Learning Quantile Functions for Temporal Point Processes with Recurrent Neural Splines

Souhaib Ben Taieb
Department of Computer Science
University of Mons
souhaib.bentaieb@umons.ac.be

Abstract

We can build flexible predictive models for rich continuous-time event data by combining the framework of temporal point processes (TPP) with (recurrent) neural networks. We propose a new neural parametrization for TPPs based on the conditional quantile function. Specifically, we use a flexible monotonic rational-quadratic spline to learn a smooth continuous quantile function. Conditioning on historical events is achieved through a recurrent neural network. This novel parametrization provides a flexible yet tractable TPP model with multiple advantages, such as analytical sampling and closed-form expressions for quantiles and prediction intervals. While neural TPP models are often trained using maximum likelihood estimation, we consider the more robust continuous ranked probability score (CRPS). We additionally derive a closed-form expression for the CRPS of our model. Finally, we demonstrate that the proposed model achieves state-of-the-art performance in standard prediction tasks on both synthetic and real-world event data.

1 Introduction

Discrete events happening at irregular intervals, also called continuous-time event data, occur across many scientific fields and applications. Examples include electronic health records, consumer purchases, financial transactions and server logs. Figure 1 shows an example of ten event sequences for a real-world dataset. It is of great interest in all of these areas to learn models which can capture the influence of past events on future ones to carry out subsequent prediction, recommendation or intervention.

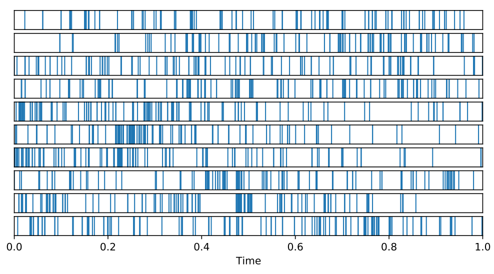

Figure 1: Ten sequences of event times.

There has been significant amount of prior work on modeling such data (Hawkes, 1971) under the framework of temporal point processes (TPP) (D. J. Daley, 2003). However, the strong parametric assumptions and restrictions of these models - which limit their flexibility for modeling complex real-world dynamics - have motivated the development of neural TPP models (Shchur et al., 2021). By combining the TPP framework with deep neural networks, neural TPP models provide a flexible framework for modeling continuous-time event data and have become the current state-of-the-art for predictive modeling with such data.

Recurrent neural TPP models are autoregressive models which characterize the conditional distribution of the next arrival time in a TPP. Various representations of this distribution have been proposed in the literature, including parametric forms for the intensity function (Mei and Eisner, 2017), the cumulative intensity function (Omi et al., 2019), and the probability density function (Shchur et al., 2020a). While a TPP can be equivalently characterized by one of these functions, choosing which function to parametrize and how to train the associated model are important design choices (Shchur et al., 2021).

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151. Copyright 2022 by the author(s).

We make the following main contributions:

1. We propose a new neural parametrization for TPPs based on the conditional quantile function. Specifically, we learn a smooth continuous quantile function represented by a monotonic rational-quadratic spline (RQS) (Durkan et al., 2019). Conditioning on historical events is achieved through a recurrent neural network (RNN). This novel parametrization leads to a flexible yet tractable neural TPP model with many advantages over the existing density-based neural TPP models. In fact, quantiles are optimal predictions for a large class of loss functions that arise in many real-world prediction problems (Gneiting, 2011). Also, a closed-form expression for the quantile function enables analytical sampling. Furthermore, our model has closed-form expressions for quantiles, prediction intervals as well as the expectation. This allows us to avoid expensive sampling-based approximations. In fact, analytical sampling is not possible with the neural model proposed by Omi et al. (2019) and the mixture model of Shchur et al. (2020a) does not have a closed-form quantile function.

2. While traditional density-based neural TPP models are trained using maximum likelihood estimation (MLE), we train our model by minimizing the more robust continuous ranked probability score (CRPS) (Gneiting et al., 2007). Importantly, we derive an analytical expression for the CRPS of our model which enables more efficient parameter optimization.

3. We show through a range of experiments that the proposed model provides state-of-the-art results on both synthetic and real-world event sequence datasets.

2 Background

Temporal point processes (TPPs) are stochastic processes whose realization is a sequence of N arrival times $\mathcal{S} = \{t_1, t_2, \ldots, t_N\}$ in some time interval $[0, T]$, where $0 < t_1 < t_2 < \cdots < t_N \leq T$, and the number of events N is random (D. J. Daley, 2003). TPPs can be equivalently represented using inter-arrival times $\tau_i = t_i - t_{i-1}$ for $i = 1, \ldots, N+1$ with $t_0 = 0$ and $t_{N+1} = T$. We will use the two representations interchangeably throughout the paper. Examples of TPPs include Poisson processes, where events are independent from each other (Aalen et al., 2008), and Hawkes processes, where the occurrence of events depends on the past history of the process, with specific dynamics such as self-excitation (Hawkes, 1971).

It is of particular interest to many applications to predict the arrival time of future events from historical observations. This requires modelling the dependency between the next arrival time t and the history $\mathcal{H}_t = \{t_j : t_j < t\}$. TPPs can be uniquely characterized by the (strictly positive) conditional intensity function $\lambda^*(t) = \lambda(t|\mathcal{H}_t)$ which gives the arrival *rate* of new events conditional on the history $\mathcal{H}_t$[1].

TPP model parametrization. While intensity parametrization is popular for TPP modelling (Zhou et al., 2013; Zhang et al., 2019; Zuo et al., 2020), TPPs can also be characterized by the *distribution* of the next arrival time given $\mathcal{H}_t$, which we denote $P^*(t)$. We can represent such distribution using different functions (Shchur et al., 2021), including the cumulative distribution function (CDF) $F^*(t)$, the probability density function (PDF) $f^*(t)$, the survival function $S^*(t) = 1 - F^*(t)$, the intensity function $\lambda^*(t) = f^*(t)/S^*(t)$ or the cumulative intensity function $\Lambda^*(t) = \int_0^t \lambda^*(s)\,\mathrm{d}s$. One can define a TPP model by picking a parametric form for any of these functions, provided that the chosen parametrization defines a valid probability distribution. While any parametric form can be used, neural network parametrizations provide more flexibility and allow to capture more complex real-world dynamics and dependencies (Du et al., 2016; Shchur et al., 2020a; Mei and Eisner, 2017; Omi et al., 2019) .

TPP model training. Given a valid parametrization of one of the above functions, the most common approach to train neural TPP models is via MLE, or equivalently by negative log-likelihood (NLL) minimization. For example, given a parametrization of the intensity function, $\lambda^*_{\boldsymbol{\theta}}(t)$, the PDF $f^*_{\boldsymbol{\theta}}(t)$ or the CDF $F^*_{\boldsymbol{\theta}}(t)$, and a sequence of events $\mathcal{S}$, the NLL objective is given by (Rasmussen, 2018)

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{S}) = -\sum_{i=1}^{N} \log \lambda^*_{\boldsymbol{\theta}}(t_i) + \int_0^T \lambda^*_{\boldsymbol{\theta}}(s) ds \tag{1}$$

$$= -\sum_{i=1}^{N} \log f^*_{\boldsymbol{\theta}}(\tau_i) - \log\left(1 - F^*_{\boldsymbol{\theta}}(T)\right), \tag{2}$$

where $\boldsymbol{\theta} \in \boldsymbol{\Theta} \subseteq \mathbb{R}^d$ is a vector of parameters.

Choosing which function to parametrize is an important design choice especially for model training. For example, if $\lambda^*_{\boldsymbol{\theta}}(t)$ does not have an analytical integral, computing $\mathcal{L}(\boldsymbol{\theta}; \mathcal{S})$ in (1) will require numerical integration which can be computationally expensive and can lead to poor accuracy. If instead we parametrize $\Lambda^*(t)$, $\mathcal{L}(\boldsymbol{\theta}; \mathcal{S})$ is simpler to compute using differentiation (Omi et al., 2019).

[1]The * symbol denotes the dependence on the history (Rasmussen, 2018).

3 Quantile function parametrization for TPPs

We present a new neural TPP model based on the parametrization of the conditional quantile function. Let $F^*(\tau)$ be the CDF of the real-valued (continuous) random variable associated to the next inter-arrival time τ given the history $\mathcal{H}_t$. Assuming $F^*(\tau)$ is strictly increasing, the *quantile function* is the inverse of $F^*(\tau)$, given by $Q^*(\alpha) = {F^*}^{-1}(\alpha)$ where $\alpha \in [0, 1)$. In other words, $Q^* : [0, 1) \to \mathbb{R}_+$ returns the value τ such that $F^*(\tau) = \alpha$. With a quantile function, it is straightforward to generate samples using inversion sampling, i.e. by evaluating the quantile function at uniformly distributed values. Furthermore, a quantile function can be used to produce prediction intervals with a specified probability coverage. Finally, quantile functions of continuous random variables are equivariant under monotonic transformations.

While the quantile function is a useful representation of a random variable, finding a good neural parametrization is not trivial. In fact, a useful and valid parametric form for $Q^*(\alpha)$ should (i) be flexible enough to approximate any distribution, (ii) define continuous and strictly increasing functions, and (iii) have a closed-form expression for computational efficiency. We propose to represent $Q^*(\alpha)$ using a *monotonic rational-quadratic splines* (RQS) (Gregory and Delbourgo, 1982; Durkan et al., 2019), which satisfy the previous three properties. After briefly presenting RQS, we describe its neural parametrization in Section 3.2. Tail modelling and model training are discussed in Sections 3.3 and 3.4, respectively. Finally, we derive a closed-form expression for the CRPS of our model in Section 3.5.

3.1 Monotonic rational-quadratic splines

A RQS is specified using K different monotonically-increasing rational-quadratic functions with boundaries defined by $K + 1$ coordinates $\{(x^{(k)}, y^{(k)})\}_{k=0}^{K}$, known as *knots*, and $K + 1$ strictly positive values $\{\delta^{(k)}\}_{k=0}^{K}$ for the derivatives at the knots (Durkan et al., 2019).

Let $\Delta y^{(k)} = y^{(k+1)} - y^{(k)}$, $\Delta x^{(k)} = x^{(k+1)} - x^{(k)}$ and $s^{(k)} = \Delta y^{(k)}/\Delta x^{(k)}$ for $k = 0, 1, \ldots, K - 1$. Then, the expression of the kth rational quadratic for $\alpha \in [x^{(k)}, x^{(k+1)}]$, or equivalently, for $\xi \in [0, 1]$ with $\xi = \xi_k(\alpha) = (\alpha - x^{(k)})/\Delta x^{(k)}$, is given by

$$r_k(\xi) = c_1^{(k)} + \frac{c_2^{(k)}\xi^2 + c_3^{(k)}\xi}{-c_4^{(k)}\xi^2 + c_4^{(k)}\xi + c_5^{(k)}}, \quad (3)$$

where

$$c_1^{(k)} = y^{(k)}, c_2^{(k)} = \Delta y^{(k)}(s^{(k)} - \delta^{(k)}), \quad (4)$$

$$c_3^{(k)} = \Delta y^{(k)}\delta^{(k)}, c_4^{(k)} = \left[\delta^{(k+1)} + \delta^{(k)} - 2s^{(k)}\right], \quad (5)$$

$$c_5^{(k)} = s^{(k)}. \quad (6)$$

Note that if $c_4^{(k)} = 0$ and $c_2^{(k)} \neq 0$, $r_k(\xi)$ reduces to a quadratic function, while if $c_4^{(k)} = 0$ and $c_2^{(k)} = 0$, $r_k(\xi)$ is linear in ξ.

We can analytically invert a rational-quadratic function by finding the root α of the quadratic equation $\tau = r_k(\xi_k(\alpha))$. It can be shown that the unique solution is given by

$$\alpha = \Delta x^{(k)}\xi + x^{(k)}, \quad \xi = 2c/(-b + \sqrt{b^2 - 4ac}), \quad (7)$$

where

$$a = c_2^{(k)} + c_4^{(k)}(\tau - c_1^{(k)}), \quad (8)$$

$$b = c_3^{(k)} - c_4^{(k)}(\tau - c_1^{(k)}), \quad (9)$$

$$c = -c_5^{(k)}(\tau - c_1^{(k)}). \quad (10)$$

Finally, the derivative of the rational-quadratic function in (3) is given by

$$\frac{\mathrm{d}r_k(\xi)}{\mathrm{d}\xi} \quad (11)$$
$$= \frac{(s^{(k)})^2[\delta^{(k+1)}\xi^2 + 2s^{(k)}\xi(1-\xi) + \delta^{(k)}(1-\xi)^2]}{[s^{(k)} + c_4^{(k)}\xi(1-\xi)]^2}.$$

3.2 Neural quantile TPP model

Building on Gasthaus et al. (2019), we consider a family of conditional quantile functions $q_{\boldsymbol{\phi}}(\alpha|\boldsymbol{h})$ indexed by the vector $\boldsymbol{\phi}$ where $q_{\boldsymbol{\phi}}(\alpha|\boldsymbol{h}) = q_{\boldsymbol{\theta}(\boldsymbol{h};\phi)}(\alpha)$, and $q_{\boldsymbol{\theta}}(\alpha)$ is a family of quantile functions indexed by the vector $\boldsymbol{\theta}$. In other words, the value of the conditioning variable $\boldsymbol{h}$ is mapped to the parameters $\boldsymbol{\theta}$ with the mapping $\boldsymbol{\theta}(\boldsymbol{h}; \boldsymbol{\phi})$ parametrized by $\boldsymbol{\phi}$.

History embedding. While, in a TPP, the inter-arrival time of the next event can depend on all the historical events, a common assumption in neural TPP models is that event history $\mathcal{H}_t$ can be compactly represented with a vector $\boldsymbol{h} \in \mathbb{R}^H$ (Shchur et al., 2021). We make the same assumption and propose to use autoregressive neural models with recurrent encoders as in Du et al. (2016) and Shchur et al. (2020a). Specifically, an initial hidden state is sequentially updated with each observed event, and the final hidden state is used as the history embedding. Examples of update functions include the RNN, GRU or LSTM update equations (Xiao et al., 2017b; Du et al., 2016). We refer the reader to Shchur et al. (2021) for a review on history embedding in TPPs.

RQS-based quantile functions. We let the vector $\boldsymbol{\theta}$, with $3(K+1)$ components, represent $\{(x^{(k)}, y^{(k)}), \delta^{(k)}\}_{k=0}^{K}$, i.e. the knots and the derivatives at the knots for K different rational quadratics. Such vector defines a RQS on the interval $[x^{(0)}, x^{(K)}]$, as explained in Section 3.1. Then, if $(x^{(0)}, y^{(0)}) = (0,0)$, we can define a family of quantile functions, $q_{\boldsymbol{\theta}} : [0, x^{(K)}] \to [0, y^{(K)}]$, indexed by the parameter $\boldsymbol{\theta}$, where

$$q_{\boldsymbol{\theta}}(\alpha) = \sum_{k=0}^{K-1} \mathbb{1}\{\alpha \in [x^{(k)}, x^{(k+1)}]\}\ r_k\left(\xi_k(\alpha)\right), \quad (12)$$

where r_k is defined in (3).

