DSC 180B Final Report

Judy Jin z3jin@ucsd.edu

Kailing Ding k5ding@ucsd.edu

Miles Labrador mlabrado@ucsd.edu

Derek Leung djleung@ucsd.edu

1 Introduction

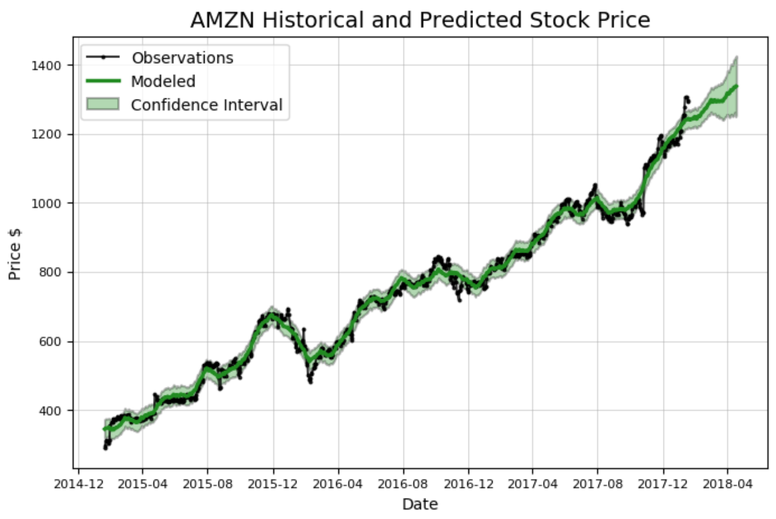
In the first quarter of our senior project, we mainly focused on uncertainty quantification with deep learning models using 3 different methods: quantile regression [2], MIS(Mean Interval Score) regression [8], and SG-MCMC [1]. And we implemented the first two methods in torchTS. In the second quarter, our research work focused on the task of advancing quantile regression using the conformal prediction technique, which provides guaranteed coverage during inference. Additionally, we designed a TimeSeriesDataset class to help prepare time-series data for our predictive model. Lastly, we helped improve the user experience of torchTS through dedicated library API documentation.

2 Background Info & Motivation

2.1 Task Description

Data that has space and time variables embedded within its structure, or spatiotemporal data, provides feature-rich information that can be utilized for insights in special tasks. By utilizing deep neural networks, spatiotemporal analysis can be used to make predictions that forecast trends given previously recorded data. However, many deep learning applications to spatiotemporal problems currently fall short with single point-estimates providing no information on the uncertainty that lies within predictions.

For example, consider an automatic stock trading system where a machine learning model predicts the stock price. A point prediction from the model might be dramatically different from the real value because of the high stochasticity of the stock market. But, on the other hand, if the model could estimate the interval which guarantees coverage of the true value with high probability, the trading system could compute the best and worst rewards and make more sensible decisions.



Preprint. Under review.

Example of stock market prediction with uncertainty quantification. Source: Will Koehrsen, Towards Data Science

Over the course of our research, we investigate different implementations of uncertainty quantification that perform differently based on their implementations and assumptions. By understanding and characterizing these differences, we hope to implement uncertainty quantification into torchTS in a manner that aligns with torchTS' friendly design philosophy.

Beyond this, we also seek to contribute to the torchTS library by implementing a data loader class. This class was to be designed to preprocess and split up data into training, calibration, and test sets in a more consistent format for our models to be more easily applied. Lastly, we aim to improve the torchTS library API documentation to present the library's functionality in an easily understood way as well as present users with examples of torchTS' spatiotemporal analysis methods being used.

2.2 Related Work

Spatiotemporal data involves collecting information with the added dimensions of space and time. Spatiotemporal data analysis builds upon existing data analysis methodology and can be used to extract insights with predictions based on the features within a dataset. Spatiotemporal forecasting implementations vary depending on the specific task. For example, spatial traffic data may have two data records that close in space and time, yet are mostly unrelated with opposite flow directions and a barrier.[5]. These quirks behind spatiotemporal phenomena mean that novel approaches were considered when developing analysis techniques suitable for spatiotemporal phenomena.

The further development of Deep Neural Networks has led to their greater adoption within many domains for problem solving, including the domain of spatiotemporal analysis. In the past, neural networks could provide accurate predictions, yet were unable to provide a sense of uncertainty attached to those predictions.[3]. Through forms of uncertainty quantification, such as quantile regression, we hope to enrich our deep learning applications to spatiotemporal data with the missing uncertainty values.

