 2\lambda\sigma \sup_y f(y) \int_{-\infty}^{0} \frac{1}{1 + e^{-z}} \, dz \\
&= 2 \log(2) \lambda\sigma \sup_y f(y).
\end{aligned}$$

Notice that the r.h.s. of (15) makes it clear that, if $f(y)$ is symmetric around μ^*, then $|F(\mu^*) - F(\mu_0)| = 0$ and there is no bias.

Affiliation:

Matteo Fasiolo
School of Mathematics
University of Bristol
University Walk
BS8 1TW Bristol, United Kingdom
E-mail: matteo.fasiolo@gmail.com
URL: http://www.bristol.ac.uk/maths/people/matteo-fasiolo/overview.html