Gregory and Delbourgo (1982) showed that a RQS as given by (12) defines a monotonic, continuously-differentiable function which passes through the knots $\{(x^{(k)}, y^{(k)})\}_{k=0}^{K}$, with the derivatives at the knots given by $\{\delta^{(k)}\}_{k=0}^{K}$. In fact, we can check that $q_{\boldsymbol{\theta}}(x^{(k)}) = y^{(k)}$ and $\frac{\mathrm{d}q_{\boldsymbol{\theta}}(x^{(k)})}{\mathrm{d}\alpha} = \delta^{(k)}$. Therefore, provided that $\delta^{(k)} > 0$ in (3) for all k, expression (12) defines a valid quantile function on $[x^{(0)}, x^{(K)}]$. Finally, to evaluate the quantile function at location α, we need to find the bin in which α lies. Since the bins are sorted, this can be done efficiently using binary search (Durkan et al., 2019).

Obtaining the parameters. Building on Durkan et al. (2019), to obtain the parameters $\boldsymbol{\theta}$ from the history embedding $\boldsymbol{h}$, we first compute an unconstrained parameter vector $\tilde{\boldsymbol{\theta}}$ of length $3K+1$ as an affine function of $\boldsymbol{h}$, i.e. $\tilde{\boldsymbol{\theta}} = \tilde{\boldsymbol{W}}\boldsymbol{h} + \tilde{\boldsymbol{b}}$, with $\tilde{\boldsymbol{W}} \in \mathbb{R}^{(3K+1)\times H}$ and $\tilde{\boldsymbol{b}} \in \mathbb{R}^{3K+1}$. Then, the vector is partitioned as $\tilde{\boldsymbol{\theta}} = [\tilde{\boldsymbol{\theta}}^{\Delta x} || \tilde{\boldsymbol{\theta}}^{\Delta y} || \tilde{\boldsymbol{\theta}}^{d}]$ where $\tilde{\boldsymbol{\theta}}^{\Delta x} \in \mathbb{R}^K$, $\tilde{\boldsymbol{\theta}}^{\Delta y} \in \mathbb{R}^K$, and $\tilde{\boldsymbol{\theta}}^{d} \in \mathbb{R}^{K+1}$. To obtain the widths and heights of the bins, we split the unit interval in K intervals by computing $\boldsymbol{\theta}^{\Delta x} = \mathrm{softmax}(\tilde{\boldsymbol{\theta}}^{\Delta x})$ and $\boldsymbol{\theta}^{\Delta y} = \mathrm{softmax}(\tilde{\boldsymbol{\theta}}^{\Delta y})$. To obtain the $K+1$ knots $\{(x^{(k)}, y^{(k)})\}_{k=0}^{K}$, it suffices to compute a cumulative sums of the K bin widths $\boldsymbol{\theta}^{\Delta x}$ and heights $\boldsymbol{\theta}^{\Delta y}$, starting at $x^{(0)} = 0$ and $y^{(0)} = 0$, respectively. Then, given $x^{(K)}$ and $y^{(K)}$, a mapping is applied from $[0,1]^2$ to $[0, x^{(K)}] \times [0, y^{(K)}]$. Strictly positive derivatives are obtained with $\boldsymbol{\theta}^{d} = \mathrm{softplus}(\tilde{\boldsymbol{\theta}}^{d})$, from which we obtain the values of the derivatives at each knot $\{\delta^{(k)}\}_{k=0}^{K}$.

Useful properties. Representing a quantile function with a RQS has many advantages. In fact, RQS are more flexible than quadratic splines, and allow to match arbitrary values and derivatives of a function at two boundary knots. As pointed out by Durkan et al. (2019), with enough bins, a differentiable monotonic spline defined on a given interval will approximate any differentiable monotonic function on that interval. Furthermore, having a closed-form expression for the quantile function enables analytical sampling, as well as efficient computation of quantiles and prediction intervals for any specified probability coverage. Also, since the quantile function is integrable, the expectation has a closed-form expression (see Appendix C.2).

In addition, each rational-quadratic function defining the RQS is analytically invertible and has a closed-form derivative. Therefore, given a quantile function $q_{\boldsymbol{\theta}}(\cdot)$ as defined in (12), closed-form expressions for the CDF and PDF at location τ can be computed using

$$F_{\boldsymbol{\theta}}(\tau) = q_{\boldsymbol{\theta}}^{-1}(\tau), \text{ and } f_{\boldsymbol{\theta}}(\tau) = \left[q_{\boldsymbol{\theta}}^{-1}\right]'(\tau). \quad (13)$$

Specifically, after identifying the bin in which τ lies, we can compute $q_{\boldsymbol{\theta}}^{-1}(\tau)$ or its derivative by inverting or computing the derivatives of the corresponding rational-quadratric using (7) and (11), respectively.

A closed-form expression for the CDF allows to compute the probability integral transform (PIT) for calibration diagnostics, and to efficiently compute the CRPS (see Section 3.5). Furthermore, unlike the piecewise linear splines considered by Gasthaus et al. (2019), RQS define smooth continuous quantile functions with continuous (but not smooth) PDF. Finally, while we do not train our model using NLL, it is worth mentioning that a closed-form PDF enables more efficient MLE model training.

3.3 Tail modelling

If $x^{(K)} < 1$ and $y^{(K)} < \infty$ in (12) then the property that $F_{\boldsymbol{\theta}}(\tau) = q_{\boldsymbol{\theta}}^{-1}(\tau) \to 1$ for $\tau \to \infty$ will not be satisfied. This is one of the properties that define a valid CDF for a continuous random variable [2]. We can extrapolate beyond $x^{(K)}$ and the data range using the quantile function of a known parametric distribution. Extreme value theory suggests that in many cases the distribution tails follow a Pareto or exponential distribution.

Specifically, let $q_{\boldsymbol{\gamma}} : [x^{(K)}, 1) \to [y^{(k)}, \infty)$ be a strictly increasing parametric function with parameters $\boldsymbol{\gamma} \in \boldsymbol{\Gamma}$, which satisfies

$$q_{\boldsymbol{\gamma}}(x^{(K)}) = y^{(K)} \text{ and } \frac{\mathrm{d}q_{\boldsymbol{\gamma}}(x^{(K)})}{\mathrm{d}\alpha} = \delta^{(K)}, \quad (14)$$

i.e. $q_{\boldsymbol{\gamma}}$ matches the value and the derivative of the spline at $x^{(K)}$. Then, we can define a new family of (continuously differentiable) quantile functions

[2] Rasmussen (2018) pointed out that, if $F_{\boldsymbol{\theta}}(\tau) \to x^{(K)}$ for $x^{(K)} < 1$, this means that there is a probability $x^{(K)}$ to see new events in the process, and with probability $1 - x^{(K)}$ there are no more events, and the process terminates.

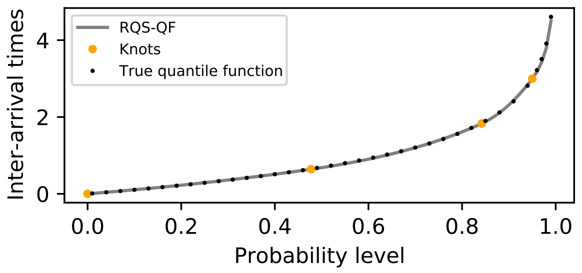

Figure 2: Example of estimated quantile function.

$q_{\bar{\boldsymbol{\theta}}} : [0, 1) \to \mathbb{R}^+$, given by

$$q_{\bar{\boldsymbol{\theta}}}(\alpha) = \begin{cases} q_{\boldsymbol{\theta}}(\alpha), & \text{if } x^{(0)} \leq \alpha \leq x^{(K)} \\ q_{\boldsymbol{\gamma}}(\alpha), & \text{if } x^{(K)} \leq \alpha < 1 \end{cases} \tag{15}$$

where $\bar{\boldsymbol{\theta}} = [\boldsymbol{\theta}^T, \boldsymbol{\gamma}^T]^T$.

To retain all the useful properties of RQS, it is important to pick parametric forms of $q_{\boldsymbol{\gamma}}$ with a closed-form inverse and derivative. In this work, for simplicity, we consider the exponential distribution which depends on a single parameter $\lambda > 0$, called rate.

To satisfy the constraints in (14), we can show that we must have $\lambda = 1/((1 - x^{(K)})\delta^{(K)})$, i.e. the quantile function for $x^{(K)} \leq \alpha < 1$ is given by

$$q_{\boldsymbol{\gamma}}(\alpha) = y^{(K)} - \delta^{(K)}(1 - x^{(K)})\log\left(\frac{1 - \alpha}{1 - x^{(K)}}\right). \tag{16}$$

See Appendix C.1 for more details on the above expression. Finally, if $q_{\bar{\boldsymbol{\theta}}}$ is given by (15), we can now verify that $F_{\bar{\boldsymbol{\theta}}}(\tau) = q_{\bar{\boldsymbol{\theta}}}^{-1}(\tau) \to 1$ for $\tau \to \infty$. Figure 2 gives an example of quantile function represented by our model in (15) with $K = 3$ and $x^{(K)} = 0.95$.

In the following, to simplify notations, $q_{\boldsymbol{\theta}}$ will refer to the quantile function defined in (15).

3.4 Model training

Probabilistic models are often trained using proper scoring rules since they make quoting the true distribution as the predictive distribution an optimal strategy in expectation (Gneiting and Katzfuss, 2014). A scoring rule is a summary measure of the quality of a probabilistic forecast which summarizes both calibration and sharpness (Gneiting et al., 2007). Sharpness relates to the concentration of the probabilistic forecast, while calibration concerns its statistical consistency with the data. Given a probabilistic forecast G and an observation τ, a scoring rule computes a score $S(G, \tau)$. It is considered *proper* if for all possible distributions G, $\mathbb{E}_P[S(P, \tau)] \leq \mathbb{E}_P[S(G, \tau)]$ where P is the true distribution of τ. It is *strictly proper* when the equality holds if and only if $P = G$.

The most common approach to train neural TPP models is via NLL minimization, which corresponds to the logarithmic scoring rule

$$\text{LogS}(f_{\bar{\boldsymbol{\theta}}}, \tau) = -\log(f_{\bar{\boldsymbol{\theta}}}(\tau)).$$

Since our model parametrization is based on the quantile function $q_{\bar{\boldsymbol{\theta}}}$, we propose to use the continuous ranked probability score (CRPS), a proper scoring rule (Gneiting et al., 2007) which can be defined as

$$\text{CRPS}(q_{\bar{\boldsymbol{\theta}}}, \tau) = \int_0^1 \text{QS}_\alpha\left(q_{\bar{\boldsymbol{\theta}}}(\alpha), \tau\right) \,\mathrm{d}\alpha\,, \tag{17}$$

where

$$\begin{aligned} &\text{QS}_\alpha\left(q_{\bar{\boldsymbol{\theta}}}(\alpha), \tau\right) \\ &\quad = 2\left(\mathbb{1}\{\tau \leq q_{\bar{\boldsymbol{\theta}}}(\alpha)\} - \alpha\right)\left(q_{\bar{\boldsymbol{\theta}}}(\alpha) - \tau\right) \end{aligned} \tag{18}$$

is the so-called quantile score (Gneiting, 2011).

Both the CRPS and the LogS are proper scoring rules but with some conceptual differences. The CRPS is called *sensitive to distance*, which means it rewards predictive distributions that place mass close to the realizing outcome. The LogS is *local* in the sense that it only considers the model's predicted probabilities of the events that have happened. In the literature, the choice between these two contrasting features has been ultimately subjective. See Gebetsberger et al. (2018) for an empirical comparison of the two scores.

However, a general concern with the LogS is that it takes large values for low-probability events, meaning that it is sensitive to outliers, leading Gneiting et al. (2007) to conclude that the CRPS is an attractive alternative. Furthermore, the quantile score decomposition of the CRPS can be useful to compare the accuracy of different models for different probability levels and can provide insight into the strengths and deficiencies of a model.

Model training with the CRPS is computationally more demanding than training with the NLL. Furthermore, while the CRPS of common parametric distributions has a closed form expression (Jordan et al., 2019), it is often not available for general distributions. Therefore, one has to numerically approximate the integral in (17) which can be inefficient when training large neural network models. Fortunately, in the next section, we derive a closed-form expression for the CRPS of our model. See Appendix B for more details on computational time.

3.5 Computation of the CRPS

Let $q_{\bar{\boldsymbol{\theta}}}$ denote the quantile function in (15), $\tau \in \mathbb{R}_+$ an observation, and $\tilde{\alpha} = q_{\bar{\boldsymbol{\theta}}}^{-1}(\tau)$. If $\tilde{\alpha} \in [0, x^{(K)}]$, we let

$l \in \{0, 1, \ldots, K-1\}$ denote the bin where $\tilde{\alpha}$ lies, i.e. $\tilde{\alpha} \in [x^{(l)}, x^{(l+1)}]$ and $\tau \in [y^{(l)}, y^{(l+1)}]$.

The CRPS given in (17) can be decomposed as

$$\begin{aligned}&\mathrm{CRPS}(q_{\bar{\boldsymbol{\theta}}}, \tau)\\ &= \tau(\tilde{\alpha} - \frac{1}{2}) - \underbrace{\int_0^1 \alpha\ q_{\bar{\boldsymbol{\theta}}}(\alpha)\,\mathrm{d}\alpha}_{A} + \underbrace{\int_{\tilde{\alpha}}^1 q_{\bar{\boldsymbol{\theta}}}(\alpha)\,\mathrm{d}\alpha}_{B}, \end{aligned} \tag{19}$$

where

$$\begin{aligned}A = \sum_{k=0}^{K-1} \Bigg[[\Delta x^{(k)}]^2 \int_0^1 \xi\ r_k(\xi)\,\mathrm{d}\xi \\ + x^{(k)} \Delta x^{(k)} \int_0^1 r_k(\xi)\,\mathrm{d}\xi \Bigg] + \int_{x^{(K)}}^1 \alpha\ q_{\boldsymbol{\gamma}}(\alpha)\,\mathrm{d}\alpha\,,\end{aligned} \tag{20}$$

and

$$B = \begin{cases} B_1(l) + B_2 + B_3, & \text{if } 0 \le \tilde{\alpha} \le x^{(K-1)} \\ B_1(K-1) + B_3, & \text{if } x^{(K-1)} \le \tilde{\alpha} \le x^{(K)} \\ B_3. & \text{if } x^{(K)} \le \tilde{\alpha} < 1 \end{cases} \tag{21}$$

with

$$B_1(l) = \Delta x^{(l)} \int_{\frac{\tilde{\alpha} - x^{(l)}}{\Delta x^{(l)}}}^1 r_l(\xi)\,\mathrm{d}\xi\,, \tag{22}$$

$$B_2 = \sum_{k=l+1}^{K-1} \Delta x^{(k)} \int_0^1 r_k(\xi)\,\mathrm{d}\xi\,, \tag{23}$$

$$B_3 = \int_{x^{(K)}}^1 q_{\boldsymbol{\gamma}}(\alpha)\,\mathrm{d}\alpha\,. \tag{24}$$

Appendix A.1 gives full details of the above expressions. As can be seen in expressions (20) and (21), a closed-form expression for the CRPS in (19) relies on closed-form expressions for the following four integrals $\int r_k(\xi)\,\mathrm{d}\xi$, $\int \xi\ r_k(\xi)\,\mathrm{d}\xi$, $\int q_{\boldsymbol{\gamma}}(\alpha)\,\mathrm{d}\alpha$, and $\int \alpha q_{\boldsymbol{\gamma}}(\alpha)\,\mathrm{d}\alpha$. In the following, we show that these four integrals have closed-form expressions. We omit the superscript $^{(k)}$ to simplify notations.

Integrals involving $r_k(\cdot)$. If $c_4 = 0$, $r_k(\cdot)$ in (3) reduces to a quadratic function, and the integrals can be easily computed. We have

$$\begin{aligned}\int_z^1 r_k(\xi)\,\mathrm{d}\xi = y^{(k)}(1-z) + \frac{\Delta x^{(k)} \delta^{(k)}}{2}(1-z^2) \\ + \frac{\Delta x^{(k)}(s^{(k)} - \delta^{(k)})}{3}(1-z^3),\end{aligned} \tag{25}$$

and

$$\begin{aligned}&\int_0^1 \xi\ r_k(\xi)\,\mathrm{d}\xi \\ &= \frac{y^{(k)}}{2} + \frac{\Delta x^{(k)} \delta^{(k)}}{3} + \frac{\Delta x^{(k)}(s^{(k)} - \delta^{(k)})}{4}.\end{aligned} \tag{26}$$

If $c_4 \neq 0$, we have

$$\begin{aligned}&\int_z^1 r_k(\xi)\,\mathrm{d}\xi \\ &= \left(c_1 - \frac{c_2}{c_4}\right)[1-z] + \int_z^1 \frac{A\xi + B}{a\xi^2 + b\xi + c}\,\mathrm{d}\xi\,,\end{aligned} \tag{27}$$

where $A = c_3 + c_2$, $B = \frac{c_2 c_5}{c_4}$, $a = -c_4$, $b = c_4$ and $c = c_5$. We also have

$$\begin{aligned}\int_0^1 \xi\ r_k(\xi)\,\mathrm{d}\xi = \frac{1}{2}(c_1 - \frac{c_2}{c_4}) - \frac{(c_2 + c_3)}{c_4} \\ + \int_0^1 \frac{A\xi + B}{a\xi^2 + b\xi + c}\,\mathrm{d}\xi\,,\end{aligned} \tag{28}$$

where $A = (\frac{c_2 c_5}{c_4} + c_3 + c_2)$, $B = \frac{(c_3 + c_2)c_5}{c_4}$, $a = -c_4$, $b = c_4$ and $c = c_5$.

The RHS of (27) and (28) involve integrals of linear/quadratic rational functions, which have closed-form expressions. While these integrals look complicated, their computation simply requires solving the quadratic equation $a\xi^2 + b\xi + c = 0$. Appendix A.2 gives detailed expressions for these integrals.

Integrals involving $q_{\boldsymbol{\gamma}}(\cdot)$. A closed-form expression for these integrals will depend on the parametric form of $q_{\boldsymbol{\gamma}}$. For the exponential distribution where $q_{\boldsymbol{\gamma}}$ is given by (16), we can show that

$$\begin{aligned}&\int_{\tilde{\alpha}}^1 q_{\boldsymbol{\gamma}}(\alpha)\,\mathrm{d}\alpha \\ &= \frac{(1-\tilde{\alpha})}{\lambda}\left(y^{(K)}\lambda + 1 - \log\left(\frac{1-\tilde{\alpha}}{1-x^{(K)}}\right)\right),\end{aligned} \tag{29}$$

and

$$\begin{aligned}&\int_{x^{(K)}}^1 \alpha\ q_{\boldsymbol{\gamma}}(\alpha)\,\mathrm{d}\alpha \\ &= -\frac{\left(x^{(\mathrm{K})} - 1\right)\left(\left(2x^{(\mathrm{K})} + 2\right) y^{(K)}\lambda + x^{(\mathrm{K})} + 3\right)}{4\lambda},\end{aligned} \tag{30}$$

where $\lambda = 1/((1 - x^{(K)})\delta^{(K)})$. More details are given in Appendix C.1.

4 Related work

Neural TPPs. There has been a significant amount of work on neural TPP modelling. Shchur et al. (2021) provides a useful review of recent developments. Du et al. (2016) showed that we can build flexible TPP models using event embedding through recurrent neural networks. To improve training efficiency, new parametrization of recurrent neural TPP models with closed-form likelihoods have been proposed by Omi et al. (2019) and Shchur et al. (2020a). Zhang et al.

(2019) and Zuo et al. (2020) proposed non-reccurent TPP models based on self-attention mechanisms. Focusing on faster sampling, Shchur et al. (2020b) proposed non-reccurent TPP models based on triangular maps. Our work contributes to this field of research by proposing a new parametrization for neural TPP models which can be combined with ideas developed in the previously cited papers. Another line of work has focused on alternatives to MLE for training TPP models. Examples include adverserial learning (Xiao et al., 2017a, 2018), noise constrative estimation (Mei et al., 2020; Guo et al., 2018) and reinforcement learning (Upadhyay et al., 2018). While this line of research is orthogonal to our contribution, we also propose an alternative to MLE based on the CRPS.

Quantile function modelling. While not widely used in probabilistic modelling, quantile function estimation has been considered in many applications (Gilchrist, 2000; Koenker et al., 2013). The closely related problem of quantile regression has been extensively studied with applications in many domains (Koenker, 2017; Taylor, 2000; Wen et al., 2017). Unlike methods based on quantile function estimation, quantile regression methods often suffer from the quantile crossing problem, and cannot always be easily extended to non i.i.d. settings (Koenker and Xiao, 2006).

MLE, which is equivalent to minimizing the logarithmic scoring rule, is the most popular approach to train probabilistic models. The CRPS is more often used for probabilistic forecast evaluation rather than model training. Exceptions include Avati et al. (2020), who trained a survival model using the CRPS, and Duan et al. (2019), who proposed a gradient boosting model which can be trained using the CRPS. Ostrovski et al. (2018) applied quantile function estimation for image modeling using the CRPS but their parametrization is not constrained to be strictly increasing.

Our method is related to Gasthaus et al. (2019), who parametrized a quantile function with piecewise linear splines trained using the CRPS. However, the parametric form of their quantile function is not smooth and induces a discontinuous PDF. Furthermore, they focused on probabilistic forecasting for (regular) time series data.

Monotonic rational-quadratic splines have been recently proposed as nonlinear transformation with better properties for normalizing flow models (Durkan et al., 2019). Except Hutson (2001) who used RQS to fit an (unconditional) quantile function for a given sample of data, we are unaware of methods based on RQS to fit a conditional quantile function using neural models.

5 Experiments

We evaluate our new TPP parametrization on the task of event time prediction. Specifically, the task is to predict the distribution of the time until the next event τ_i given the history $\mathcal{H}_{t_i}$. We use five simulated datasets from different processes, as described in Omi et al. (2019): *Hawkes 1 and 2* (Hawkes, 1971), *Self-correcting* (Isham and Westcott, 1979), *Renewal* (Cox, 1967) and *Poisson*. We also consider twelve real-world datasets: *Taxi* (customer pickupts), *Wikipedia* (article edits), *Reddit*, *Reddit-C*, *Reddit-S* (comments), *Yelp*, *Yelp-A*, *Yelp-M* (check-ins to restaurants), *PUBG* (online gaming), *LastFM* (music playback), *Twitter* (tweets) and *MOOC* (online courses). Each dataset consists of multiple sequences of event times, and has been used for (neural) TPP modelling (Shchur et al., 2021, 2020a). See Appendix D for more details.

Setup. We compare our model, denoted as RQS-QF, with three baselines: (1) the Exponential model (constant intensity/exponential distribution) (Upadhyay et al., 2018), (2) the RMTPP model (exponential intensity/Gompertz distribution) (Du et al., 2016), and (3) the LogNormMix model (flexible intensity/a mixture of log-normal with $K = 64$ components) (Shchur et al., 2020a).

For a fair comparison, we use the same setup as Shchur et al. (2020a). All models use the same RNN architecture where the size of the RNN hidden vector is $H = 64$ and the batch size is 64. For the baselines, the L_2 regularization hyperparameter is selected in the set $\{0, 10^{-2}, 10^{-3}\}$. For our model, we select the number of bins K in the set $\{1, 2, 3, 5, 8, 10, 15\}$ and use $x^{(K)} = 0.95$ for the (upper) tail knot. As can be seen in Table 6 in Appendix F, our model is robust to the choice of K.

For each dataset, we partitioned the sequences into training/validation/test sequences (60%, 20%, 20%). Our model and the baselines are trained by minimizing the CRPS and the NLL, respectively. For all models, we use the Adam optimizer (Kingma and Ba, 2015) and a learning rate of 10^{-3}. We use the validation set for hyperparameters tuning including early stopping. We pick the hyperparameter configuration that achieves the smallest validation error, averaged over three random initializations.

We evaluate the different models using multiple metrics computed on a large sample from the estimated predictive distributions for each holdout sequence. To quantify the quantile prediction accuracy, we compute the *quantile score* (QS) as defined in (18) for probability levels ranging from 1% to 99%. We report the mean quantile score averaged over all the levels (QSm), as

		NLL	QSm	MACE	QS50	QS90	IS50	IS90	SMAPE
Poisson	RQS-QF	—	**0.496**	**0.011**	**0.686**	**0.463**	**2.232**	**3.966**	**0.783**
	LogNormMix	**0.989**	**0.496**	0.011	**0.686**	**0.462**	**2.232**	**3.968**	**0.784**
	RMTPP	**0.989**	**0.496**	**0.011**	**0.686**	**0.462**	**2.231**	**3.971**	**0.783**
	Exponential	**0.989**	**0.496**	**0.011**	**0.686**	**0.463**	**2.232**	**3.969**	**0.783**
Renewal	RQS-QF	—	**0.802**	0.011	**0.921**	**1.229**	**3.461**	**11.640**	**1.083**
	LogNormMix	**0.260**	**0.802**	**0.009**	**0.921**	**1.230**	**3.461**	**11.657**	**1.083**
	RMTPP	0.993	0.911	0.219	1.112	**1.237**	3.922	11.809	1.240
	Exponential	0.981	0.905	0.217	1.104	**1.234**	3.897	11.802	1.236
Hawkes1	RQS-QF	—	**0.715**	**0.011**	**0.887**	**0.940**	**3.176**	**8.176**	**0.901**
	LogNormMix	**0.513**	**0.723**	0.014	**0.891**	**0.968**	**3.203**	8.482	**0.904**
	RMTPP	0.735	0.759	0.106	0.964	**0.949**	3.335	8.525	0.949
	Exponential	0.726	0.804	0.095	0.964	1.140	3.531	10.220	1.007
Self-correcting	RQS-QF	—	0.344	**0.010**	0.496	0.232	1.544	2.436	**0.584**
	LogNormMix	0.784	0.351	0.013	0.509	0.230	1.580	2.293	0.593
	RMTPP	**0.775**	**0.340**	**0.010**	**0.494**	**0.225**	**1.533**	**2.241**	**0.583**
	Exponential	0.935	0.415	0.073	0.600	0.316	1.834	3.202	0.700
Hawkes2	RQS-QF	—	**0.783**	**0.010**	**0.967**	**1.058**	**3.482**	**9.007**	**1.166**
	LogNormMix	**0.016**	**0.793**	0.011	**0.969**	1.091	**3.528**	9.292	1.176
	RMTPP	0.688	0.870	0.192	1.110	**1.068**	3.820	9.713	1.250
	Exponential	0.684	0.901	0.185	1.092	1.261	3.964	11.164	1.330

Table 1: Various metrics computed on the test sequences for the synthetic datasets.

well as the 50-th and the 90-th quantile score (QS50 and QS90, respectively). To quantify quantile calibration, we compute the *mean absolute (quantile) calibration error* (MACE). We also evaluate the accuracy of (equal-tailed) prediction intervals centered at the median for target coverage probabilities ranging from 2% to 98% using the *interval score* (IS). The interval score is a proper scoring function that rewards calibrated and sharp prediction intervals (Winkler, 1972). We report the interval score for a 50% and a 90% prediction interval (IS50 and IS90, respectively). We further report the *interval width* (IW) as a measure of sharpness. Finally, to evaluate point prediction accuracy, we compute the Symmetric Mean Absolute Percentage Error (SMAPE) with the median as point forecast. While a lower value is better for all metrics, sharpness (as measured by IW) should be minimized subject to calibration. Appendix E.3 provides precise definitions of all metrics used. Open-source code to reproduce all the experiments is available on GitHub[3].

Results. Tables 1 and 2 summarize the results for synthetic and real-world datasets, respectively. Tables with standard errors are given in Appendix F.

As expected, all models are able to capture the exponential distribution of the Poisson process. As a toy example, Figure 2 shows that our model is able to recover the quantile function of the exponential distribution associated to the Poisson process. For the renewal and Hawkes processes, the more flexible models (RQS-QF, LogNormMix) achieve better performance than the simpler baselines. The results on the synthetic datasets show that our model is able to accurately estimate the conditional distribution for a variety of TPPs.

For the real-world datasets, we can see that the more flexible models (RQS-QF, LogNormMix) dominates the simple baselines (RMTPP, Exponential) for almost all metrics. This suggests that the higher model capacity of neural TPP methods is useful for rich real-world event sequence data. Compared with the strongest baseline (LogNormMix), our model consistently achieves either lower or comparable quantile and interval scores. In terms of model calibration, our model has either lower or comparable MACE than the best baselines on ten out of twelve datasets. This shows both the flexibility of our model as well as the benefit of CRPS model training. Having accurate and calibrated quantiles and prediction intervals is essential for many real-world prediction problems.

6 Future work and conclusions

We proposed a new neural parametrization for TPPs where a smooth continuous quantile function is estimated by minimizing the CRPS of a recurrent neural spline. Unlike density-based neural TPP models,

[3]`https://github.com/bsouhaib/qf-tpp`

		NLL	QSm	MACE	QS50	QS90	IS50	IS90	SMAPE
Yelp A	RQS-QF	—	**0.436**	**0.059**	0.592	**0.443**	**2.074**	**4.054**	0.861
	LogNormMix	**0.660**	0.446	0.061	**0.589**	0.502	2.109	4.349	0.858
	RMTPP	0.693	0.447	0.067	0.602	0.466	2.111	4.234	0.859
	Exponential	0.683	0.453	0.066	0.592	0.515	2.118	4.851	**0.848**
Yelp M	RQS-QF	—	**0.261**	**0.048**	**0.351**	**0.279**	**1.252**	**2.682**	**0.900**
	LogNormMix	**−0.181**	0.277	0.050	0.358	0.334	1.336	2.984	0.913
	RMTPP	−0.111	0.270	0.066	0.363	**0.283**	1.293	**2.619**	**0.901**
	Exponential	−0.120	0.300	0.061	0.369	0.401	1.400	3.999	0.926
PUBG	RQS-QF	—	**0.214**	**0.035**	**0.290**	**0.220**	**0.982**	1.950	0.805
	LogNormMix	**−1.095**	0.234	0.038	0.308	0.260	1.070	2.219	0.877
	RMTPP	−0.096	0.215	0.042	0.292	**0.219**	**0.984**	**1.926**	**0.802**
	Exponential	−0.096	0.235	0.041	0.310	0.263	1.074	2.260	0.878
Taxi	RQS-QF	—	**0.128**	0.040	**0.172**	0.131	**0.623**	1.307	**0.805**
	LogNormMix	**−0.568**	0.129	0.043	**0.174**	0.133	**0.635**	**1.267**	0.814
	RMTPP	**−0.563**	**0.128**	0.040	**0.173**	**0.128**	**0.624**	**1.227**	**0.805**
	Exponential	**−0.563**	0.137	**0.038**	0.178	0.156	0.656	1.554	0.819
Yelp	RQS-QF	—	**1.894**	**0.021**	**2.586**	**1.886**	**9.644**	**18.463**	0.961
	LogNormMix	**1.620**	4.095	0.026	2.659	2.047	10.021	19.664	0.979
	RMTPP	1.960	**1.908**	0.047	**2.595**	**1.897**	**9.704**	18.922	**0.950**
	Exponential	1.961	2.007	0.045	2.675	2.151	10.122	21.360	0.970
Reddit-S	RQS-QF	—	**0.011**	0.011	**0.015**	**0.011**	**0.056**	**0.103**	0.782
	LogNormMix	**−3.207**	0.011	0.031	0.016	0.011	0.057	0.110	0.801
	RMTPP	−3.005	**0.011**	**0.010**	**0.015**	**0.011**	**0.056**	**0.103**	**0.780**
	Exponential	−3.005	0.012	0.012	0.016	0.014	0.061	0.136	0.830
MOOC	RQS-QF	—	**6.604**	**0.072**	**6.873**	**12.050**	**27.869**	**124.777**	**1.315**
	LogNormMix	**−1.764**	**6.604**	0.083	**6.875**	**12.217**	**27.892**	**126.385**	1.370
	RMTPP	2.912	9.140	0.414	10.812	12.848	39.205	131.462	1.867
	Exponential	2.891	9.061	0.415	10.627	13.002	38.843	132.194	1.877
Reddit-C	RQS-QF	—	**0.044**	0.046	**0.057**	**0.050**	**0.864**	**2.129**	0.811
	LogNormMix	**−2.451**	0.045	0.052	**0.058**	0.053	0.897	2.272	0.818
	RMTPP	−2.321	**0.044**	0.047	**0.058**	**0.049**	**0.849**	**2.201**	**0.804**
	Exponential	−2.318	0.048	**0.038**	0.061	0.061	**0.887**	**2.243**	0.872
Twitter	RQS-QF	—	**0.517**	0.095	**0.637**	**0.699**	**2.939**	**6.847**	**1.013**
	LogNormMix	**−0.054**	**0.517**	**0.092**	**0.634**	**0.705**	**2.924**	**6.952**	**1.012**
	RMTPP	0.528	0.572	0.181	0.730	**0.710**	3.080	7.862	1.196
	Exponential	0.527	0.564	0.179	0.716	**0.714**	3.054	7.684	1.204
Reddit	RQS-QF	—	9.417	0.058	12.467	**10.890**	41.946	**93.385**	1.152
	LogNormMix	**2.940**	**9.369**	**0.053**	**12.368**	**10.971**	**41.647**	**94.204**	**1.148**
	RMTPP	3.612	10.060	0.149	13.428	11.030	44.567	99.713	1.195
	Exponential	3.607	10.032	0.149	13.376	11.129	44.383	99.757	1.197
LastFM	RQS-QF	—	**0.373**	0.039	**0.441**	**0.544**	**7.646**	**24.723**	**0.786**
	LogNormMix	**−2.975**	0.430	**0.030**	0.457	0.634	**7.960**	**26.432**	0.818
	RMTPP	−1.430	0.430	0.260	0.532	0.560	**8.054**	**26.347**	1.365
	Exponential	−1.380	0.557	0.230	0.510	1.068	**8.053**	2098.119	1.479
Wikipedia	RQS-QF	—	**2.959**	0.046	**3.455**	**4.411**	**15.305**	**46.521**	1.138
	LogNormMix	**0.239**	**2.978**	**0.037**	**3.453**	4.511	**15.405**	47.855	**1.124**
	RMTPP	1.921	3.474	0.281	4.304	4.508	17.445	50.984	1.494
	Exponential	1.897	3.399	0.271	4.070	4.830	17.197	52.587	1.523

Table 2: Various metrics computed on the test sequences for the real-world datasets.

our model directly parametrizes the quantile function which brings multiple advantages. In fact, quantile predictions are optimal under a large class of loss functions which arise in many real-world prediction problems. Furthermore, our model has closed-form expressions for quantiles, prediction intervals and the expectation. This allows us to avoid expensive sampling-based approximations. Importantly, we derive an analytical expression for the CRPS of our model which enables more efficient parameter optimization. We also demonstrate that our parametrization leads to a flexible yet tractable neural TPP model with state-of-the-art performance in standard prediction tasks on both synthetic and real-world event data. We hope our method will pave the way to new approaches for parametrizing, training, and evaluating neural TPP models. For future work, we propose to extend our model to marked TPPs as well as multivariate TPPs using multivariate scoring rules.

Acknowledgements

We would like to thank four anonymous reviewers and one meta-reviewer for their time and insightful comments. Souhaib Ben Taieb was supported by the Fund for Scientific Research (F.R.S.–FNRS) (projects CDR J.0011.20 and PDR T.0011.21).

References

Odd O Aalen, Ørnulf Borgan, and Håkon K Gjessing. *Survival and Event History Analysis: A Process Point of View*. Springer, New York, NY, 2008.

Anand Avati, Tony Duan, Sharon Zhou, Kenneth Jung, Nigam H Shah, and Andrew Y Ng. Countdown regression: Sharp and calibrated survival predictions. In Ryan P Adams and Vibhav Gogate, editors, *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pages 145–155. PMLR, 2020.

David R Cox. *Renewal Theory*. Springer Netherlands, September 1967.

D Vere-Jones D. J. Daley. *An Introduction to the Theory of Point Processes (Volume I: Elementary Theory and Methods)*. Springer-Verlag New York, 2003.

Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1555–1564. ACM, August 2016.

Tony Duan, Anand Avati, Daisy Yi Ding, Khanh K Thai, Sanjay Basu, Andrew Y Ng, and Alejandro Schuler. NGBoost: Natural gradient boosting for probabilistic prediction. In *Proceedings of the 37th International Conference on Machine Learning*, October 2019.

Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In H Wallach, H Larochelle, A Beygelzimer, F dAlché Buc, E Fox, and R Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 7511–7522. Curran Associates, Inc., 2019.

Jan Gasthaus, Konstantinos Benidis, Yuyang Wang, Syama Sundar Rangapuram, David Salinas, Valentin Flunkert, and Tim Januschowski. Probabilistic forecasting with spline quantile function RNNs. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 1901–1910. PMLR, 2019.

Manuel Gebetsberger, Jakob W Messner, Georg J Mayr, and Achim Zeileis. Estimation methods for nonhomogeneous regression models: Minimum continuous ranked probability score versus maximum likelihood. *Monthly Weather Review*, 146(12):4323–4338, December 2018.

Warren Gilchrist. *Statistical Modelling with Quantile Functions*. CRC Press, May 2000.

Tilmann Gneiting. Quantiles as optimal point forecasts. *International Journal of Forecasting*, 27(2): 197–207, April 2011.

Tilmann Gneiting and Matthias Katzfuss. Probabilistic forecasting. *Annual Review of Statistics and Its Application*, 1(1):125–151, January 2014.

Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society. Series B, Statistical methodology*, 69(2):243–268, 2007.

J A Gregory and R Delbourgo. Piecewise rational quadratic interpolation to monotonic data. *IMA Journal of Numerical Analysis*, 2(2):123–130, April 1982.

E P Grimit, T Gneiting, V J Berrocal, and N A Johnson. The continuous ranked probability score for circular variables and its application to mesoscale forecast ensemble verification. *Quarterly Journal of the Royal Meteorological Society*, 132(621C):2925–2942, October 2006.

Ruocheng Guo, Jundong Li, and Huan Liu. INITIATOR: Noise-contrastive estimation for marked temporal point process. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, California, July 2018. International Joint Conferences on Artificial Intelligence Organization.

Alan G Hawkes. Spectra of some Self-Exciting and mutually exciting point processes. *Biometrika*, 58 (1):83–90, 1971.

Alan D Hutson. Rational spline estimators of the quantile function. *Communications in Statistics - Simulation and Computation*, 30(2):377–390, April 2001.

Valerie Isham and Mark Westcott. A self-correcting point process. *Stochastic Processes and their Applications*, 8(3):335–347, May 1979.

Alexander Jordan, Fabian Krüger, and Sebastian Lerch. Evaluating probabilistic forecasts with scoringrules. *Journal of Statistical Software, Articles*, 90 (12):1–37, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR: International Conference on Learning Representations*, pages 1–15. arxiv.org, 2015.

Roger Koenker. Quantile regression: 40 years on. *Annual review of economics*, 9(1):155–176, August 2017.

Roger Koenker and Zhijie Xiao. Quantile autoregression. *Journal of the American Statistical Association*, 101(475):980–990, 2006.

Roger Koenker, Samantha Leorato, and Franco Peracchi. Distributional vs. quantile regression. Technical Report 1329, December 2013.

F Laio and S Tamea. Verification tools for probabilistic forecasts of continuous hydrological variables. *Hydrology and Earth System Sciences*, 11(4):1267–1277, 2007.

Hongyuan Mei and Jason M Eisner. The neural hawkes process: A neurally Self-Modulating multivariate point process. In I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6754–6764. Curran Associates, Inc., 2017.

Hongyuan Mei, Tom Wan, and Jason Eisner. Noise-Contrastive estimation for multivariate point processes. In H Larochelle, M Ranzato, R Hadsell, M F Balcan, and H Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 5204–5214. Curran Associates, Inc., 2020.

Allan H Murphy. THE RANKED PROBABILITY SCORE AND THE PROBABILITY SCORE: A COMPARISON. *Monthly Weather Review*, 98(12): 917–924, December 1970.

Takahiro Omi, Naonori Ueda, and Kazuyuki Aihara. Fully neural network based model for general temporal point processes. In H Wallach, H Larochelle, A Beygelzimer, F dAlché Buc, E Fox, and R Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 2120–2129. Curran Associates, Inc., 2019.

Georg Ostrovski, Will Dabney, and Remi Munos. Autoregressive quantile networks for generative modeling. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3936–3945, Stockholmsmässan, Stockholm Sweden, 2018. PMLR.

Jakob Gulddahl Rasmussen. Lecture notes: Temporal point processes and the conditional intensity function. Technical report, Aalborg University, June 2018.

Oleksandr Shchur, Marin Biloš, and Stephan Günnemann. Intensity-Free learning of temporal point processes. In *International Conference on Learning Representations (ICLR)*, 2020a.

Oleksandr Shchur, Nicholas Gao, Marin Biloš, and Stephan Günnemann. Fast and flexible temporal point processes with triangular maps. In *Advances in Neural Information Processing Systems (NeurIPS)*, June 2020b.

Oleksandr Shchur, Ali Caner Türkmen, Tim Januschowski, and Stephan Günnemann. Neural temporal point processes: A review. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4585–4593, April 2021.

James W Taylor. A quantile regression neural network approach to estimating the conditional density of multiperiod returns. *Business India*, 19(4):299–311, 2000.

Utkarsh Upadhyay, Abir De, and Manuel Gomez Rodriguez. Deep reinforcement learning of marked temporal point processes -. In S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi, and R Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 3168–3178. Curran Associates, Inc., 2018.

Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A Multi-Horizon quantile recurrent forecaster. November 2017.

Robert L Winkler. A Decision-Theoretic approach to interval estimation. *Journal of the American Statistical Association*, 67(337):187–191, 1972.

S Xiao, H Xu, J Yan, M Farajtabar, X Yang, and others. Learning conditional generative models for temporal point processes. *Thirty-Second AAAI*, 2018.

Shuai Xiao, Mehrdad Farajtabar, Xiaojing Ye, Junchi Yan, Le Song, and Hongyuan Zha. Wasserstein learning of deep generative point process models. In *Advances in Neural Information Processing Systems*, May 2017a.

Shuai Xiao, Junchi Yan, Stephen M Chu, Xiaokang Yang, and Hongyuan Zha. Modeling the intensity function of point process via recurrent neural networks. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 1597–1603, May 2017b.

Qiang Zhang, Aldo Lipani, Omer Kirnap, and Emine Yilmaz. Self-Attentive hawkes processes. In *Proceedings of Machine Learning Research*, pages 11183–11119, July 2019.

Ke Zhou, Hongyuan Zha, and Le Song. Learning triggering kernels for multi-dimensional hawkes processes. In *International Conference on Machine Learning*, pages 1301–1309, February 2013.

Simiao Zuo, Haoming Jiang, Zichong Li, Tuo Zhao, and Hongyuan Zha. Transformer hawkes process. In *International Conference on Machine Learning*, pages 11692–11702, February 2020.

A Computation of the CRPS

A.1 Integral decomposition

Consider a quantile function $q_{\boldsymbol{\theta}}$ as defined in (15), an observation $\tau \in \mathbb{R}_+$ and let $\tilde{\alpha} = q_{\bar{\boldsymbol{\theta}}}^{-1}(\tau)$. The CRPS defined in (17) can be decomposed as follows:

$$\begin{aligned}
\mathrm{CRPS}(q_{\bar{\boldsymbol{\theta}}}, \tau) &= \int_0^1 \left(\mathbb{1}\{\tau < q_{\bar{\boldsymbol{\theta}}}(\alpha)\} - \alpha\right)\left(q_{\bar{\boldsymbol{\theta}}}(\alpha) - \tau\right) \mathrm{d}\alpha && (31)\\
&= \int_0^{\tilde{\alpha}} \alpha(\tau - q_{\bar{\boldsymbol{\theta}}}(\alpha))\, \mathrm{d}\alpha + \int_{\tilde{\alpha}}^1 (1-\alpha)\left(q_{\bar{\boldsymbol{\theta}}}(\alpha) - \tau\right) \mathrm{d}\alpha && (32)\\
&= \int_0^{\tilde{\alpha}} (\alpha\tau - \alpha q_{\bar{\boldsymbol{\theta}}}(\alpha))\, \mathrm{d}\alpha + \int_{\tilde{\alpha}}^1 (q_{\bar{\theta}}(\alpha) - \tau - \alpha q_{\bar{\boldsymbol{\theta}}}(\alpha) + \alpha\tau)\, \mathrm{d}\alpha && (33)\\
&= \int_0^1 \alpha\tau\, \mathrm{d}\alpha - \int_0^1 \alpha q_{\bar{\boldsymbol{\theta}}}(\alpha)\, \mathrm{d}\alpha + \int_{\tilde{\alpha}}^1 q_{\bar{\boldsymbol{\theta}}}(\alpha)\, \mathrm{d}\alpha - \tau \int_{\tilde{\alpha}}^1 \mathrm{d}\alpha && (34)\\
&= \tau \int_0^1 \alpha\, \mathrm{d}\alpha - \tau \int_{\tilde{\alpha}}^1 \mathrm{d}\alpha - \int_0^1 \alpha q_{\bar{\boldsymbol{\theta}}}(\alpha)\, \mathrm{d}\alpha + \int_{\tilde{\alpha}}^1 q_{\bar{\boldsymbol{\theta}}}(\alpha)\, \mathrm{d}\alpha && (35)\\
&= \tau(\tilde{\alpha} - \frac{1}{2}) - \underbrace{\int_0^1 \alpha\; q_{\boldsymbol{\theta}}(\alpha)\, \mathrm{d}\alpha}_{A} + \underbrace{\int_{\tilde{\alpha}}^1 q_{\boldsymbol{\theta}}(\alpha)\, \mathrm{d}\alpha}_{B}. && (36)
\end{aligned}$$

Using (15) and applying the change of variables $\xi = (\alpha - x^{(k)})/\Delta x^{(k)}$ for $k = 1, 2, \ldots, K-1$, expression A in (36) can be written as

$$\begin{aligned}
A &= \int_0^1 \alpha\; q_{\bar{\boldsymbol{\theta}}}(\alpha)\, \mathrm{d}\alpha && (37)\\
&= \int_0^{x^{(K)}} \alpha\; q_{\boldsymbol{\theta}}(\alpha)\, \mathrm{d}\alpha + \int_{x^{(K)}}^1 \alpha\; q_{\boldsymbol{\gamma}}(\alpha)\, \mathrm{d}\alpha && (38)\\
&= \sum_{k=0}^{K-1} \int_{x^{(k)}}^{x^{(k+1)}} \alpha\; q_{\boldsymbol{\theta}}(\alpha)\, \mathrm{d}\alpha + \int_{x^{(K)}}^1 \alpha\; q_{\boldsymbol{\gamma}}(\alpha)\, \mathrm{d}\alpha && (39)\\
&= \sum_{k=0}^{K-1} \left[[\Delta x^{(k)}]^2 \int_0^1 \xi\; r_k(\xi)\, \mathrm{d}\xi + x^{(k)} \Delta x^{(k)} \int_0^1 r_k(\xi)\, \mathrm{d}\xi \right] + \int_{x^{(K)}}^1 \alpha\; q_{\boldsymbol{\gamma}}(\alpha)\, \mathrm{d}\alpha\,. && (40)
\end{aligned}$$

Recall that $\tilde{\alpha} = q_{\bar{\boldsymbol{\theta}}}^{-1}(\tau)$. Computing expression B in (36) will depend on where $\tilde{\alpha}$ lies. Let us consider the following three cases:

Case 1 ($\tilde{\alpha} < x^{(K-1)}$)**.** Let $l \in \{0, 1, \ldots, K-1\}$ denote the bin where $\tilde{\alpha}$ lies, i.e. $\tilde{\alpha} \in [x^{(l)}, x^{(l+1)}]$ and $\tau \in [y^{(l)}, y^{(l+1)}]$. We can write

$$\begin{aligned}
B &= \int_{\tilde{\alpha}}^1 q_{\bar{\boldsymbol{\theta}}}(\alpha)\, \mathrm{d}\alpha && (41)\\
&= \int_{\tilde{\alpha}}^{x^{(l+1)}} q_{\boldsymbol{\theta}}(\alpha)\, \mathrm{d}\alpha + \sum_{k=l+1}^{K-1} \int_{x^{(k)}}^{x^{(k+1)}} q_{\boldsymbol{\theta}}(\alpha)\, \mathrm{d}\alpha + \int_{x^{(K)}}^1 q_{\boldsymbol{\gamma}}(\alpha)\, \mathrm{d}\alpha && (42)\\
&= \Delta x^{(l)} \int_{\frac{\tilde{\alpha} - x^{(l)}}{\Delta x^{(l)}}}^1 r_l(\xi)\, \mathrm{d}\xi + \sum_{k=l+1}^{K-1} \Delta x^{(k)} \int_0^1 r_k(\xi)\, \mathrm{d}\xi + \int_{x^{(K)}}^1 q_{\boldsymbol{\gamma}}(\alpha)\, \mathrm{d}\alpha\,. && (43)
\end{aligned}$$

Case 2 ($x^{(K-1)} \leq \tilde{\alpha} < x^{(K)}$)**.**

$$B = \int_{\tilde{\alpha}}^1 q_{\bar{\boldsymbol{\theta}}}(\alpha)\, \mathrm{d}\alpha = \int_{\tilde{\alpha}}^{x^{(K)}} q_{\boldsymbol{\theta}}(\alpha)\, \mathrm{d}\alpha + \int_{x^{(K)}}^1 q_{\boldsymbol{\gamma}}(\alpha)\, \mathrm{d}\alpha\,. \quad (44)$$

Case 3 ($x^{(K)} \leq \tilde{\alpha} < 1$).

$$B = \int_{\tilde{\alpha}}^{1} q_{\bar{\boldsymbol{\theta}}}(\alpha)\,\mathrm{d}\alpha = \int_{\tilde{\alpha}}^{1} q_{\boldsymbol{\gamma}}(\alpha)\,\mathrm{d}\alpha\,. \tag{45}$$

A.2 Integral of linear/quadratic rational functions

As can be seen in expressions (27) and (28), computing the CRPS reduces to computing integrals of linear/quadratic rational functions. Specifically, we need to compute

$$\int_{z}^{1} \frac{A\xi + B}{a\xi^2 + b\xi + c}\,\mathrm{d}\xi\,, \tag{46}$$

where $z \in [0, 1]$, $a \neq 0$ and $b \neq 0$.

The solution of this integral will depend on the sign of the discriminant $\Delta = b^2 - 4ac$ associated to the quadratic equation $a\xi^2 + b\xi + c = 0$. Let us consider the three possible cases.

Case 1: $\Delta > 0$. Since $a\xi^2 + b\xi + c = a(\xi - \xi_1)(\xi - \xi_2)$ with $\xi_1, \xi_2 = \frac{-b \pm \sqrt{\Delta}}{2a}$, the integrand in (46) can be written as

$$\frac{A\xi + B}{a\xi^2 + b\xi + c} = \frac{1}{a}\frac{A\xi + B}{(\xi - \xi_1)(\xi - \xi_2)} = \frac{1}{a}\left[\frac{R}{(\xi - \xi_1)} + \frac{L}{(\xi - \xi_2)}\right],$$

where $R = \frac{A\xi_1 + B}{\xi_1 - \xi_2}$ and $L = \frac{A\xi_2 + B}{\xi_2 - \xi_1}$. The solution is given by

$$\int_{z}^{1} \frac{A\xi + B}{a\xi^2 + b\xi + c}\,\mathrm{d}\xi = \frac{1}{a}\left[\int_{z}^{1} \frac{R}{(\xi - \xi_1)}\,\mathrm{d}\xi + \int_{z}^{1} \frac{L}{(\xi - \xi_2)}\,\mathrm{d}\xi\right] = \frac{1}{a}\left[R\,\ln|\xi - \xi_1| + L\,\ln|\xi - \xi_2|\right]_z^1.$$

Case 2: $\Delta = 0$. Since $a\xi^2 + b\xi + c = a(\xi - \xi_0)^2$ with $\xi_0 = \frac{-b}{2a}$, we can write

$$\frac{A\xi + B}{a\xi^2 + b\xi + c} = \frac{1}{a}\frac{A\xi + B}{(\xi - \xi_0)^2} = \frac{1}{a}\left[\frac{A}{\xi - \xi_0} + \frac{B + A\xi_0}{(\xi - \xi_0)^2}\right].$$

The solution is given by

$$\int_{z}^{1} \frac{A\xi + B}{a\xi^2 + b\xi + c}\,\mathrm{d}\xi = \frac{1}{a}\left[A\,\ln|\xi - \xi_0| - \frac{B + A\xi_0}{\xi - \xi_0}\right]_z^1.$$

Case 3: $\Delta < 0$. We have an irreducible quadratic denominator. By completing the square, we obtain

$$a\xi^2 + b\xi + c = a\left[\left(\xi + \frac{b}{2a}\right)^2 + \frac{4ac - b^2}{4a^2}\right].$$

The solution is given by

$$\int_{z}^{1} \frac{A\xi + B}{a\xi^2 + b\xi + c}\,\mathrm{d}\xi = \frac{1}{a}\int_{z + \frac{b}{2a}}^{1 + \frac{b}{2a}} \frac{Au + B'}{u^2 + m^2}\,\mathrm{d}u = \frac{1}{a}\left[\frac{A}{2}\,\ln(u^2 + m^2) + \frac{B'}{m}\,\arctan\left(\frac{u}{m}\right)\right]_{z + \frac{b}{2a}}^{1 + \frac{b}{2a}},$$

where $u = \xi + \frac{b}{2a}$, $m^2 = \frac{4ac - b^2}{4a^2}$ and $B' = B - \frac{Ab}{2a}$.

All these expressions can be further simplified by replacing A, B, a, b, and c with their values given in (27) and (28).

B Computational time

Tables 3 and 4 give the computing time of our model (in seconds) averaged over ten epochs for a single forward and backward pass, respectively. The numbers in brackets give the associated standard errors. `RQS-QF-CRPS` computes the closed-form expression of the CRPS using (36) while `RQS-QF-CRPS-x` approximates the integral in (31) using the trapezoid rule with `x` evenly spaced numbers.

As expected, we can see that more bins lead to higher computing time regardless of the method used to compute the CRPS. Furthermore, we can see that `RQS-QF-CRPS-500` takes more time than `RQS-QF-CRPS` for both forward and backward passes. As can be seen in Table 4, the increase in computing time is higher for the backward pass. While it is possible to reduce computing time by using less evaluations (`RQS-QF-CRPS-200` or `RQS-QF-CRPS-100`), the approximation of the CRPS will be less accurate. Using the closed-form expression of the CRPS allows us to avoid approximation errors while having lower computing time than an approximation with a large number of evaluations.

	1	2	3	5	8	10	15
RQS-QF-CRPS	0.288 (0.059)	1.002 (0.229)	1.101 (0.154)	1.384 (0.207)	1.960 (0.329)	1.896 (0.467)	2.110 (0.447)
RQS-QF-CRPS-100	0.311 (0.092)	0.266 (0.026)	0.320 (0.107)	0.361 (0.111)	0.340 (0.058)	0.317 (0.048)	0.339 (0.059)
RQS-QF-CRPS-200	0.450 (0.039)	0.564 (0.047)	0.534 (0.049)	0.562 (0.052)	0.604 (0.063)	0.633 (0.069)	0.673 (0.083)
RQS-QF-CRPS-500	1.583 (0.392)	1.718 (0.439)	1.534 (0.407)	1.631 (0.375)	2.081 (0.710)	1.907 (0.436)	2.143 (0.695)

Table 3: Average execution time (in seconds) over 10 epochs for a forward pass with a different number of bins.

	1	2	3	5	8	10	15
RQS-QF-CRPS	0.118 (0.056)	0.700 (0.202)	0.780 (0.133)	0.986 (0.223)	1.396 (0.275)	1.399 (0.312)	1.633 (0.337)
RQS-QF-CRPS-100	0.390 (0.105)	0.363 (0.046)	0.438 (0.139)	0.485 (0.140)	0.505 (0.096)	0.487 (0.059)	0.553 (0.077)
RQS-QF-CRPS-200	0.671 (0.072)	0.859 (0.085)	0.825 (0.084)	0.884 (0.094)	0.994 (0.106)	1.083 (0.109)	1.206 (0.143)
RQS-QF-CRPS-500	3.178 (0.472)	3.339 (0.519)	3.301 (0.507)	3.477 (0.445)	4.199 (1.026)	4.139 (0.544)	4.575 (1.047)

Table 4: Average execution time (in seconds) over 10 epochs for a backward pass with a different number of bins.

Having a closed-form expression for the quantile function enables more efficient prediction intervals computation. The following Tables give the computing time (in seconds) to generate 100 equal-tailed 95% prediction intervals for both LogNormMix (with different sample sizes) and our method (with different number of knots), respectively. We can see that sampling-based approximations of prediction intervals (LogNormMix) are significantly slower to compute than prediction intervals directly extracted from the quantile function (our method). For LogNormMix, the computing time increases significantly with the sample size while it is relatively stable with respect to the number of knots for our method.

50	100	200	500	1000
2.17	4.15	9.76	26.03	55.05

1	2	3	5	8	10
0.55	0.61	0.63	0.67	0.70	0.78

Table 5: Average execution time (in seconds) to generate 100 equal-tailed 95% prediction intervals. LogNormMix, with different sample sizes (left) and our method, with different number of knots (right).

C More on our quantile model

C.1 Tail modelling

Let $(x^{(K)}, y^{(K)})$ denote the tail knot, $F_{\boldsymbol{\theta}}$, the CDF associated to the quantile function in (12) with $F_{\boldsymbol{\theta}}(y^{(K)}) = 1$, and $F_{\boldsymbol{\gamma}}$, the CDF of the tail with $F_{\boldsymbol{\gamma}}(0) = 0$. We consider the following CDF mixture for a continuous random variable $\tau \in \mathbb{R}_+$:

$$F_{\bar{\boldsymbol{\theta}}}(\tau) = x^{(K)} F_{\boldsymbol{\theta}}(\tau) + (1 - x^{(K)}) F_{\boldsymbol{\gamma}}(\tau - y^{(K)}),$$

for which the quantile function is given by

$$F_{\bar{\boldsymbol{\theta}}}^{-1}(\alpha) = \begin{cases} F_{\boldsymbol{\theta}}^{-1}(\alpha / x^{(K)}), & \text{if } 0 \leq \alpha \leq x^{(K)} \\ y^{(K)} + F_{\boldsymbol{\gamma}}^{-1}(\frac{\alpha - x^{(K)}}{1 - x^{(K)}}). & \text{if } \alpha \geq x^{(K)} \end{cases}. \tag{47}$$

Note that by construction, the quantile function in (47) satisfies the first constraint in (14), i.e. $F_{\bar{\boldsymbol{\theta}}}^{-1}(x^{(K)}) = y^{(K)}$.

For the tail distribution, we consider an exponential distribution with rate $\lambda > 0$, for which the quantile function is given by

$$F_{\boldsymbol{\gamma}}^{-1}(\alpha) = -\log(1-\alpha)/\lambda. \tag{48}$$

By plugging in (48) in (47), the quantile function for $\alpha \geq x^{(K)}$ is given by

$$q_{\boldsymbol{\gamma}}(\alpha) = y^{(K)} - \frac{1}{\lambda} \log\left(\frac{1-\alpha}{1-x^{(K)}} \right). \tag{49}$$

In order to satisfy the second constraints in (14), i.e. $\frac{\mathrm{d}q_{\boldsymbol{\gamma}}(x^{(K)})}{\mathrm{d}\alpha} = \delta^{(K)}$, we can check that we must have

$$\lambda = 1/((1 - x^{(K)})\delta^{(K)}).$$

Finally, computing expressions (40) and (43) depends on integrals which involve the tail quantile function in (49). We can show that

$$\int_{\tilde{\alpha}}^{1} q_{\boldsymbol{\gamma}}(\alpha)\,\mathrm{d}\alpha = \frac{(1-\tilde{\alpha})}{\lambda} \left(y^{(K)}\lambda + 1 - \log\left(\frac{1-\tilde{\alpha}}{1-x^{(K)}} \right) \right), \tag{50}$$

and

$$\int_{x^{(K)}}^{1} \alpha\ q_{\boldsymbol{\gamma}}(\alpha)\,\mathrm{d}\alpha = -\frac{\left(x^{(\mathrm{K})} - 1\right)\left(\left(2x^{(\mathrm{K})} + 2\right) y^{(K)}\lambda + x^{(\mathrm{K})} + 3\right)}{4\lambda}.$$

C.2 Closed-form expression for the expectation

Let $q_{\bar{\boldsymbol{\theta}}}$ be the quantile function of the continuous random variable $\tau \in \mathbb{R}_+$, as defined in (15). The expectation is given by

$$\mathbb{E}[\tau] = \int_0^1 q_{\bar{\boldsymbol{\theta}}}(\alpha)\,\mathrm{d}\alpha\,, \tag{51}$$

$$= \int_0^{x^{(K)}} q_{\boldsymbol{\theta}}(\alpha)\,\mathrm{d}\alpha + \int_{x^{(K)}}^1 q_{\boldsymbol{\gamma}}(\alpha)\,\mathrm{d}\alpha\,, \tag{52}$$

which can be computed using expressions (41) and (50) with $\tilde{\alpha} = 0$ and $l = 0$.

C.3 Transformations

A useful property of quantile functions for continuous random variables is that they are equivariant under monotonic transformations. Consider the random variable $Y \in \mathbb{R}_+$, the strictly monotonic transformation g, and define $Z = g(Y)$. Let $q_{\boldsymbol{\theta};Y}$ and $q_{\tilde{\boldsymbol{\theta}};Z}$ be the model for the quantile function of Y and Z, respectively. We can write

$$q_{\boldsymbol{\theta};Y}(\alpha) = g^{-1}(q_{\tilde{\boldsymbol{\theta}};Z}(\alpha))$$

and

$$q'_{\boldsymbol{\theta};Y}(\alpha) = (g^{-1})'(q_{\tilde{\boldsymbol{\theta}};Z}(\alpha))q'_{\tilde{\boldsymbol{\theta}};Z}(\alpha).$$

For the CDFs and their derivaties, we can write

$$F_{\boldsymbol{\theta};Y}(y) = F_{\tilde{\boldsymbol{\theta}};Z}(g(y))$$

and

$$F'_{\boldsymbol{\theta};Y}(y) = F'_{\tilde{\boldsymbol{\theta}};Z}(g(y))\; g'(y).$$

In our experiments, to stabilize the variance of the data, we used the transformation $g(Y) = \log(1+Y)/\hat{\sigma}$ where $\hat{\sigma}$ is the empirical standard deviation of the inter-arrival times computed using the training sequences.

D Datasets

We use five simulated datasets from different processes, as described in Omi et al. (2019): *Hawkes 1 and 2* (Hawkes, 1971), *Self-correcting* (Isham and Westcott, 1979), *Renewal* (Cox, 1967) and *Poisson*. For each synthetic dataset, there are 64 sequences with 1024 events. See Appendix E.1 in Shchur et al. (2020a) for more details. We also consider twelve real-world datasets: *Taxi* (customer pickupts), *Wikipedia* (article edits), *Reddit*, *Reddit-C*, *Reddit-S* (comments), *Yelp*, *Yelp-A*, *Yelp-M* (check-ins to restaurants), *PUBG* (online gaminig), *LastFM* (music playback), *Twitter* (tweets) and *MOOC* (online courses). Each dataset consists of multiple sequences of event times, and has been used for (neural) TPP modelling (Shchur et al., 2021, 2020a). Appendix E.2 in Shchur et al. (2020a) gives more details for *Wikipedia*, *Reddit*, *Yelp*, *LastFM*, *Twitter* and *MOOC*, while Appendix D in Shchur et al. (2021) has more information for the other datasets. For each dataset, we remove sequences with less than five events. For *Reddit* and *MOOC*, we reduce the number of sequences to save computational time. Specifically, for *Reddit*, we use a subset of 2290 sequences with 101529 events (instead of 10000 sequences with 672350 events). For *MOOC*, we use 2178 sequences with 81955 events (instead of 7047 sequences with 396633 events).

E Experiments

E.1 Additional setup details

The training sequences were split into sequences of length at most 128 and we perform training using mini-batches composed of 64 sequences. We define an epoch as the processing of all the training sequences. We use the validation set for hyperparameters tuning including early stopping. We pick the hyperparameter configuration that achieves the smallest validation error, averaged over three random initializations. With the NLL, we use a patience of $p = 100$ epochs with a maximum of 2000 epochs. With the CRPS, we use a patience of $p = 50$ epochs with a maximum of 700 epochs. In other words, if there is no improvement after p epochs, we stop training the model and return the model parameters with the lowest validation error. For *LastFM* and *Reddit*, we used $K = 8$ components for LogNormMix since we found that sampling from the model was more stable with less components. Note that Shchur et al. (2020a) found that LogNormMix was robut to the choice of K. We used a machine composed of an Intel Xeon Platinum 8275CL @ 3.00GHz CPU, 192GB RAM. Our implementation uses PyTorch[4] and builds on the implementations of Durkan et al. (2019)[5] and Shchur et al. (2020a)[6].

E.2 Computation of the CRPS

Since not all compared models have a closed-form expression for the CRPS, we use a sampling-based approach to compute the CRPS. Let $F_{\boldsymbol{\theta}}$ be a predictive CDF and assume it has finite first moment. The CRPS can be written as

$$\text{CRPS}(F_{\boldsymbol{\theta}}, \tau) = \mathbb{E}_{F_{\boldsymbol{\theta}}}|X_1 - \tau| - \frac{1}{2}\mathbb{E}_{F_{\boldsymbol{\theta}},F_{\boldsymbol{\theta}}}[X_1 - X_2], \tag{53}$$

where X_1 and X_2 are two independent random variables with distribution $F_{\boldsymbol{\theta}}$ (Gneiting et al., 2007).

If we draw n samples, $\tau_1, \tau_2, \ldots, \tau_n$, from $F_{\boldsymbol{\theta}}$, a natural approximation of $F_{\boldsymbol{\theta}}$ is given by the empirical CDF

$$F_{\boldsymbol{\theta}}^{(n)}(\tau) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\{\tau_i \leq \tau\}.$$

[4] `https://pytorch.org/`.

[5] `https://github.com/bayesiains/nsf`

[6] `https://github.com/shchur/ifl-tpp/tree/original-code`

Then, using (53), we can compute (Grimit et al., 2006)

$$\mathrm{CRPS}(F_{\boldsymbol{\theta}}^{(n)}, \tau) = \frac{1}{n}\sum_{i=1}^{n} |\tau_i - \tau| - \frac{1}{2n^2}\sum_{i=1}^{n}\sum_{j=1}^{n} |\tau_i - \tau_j|. \tag{54}$$

We used a computationally more efficient and algebraically equivalent representation of the CRPS, given by (Murphy, 1970; Laio and Tamea, 2007)

$$\mathrm{CRPS}(F_{\boldsymbol{\theta}}^{(n)}, \tau) = \frac{2}{n^2}\sum_{i=1}^{n} |\tau_{(i)} - \tau| \left(n \, \mathbb{1}\{\tau < \tau_{(i)}\} - i + \frac{1}{2} \right), \tag{55}$$

where $\tau_{(1)}, \tau_{(2)}, \dots, \tau_{(n)}$ are the sorted samples. We used $n = 500$ in all our experiments. We refer the reader to Jordan et al. (2019) for more details.

E.3 Metrics

All the metrics are computed by summing over all the S test sequences, i.e. $s = 1, 2, \dots, S$, and over all the observations, i.e. $i = 1, 2, \dots, N_s$ where N_s is the number of events for the s-th sequence.

We compute the mean quantile score (QSm) by averaged the quantile score (QS) as defined in (18) for all probability levels ranging from 1% to 99%. In other words, we compute

$$\mathrm{QSm} = \frac{1}{99}\sum_{k=1}^{99} \mathrm{QS}^{(k)},$$

where

$$\mathrm{QS}^{(k)} = \frac{1}{S}\frac{1}{N_s}\sum_{s=1}^{S}\sum_{i=1}^{N_s} \mathrm{QS}_{k/100}\left(q_{\boldsymbol{\theta}_{s,i}}(k/100), \tau_{s,i} \right), \tag{56}$$

and $\boldsymbol{\theta}_{s,i}$ are the parameters used to predict the i-th observation in the s-th sequence. In the main text, we used QS50 and QS90 as a shorthand for $\mathrm{QS}^{(50)}$ and $\mathrm{QS}^{(90)}$, respectively.

Similarly, we define the mean absolute calibration error (MACE) as

$$\mathrm{MACE} = \frac{1}{99}\sum_{k=1}^{99} \mathrm{ACE}^{(k)}, \tag{57}$$

where

$$ACE^{(k)} = \left| \frac{k}{100} - \frac{1}{S}\frac{1}{N_s}\sum_{s=1}^{S}\sum_{i=1}^{N_s} \cdot I\left(\tau_{s,i} \leq q_{\boldsymbol{\theta}_{s,i}}(k/100) \right) \right|, \tag{58}$$

and I is an indicator function.

For a continuous distribution, the $(1-\alpha) \times 100\%$ equal-tailed prediction interval centered at the median, with $\alpha \in (0, 1)$, should have $\alpha/2$ probability mass in both tails. We evaluate the accuracy of equal-tailed prediction intervals for target coverage probabilities ranging from 2% to 98% using the interval score (IS).

Let $\tau_{s,i}$ be a realization, and $[l_{s,i}^{(k)}, u_{s,i}^{(k)}]$, the $(1-\rho) \times 100\%$ prediction intervals with $l_{s,i}^{(k)} = q_{\boldsymbol{\theta}_{s,i}}(0.5 - k/100)$, $u_{s,i}^{(k)} = q_{\boldsymbol{\theta}_{s,i}}(0.5 + k/100)$, $\rho = \frac{2k}{100}$, and $k = 1, 2, \dots, 48$. The interval score is given by

$$\mathrm{IS}^{(k)} = \frac{1}{S}\frac{1}{N_s}\sum_{s=1}^{S}\sum_{i=1}^{N_s} (u_{s,i}^{(k)} - l_{s,i}^{(k)}) + \frac{2}{\rho}(l_{s,i}^{(k)} - \tau_{s,i}) I(\tau_{s,i} < l_{s,i}^{(k)}) + \frac{2}{\rho}(\tau_{s,i} - u_{s,i}^{(k)}) I(\tau_{s,i} > u_{s,i}^{(k)}). \tag{59}$$

Note that the interval score essentially combines the interval width with the quantile scores for both the $1-\rho/2$ and the $\rho/2$ quantiles. In the main text, we used IS50 and IS90 as a shorthand for $\mathrm{IS}^{(25)}$ and $\mathrm{IS}^{(45)}$, respectively.

Finally, using the median as point forecast, we compute the Symmetric Mean Absolute Percentage Error (SMAPE) as follows:

$$\text{SMAPE} = \frac{1}{S}\frac{1}{N_s}\sum_{s=1}^{S}\sum_{i=1}^{N_s} 2 \times \frac{|\tau_{s,i} - \hat{\tau}_{s,i}|}{|\tau_{s,i}| + |\hat{\tau}_{s,i}|},$$

where $\hat{\tau}_{s,i} = q_{\boldsymbol{\theta}_{s,i}}(0.5)$.

F Additional results

Table 6 gives the average quantile score on the test set for our model with different number of bins. Tables 7 and 8 give the same information as Tables 1 and 2 ncluding standard errors. Detailed results are also given in Figure 3 for synthetic datasets, and Figures 4, 5 and 6 for real-world datasets. For a better visualization, we only plot the results of our model and the best baseline (LogNormMix).

	1	2	3	8	10	15
Yelp A	0.437	0.438	0.436	0.438	0.435	0.435
Taxi	0.128	0.128	0.128	0.128	0.128	0.129
Yelp M	0.264	0.264	0.264	0.262	0.260	0.261
Twitter	0.519	0.516	0.516	0.516	0.516	0.516
Wikipedia	2.969	2.953	2.954	2.952	2.953	2.952
PUBG	0.214	0.214	0.214	0.214	0.214	0.214
Yelp	1.894	1.892	1.891	1.889	1.889	1.889
Reddit-C	0.044	0.044	0.044	0.044	0.044	0.044
Reddit-S	0.011	0.011	0.011	0.011	0.011	0.011
LastFM	0.374	0.372	0.372	0.373	0.372	0.372
MOOC	6.695	6.570	6.565	6.563	6.562	6.560
Reddit	9.481	9.423	9.425	9.404	9.410	9.395

Table 6: Average quantile score (QSm) for RQS-QF with different number of bins.

		NLL	QSm	MACE	QS50	QS90	IS50	IS90
Poisson	RQS-QF	—	**0.496** (0.003)	**0.011** (0.000)	**0.686** (0.005)	**0.463** (0.001)	**2.232** (0.012)	**3.966** (0.009)
	LogNormMix	**0.989** (0.005)	**0.496** (0.002)	0.011 (0.000)	**0.686** (0.005)	**0.462** (0.001)	**2.232** (0.012)	**3.968** (0.008)
	RMTPP	**0.989** (0.005)	**0.496** (0.003)	**0.011** (0.001)	**0.686** (0.005)	**0.462** (0.002)	**2.231** (0.012)	**3.971** (0.010)
	Exponential	**0.989** (0.005)	**0.496** (0.002)	**0.011** (0.001)	**0.686** (0.005)	**0.463** (0.001)	**2.232** (0.012)	**3.969** (0.006)
Renewal	RQS-QF	—	**0.802** (0.009)	0.011 (0.001)	**0.921** (0.012)	**1.229** (0.019)	**3.461** (0.040)	**11.640** (0.146)
	LogNormMix	**0.260** (0.006)	**0.802** (0.009)	**0.009** (0.000)	**0.921** (0.012)	**1.230** (0.020)	**3.461** (0.040)	**11.657** (0.157)
	RMTPP	0.993 (0.009)	0.911 (0.010)	0.219 (0.002)	1.112 (0.013)	**1.237** (0.019)	3.922 (0.041)	11.809 (0.166)
	Exponential	0.981 (0.010)	0.905 (0.012)	0.217 (0.002)	1.104 (0.018)	**1.234** (0.021)	3.897 (0.053)	11.802 (0.166)
Hawkes1	RQS-QF	—	**0.715** (0.024)	**0.011** (0.000)	**0.887** (0.037)	**0.940** (0.039)	**3.176** (0.108)	**8.176** (0.250)
	LogNormMix	**0.513** (0.037)	**0.723** (0.025)	0.014 (0.003)	**0.891** (0.038)	**0.968** (0.044)	**3.203** (0.114)	8.482 (0.277)
	RMTPP	0.735 (0.045)	0.759 (0.023)	0.106 (0.003)	0.964 (0.032)	**0.949** (0.040)	3.335 (0.096)	8.525 (0.327)
	Exponential	0.726 (0.043)	0.804 (0.025)	0.095 (0.003)	0.964 (0.036)	1.140 (0.043)	3.531 (0.109)	10.220 (0.318)
Self-correcting	RQS-QF	—	0.344 (0.001)	**0.010** (0.001)	0.496 (0.001)	0.232 (0.001)	1.544 (0.004)	2.436 (0.000)
	LogNormMix	0.784 (0.001)	0.351 (0.002)	0.013 (0.002)	0.509 (0.004)	0.230 (0.002)	1.580 (0.008)	2.293 (0.011)
	RMTPP	**0.775** (0.002)	**0.340** (0.001)	**0.010** (0.000)	**0.494** (0.002)	**0.225** (0.002)	**1.533** (0.005)	**2.241** (0.011)
	Exponential	0.935 (0.001)	0.415 (0.002)	0.073 (0.000)	0.600 (0.004)	0.316 (0.001)	1.834 (0.009)	3.202 (0.017)
Hawkes2	RQS-QF	—	**0.783** (0.019)	**0.010** (0.000)	**0.967** (0.030)	**1.058** (0.026)	**3.482** (0.085)	**9.007** (0.160)
	LogNormMix	**0.016** (0.029)	**0.793** (0.021)	0.011 (0.001)	**0.969** (0.031)	1.091 (0.037)	**3.528** (0.096)	9.292 (0.225)
	RMTPP	0.688 (0.034)	0.870 (0.019)	0.192 (0.002)	1.110 (0.030)	**1.068** (0.030)	3.820 (0.083)	9.713 (0.246)
	Exponential	0.684 (0.032)	0.901 (0.020)	0.185 (0.002)	1.092 (0.030)	1.261 (0.036)	3.964 (0.088)	11.164 (0.273)

Table 7: Various metrics computed on the test sequences (synthetic datasets). Standard errors are given in brackets.

		NLL	QSm	MACE	QS50	QS90	IS50	IS90
Yelp A	RQS-QF	—	**0.436** (0.004)	**0.059** (0.001)	0.592 (0.006)	**0.443** (0.004)	**2.074** (0.033)	**4.054** (0.030)
	LogNormMix	**0.660** (0.007)	0.446 (0.001)	0.061 (0.002)	**0.589** (0.002)	0.502 (0.003)	2.109 (0.012)	4.349 (0.037)
	RMTPP	0.693 (0.007)	0.447 (0.002)	0.067 (0.001)	0.602 (0.002)	0.466 (0.006)	2.111 (0.012)	4.234 (0.071)
	Exponential	0.683 (0.007)	0.453 (0.002)	0.066 (0.001)	0.592 (0.004)	0.515 (0.003)	2.118 (0.021)	4.851 (0.024)
Yelp M	RQS-QF	—	**0.261** (0.003)	**0.048** (0.002)	**0.351** (0.002)	**0.279** (0.008)	**1.252** (0.005)	**2.682** (0.064)
	LogNormMix	**−0.181** (0.012)	0.277 (0.005)	0.050 (0.001)	0.358 (0.005)	0.334 (0.018)	1.336 (0.014)	2.984 (0.171)
	RMTPP	−0.111 (0.012)	0.270 (0.005)	0.066 (0.001)	0.363 (0.006)	**0.283** (0.010)	1.293 (0.018)	**2.619** (0.065)
	Exponential	−0.120 (0.013)	0.300 (0.003)	0.061 (0.002)	0.369 (0.005)	0.401 (0.005)	1.400 (0.009)	3.999 (0.014)
PUBG	RQS-QF	—	**0.214** (0.001)	**0.035** (0.000)	**0.290** (0.001)	**0.220** (0.001)	**0.982** (0.002)	1.950 (0.002)
	LogNormMix	**−1.095** (0.009)	0.234 (0.001)	0.038 (0.001)	0.308 (0.001)	0.260 (0.001)	1.070 (0.002)	2.219 (0.007)
	RMTPP	−0.096 (0.002)	0.215 (0.001)	0.042 (0.000)	0.292 (0.001)	**0.219** (0.001)	**0.984** (0.002)	**1.926** (0.005)
	Exponential	−0.096 (0.002)	0.235 (0.001)	0.041 (0.001)	0.310 (0.001)	0.263 (0.001)	1.074 (0.002)	2.260 (0.006)
Wikipedia	RQS-QF	—	**2.959** (0.038)	0.046 (0.003)	**3.455** (0.059)	**4.411** (0.057)	**15.305** (0.171)	**46.521** (0.472)
	LogNormMix	**0.239** (0.068)	**2.978** (0.036)	**0.037** (0.000)	**3.453** (0.060)	4.511 (0.051)	**15.405** (0.176)	47.855 (0.425)
	RMTPP	1.921 (0.050)	3.474 (0.040)	0.281 (0.001)	4.304 (0.066)	4.508 (0.054)	17.445 (0.180)	50.984 (0.552)
	Exponential	1.897 (0.043)	3.399 (0.052)	0.271 (0.002)	4.070 (0.093)	4.830 (0.050)	17.197 (0.166)	52.587 (0.484)
LastFM	RQS-QF	—	**0.373** (0.005)	0.039 (0.002)	**0.441** (0.005)	**0.544** (0.013)	**7.646** (0.651)	**24.723** (3.941)
	LogNormMix	**−2.975** (0.044)	0.430 (0.007)	**0.030** (0.001)	0.457 (0.004)	0.634 (0.010)	**7.960** (0.634)	**26.432** (4.095)
	RMTPP	−1.430 (0.057)	0.430 (0.007)	0.260 (0.003)	0.532 (0.010)	0.560 (0.014)	**8.054** (0.718)	**26.347** (3.760)
	Exponential	−1.380 (0.048)	0.557 (0.080)	0.230 (0.002)	0.510 (0.004)	1.068 (0.325)	**8.053** (0.701)	2098.119 (1694.523)
Taxi	RQS-QF	—	**0.128** (0.001)	0.040 (0.001)	**0.172** (0.002)	0.131 (0.002)	**0.623** (0.021)	1.307 (0.082)
	LogNormMix	**−0.568** (0.013)	0.129 (0.002)	0.043 (0.001)	**0.174** (0.003)	0.133 (0.002)	**0.635** (0.022)	**1.267** (0.063)
	RMTPP	**−0.563** (0.012)	**0.128** (0.001)	0.040 (0.001)	**0.173** (0.002)	**0.128** (0.002)	**0.624** (0.022)	**1.227** (0.060)
	Exponential	**−0.563** (0.013)	0.137 (0.002)	**0.038** (0.000)	0.178 (0.003)	0.156 (0.002)	0.656 (0.019)	1.554 (0.056)
Twitter	RQS-QF	—	**0.517** (0.020)	0.095 (0.003)	**0.637** (0.031)	**0.699** (0.027)	**2.939** (0.108)	**6.847** (0.188)
	LogNormMix	**−0.054** (0.070)	**0.517** (0.021)	**0.092** (0.002)	**0.634** (0.032)	**0.705** (0.038)	**2.924** (0.112)	**6.952** (0.322)
	RMTPP	0.528 (0.054)	0.572 (0.019)	0.181 (0.004)	0.730 (0.028)	**0.710** (0.032)	3.080 (0.095)	7.862 (0.300)
	Exponential	0.527 (0.055)	0.564 (0.021)	0.179 (0.004)	0.716 (0.032)	**0.714** (0.031)	3.054 (0.101)	7.684 (0.281)
Yelp	RQS-QF	—	**1.894** (0.022)	**0.021** (0.000)	**2.586** (0.038)	**1.886** (0.022)	**9.644** (0.153)	**18.463** (0.237)
	LogNormMix	**1.620** (0.039)	4.095 (1.708)	0.026 (0.001)	2.659 (0.038)	2.047 (0.030)	10.021 (0.137)	19.664 (0.173)
	RMTPP	1.960 (0.023)	**1.908** (0.022)	0.047 (0.001)	**2.595** (0.038)	**1.897** (0.023)	**9.704** (0.151)	18.922 (0.232)
	Exponential	1.961 (0.024)	2.007 (0.025)	0.045 (0.002)	2.675 (0.040)	2.151 (0.037)	10.122 (0.147)	21.360 (0.293)
Reddit-C	RQS-QF	—	**0.044** (0.001)	0.046 (0.003)	**0.057** (0.001)	**0.050** (0.001)	**0.864** (0.049)	**2.129** (0.132)
	LogNormMix	**−2.451** (0.024)	0.045 (0.001)	0.052 (0.000)	**0.058** (0.001)	0.053 (0.001)	0.897 (0.049)	2.272 (0.130)
	RMTPP	−2.321 (0.018)	**0.044** (0.001)	0.047 (0.002)	**0.058** (0.001)	**0.049** (0.001)	**0.849** (0.047)	**2.201** (0.143)
	Exponential	−2.318 (0.018)	0.048 (0.001)	**0.038** (0.001)	0.061 (0.001)	0.061 (0.001)	**0.887** (0.049)	**2.243** (0.141)
Reddit-S	RQS-QF	—	**0.011** (0.000)	0.011 (0.000)	**0.015** (0.000)	**0.011** (0.000)	**0.056** (0.000)	**0.103** (0.001)
	LogNormMix	**−3.207** (0.023)	0.011 (0.000)	0.031 (0.009)	0.016 (0.000)	0.011 (0.000)	0.057 (0.001)	0.110 (0.003)
	RMTPP	−3.005 (0.008)	**0.011** (0.000)	**0.010** (0.000)	**0.015** (0.000)	**0.011** (0.000)	**0.056** (0.000)	**0.103** (0.001)
	Exponential	−3.005 (0.008)	0.012 (0.000)	0.012 (0.000)	0.016 (0.000)	0.014 (0.000)	0.061 (0.000)	0.136 (0.001)
MOOC	RQS-QF	—	**6.604** (0.112)	**0.072** (0.001)	**6.873** (0.152)	**12.050** (0.280)	**27.869** (0.653)	**124.777** (2.003)
	LogNormMix	**−1.764** (0.059)	**6.604** (0.111)	0.083 (0.001)	**6.875** (0.152)	**12.217** (0.264)	**27.892** (0.644)	**126.385** (2.092)
	RMTPP	2.912 (0.022)	9.140 (0.118)	0.414 (0.001)	10.812 (0.165)	12.848 (0.223)	39.205 (0.646)	131.462 (2.621)
	Exponential	2.891 (0.019)	9.061 (0.099)	0.415 (0.002)	10.627 (0.128)	13.002 (0.215)	38.843 (0.524)	132.194 (2.493)
Reddit	RQS-QF	—	9.417 (0.046)	0.058 (0.001)	12.467 (0.058)	**10.890** (0.124)	41.946 (0.232)	**93.385** (1.123)
	LogNormMix	**2.940** (0.024)	**9.369** (0.047)	**0.053** (0.001)	**12.368** (0.066)	**10.971** (0.119)	**41.647** (0.211)	**94.204** (0.930)
	RMTPP	3.612 (0.007)	10.060 (0.055)	0.149 (0.002)	13.428 (0.072)	11.030 (0.103)	44.567 (0.246)	99.713 (1.018)
	Exponential	3.607 (0.007)	10.032 (0.049)	0.149 (0.001)	13.376 (0.063)	11.129 (0.110)	44.383 (0.216)	99.757 (1.160)

Table 8: Various metrics computed on the test sequences (real-world datasets). Standard errors are given in brackets.

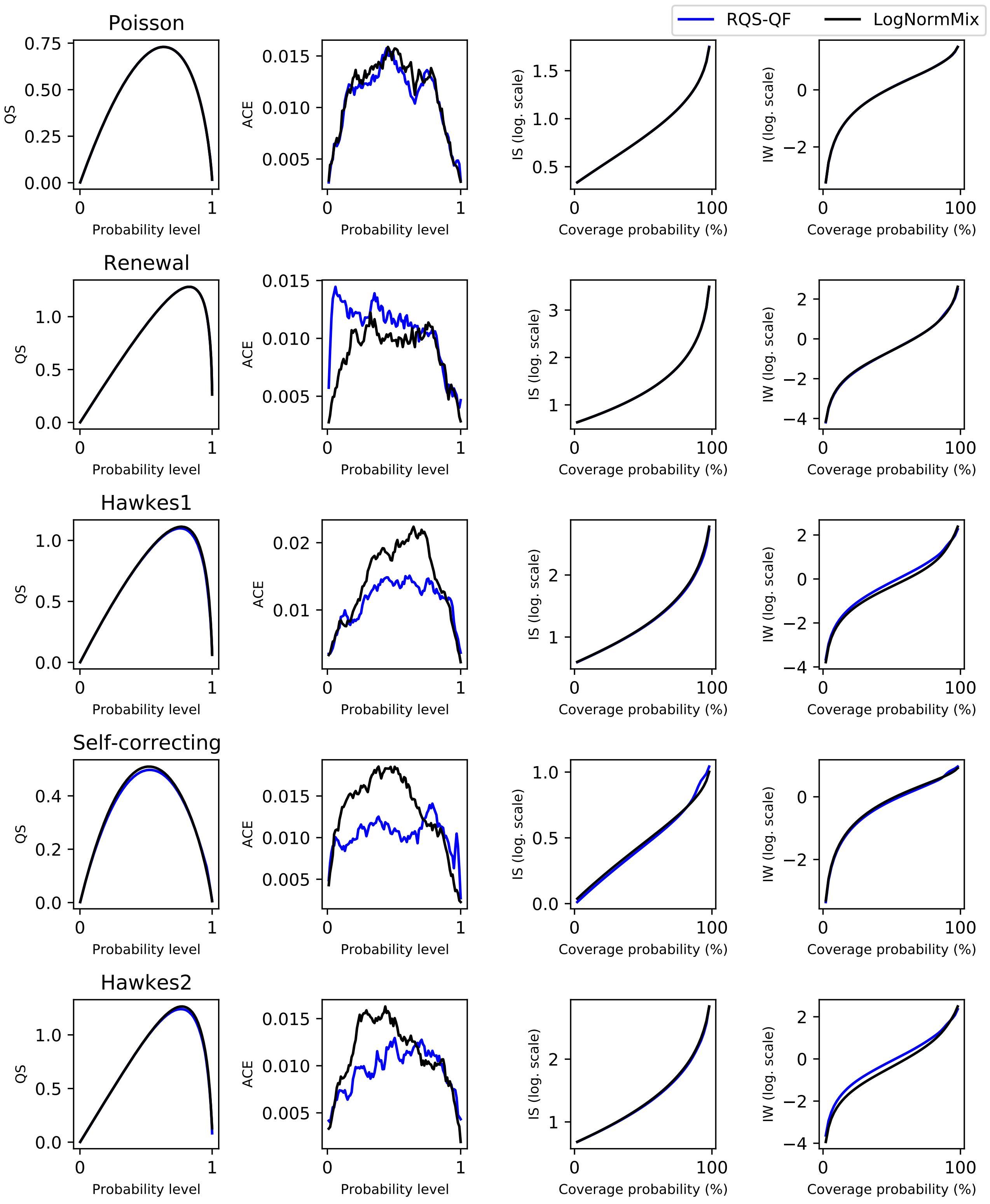

Figure 3: Various metrics for all probability levels and coverage probabilities computed on the test sequences (synthetic datasets).

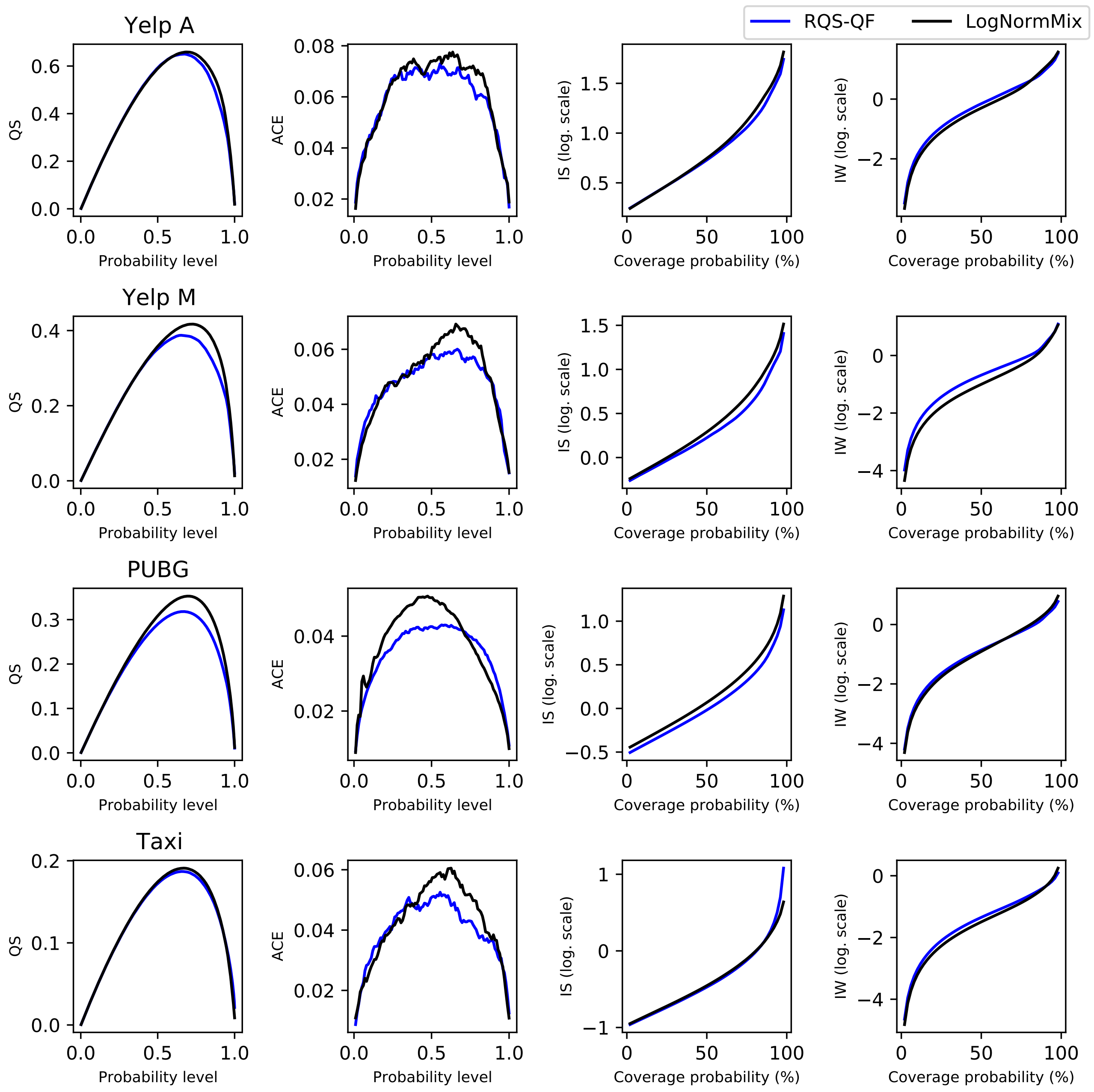

Figure 4: Various metrics for all probability levels and coverage probabilities computed on the test sequences (real-world datasets).

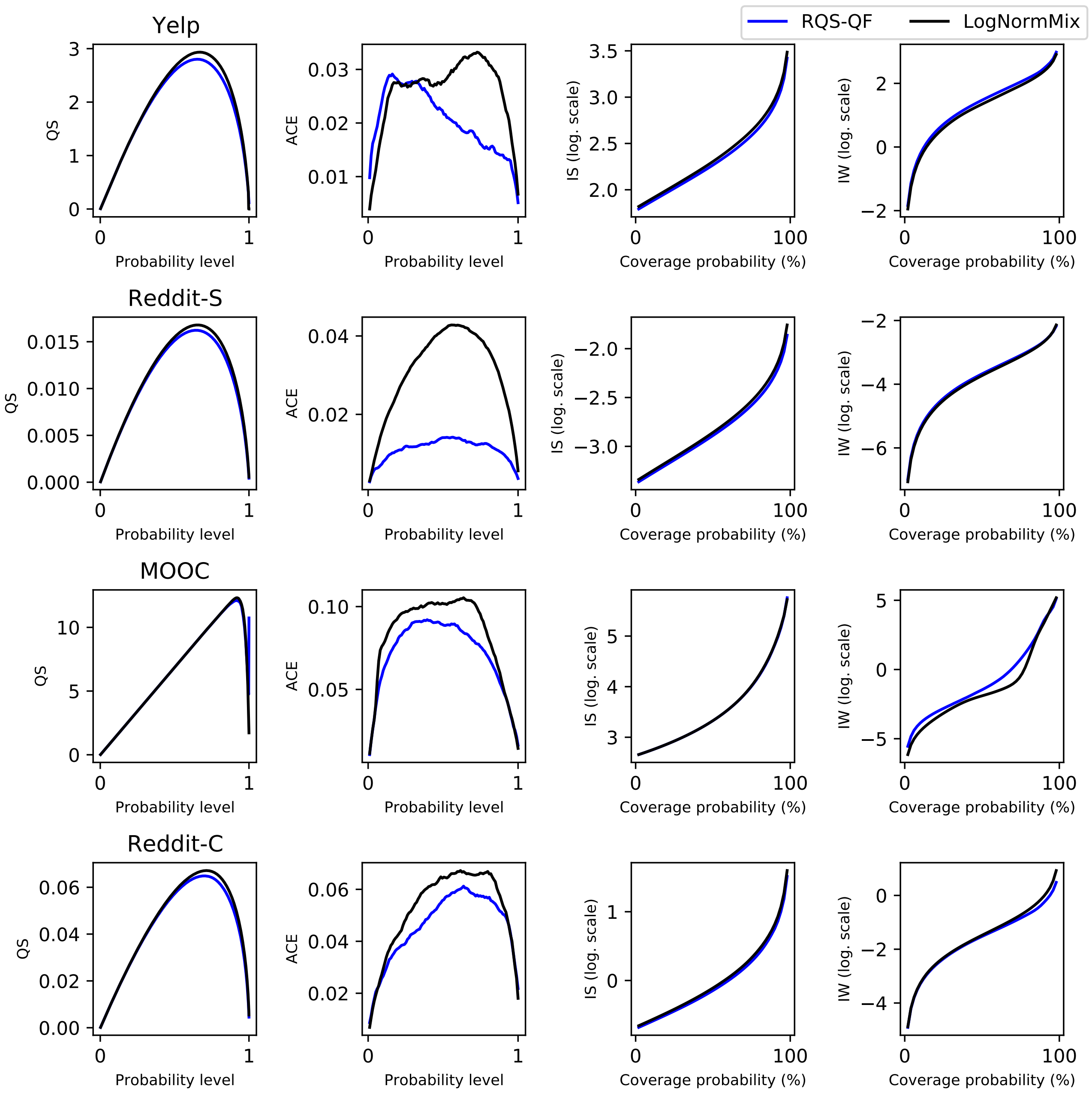

Figure 5: Various metrics for all probability levels and coverage probabilities computed on the test sequences (real-world datasets).

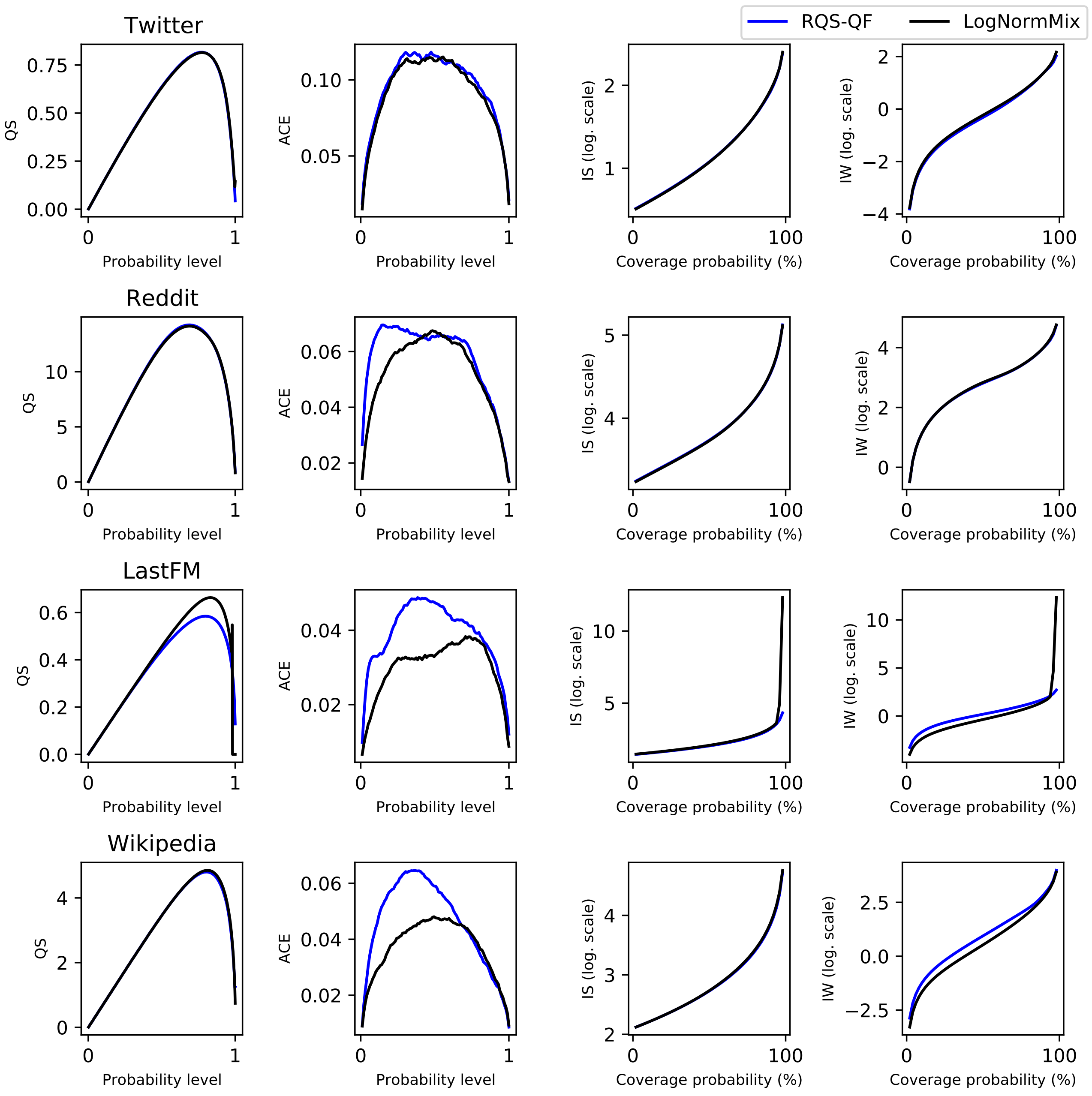

Figure 6: Various metrics for all probability levels and coverage probabilities computed on the test sequences (real-world datasets).