[6] proposed a conformal quantile regression model which aimed to combine the strengths of conformal prediction, which produces a confidence interval using finite samples, without making distributional assumptions, and quantile regression, which is adaptive to heteroscedasticity. They also proved the coverage guarantee with predictions using this strategy. In our work, we primarily adopt the model methodology from the above paper, and implement our code largely dependent on the conformal example Romano et al. provided.

We also refer to [7], in which they applied the conformal prediction on dynamic time-series. In their work, [7] proposed a conformal prediction method that could aggregate bootstraps from the trained estimators to avoid outfitting. Specifically, their method did not require data-splitting nor training multiple estimators, which made the model easier to implement. We will target this method in the future.

2.3 Motivation

The motivation of adding uncertainty quantification prediction to our model in *torchTS* is to provide a guarantee of valid coverage with high probability that prediction intervals hold the true value. In other words, if we use conformal prediction method in our model, when we make prediction intervals, the intervals will cover most of possible outcomes. This is crucial in many real-world applications such as stock price prediction, insurance calculation, and risk assessment, etc. because it helps minimize the risks.

3 Uncertainty Quantification

3.1 Quantile Regression

The first model we explore is quantile regression where we generate one quantile prediction each time the model is trained. Training this model multiple times with different quantile levels provides us with a confidence interval which demonstrates uncertainty quantification at our desired levels of confidence. For this implementation, the loss function calculates a quantile loss called the pinball loss function, defined as:

$$L_{Quantile}(y, f(x), \theta, p))$$
$$= min_{\theta}\{\mathbb{E}_{(x,y)\sim D}[(y - f(x))(p - \nVdash\{y < f(x)\})]\}$$

Where our loss is computed as a function of the actual value y, the input x, output f(x) of a neural network, and our fixed confidence level p, parameterized by θ. One potential issue using this model is that the prediction for different quantiles may cross each other if the amount of data the model is trained on is not large enough. In order to artificially increase the data we train this model on, we can sample multiple different subsets of data from our training dataset.

3.2 MIS Regression

The second model we investigate is Mean Interval Score (MIS) Regression. We run the model multiple times, each time updating our parameters according to a loss function that is mathematically the same as MIS, also known as Winkler loss, and is formalized as:

$$MIS = \frac{1}{h}\sum_{j=1}^{h}((u_{t+j} - l_{t+j}) + \frac{2}{\alpha}(l_{t+j} - y_{t+j})1(y_{t+j} < l_{t+j}) + \frac{2}{\alpha}(y_{t+j} - u_{t+j})1(y_{t+j} > u_{t+j}))$$

Within our investigation, we fix our confidence level (α) at 95%, since if more than one confidence level were to be used the runtime complexity of computing MIS would multiply by n. 1 is an indicator function in this equation, where the function value should be treated as 1 when the inequality condition is true and 0 when the inequality condition is false.

This loss function is comprised of 3 sections. The first section produces a penalty the size of the distance between the upper and lower bounds of our predicted interval. The second section produces a penalty the size of the distance between the lower bound and the actual value, scaled by $\frac{2}{\alpha}$ when the actual value is less than that of the lower bound. The third part produces a penalty the size of the distance between the actual value and the upper bound scaled by $\frac{2}{\alpha}$ when the actual value is higher than the predicted value. Since the loss function of MIS regression jointly includes the upper and lower bounds, the result outputs both, unlike quantile regression.

3.3 SG-MCMC

The final model we examine is stochastic gradient Markov chain Monte Carlo. This form of gradient descent is useful in allowing us to calculate our quantiles according to subsets of the training data set which are selected based on the posterior distribution over the parameter space. Also, we follow the stochastic gradient thermostat method (SGNHT), whose purpose is to control gradient noise, which is typically characterized by heavy tails. We generate samples of model parameters θ as a function of our loss function L(θ), diffusion coefficients A, and learning rate h, in addition to auxiliary variables p ϵ $\mathbb{R}^d$ and ζ ϵ R. We randomly initialize θ, p, and ζ and update according to the rule:

$$\theta_{k+1} = \theta_k + p_k h$$
$$p_{k+1} = p_k - L(\theta)h - \zeta_k p_k h + \mathcal{N}(0, 2Ah)$$
$$\zeta_{k+1} = \zeta_k + \left(\frac{p_k^t p_k}{d} - 1\right) h$$

Where after the kth iteration, θ follows the distribution of the posterior. By running for multiple θ with different samples according to the posterior, we quantify the uncertainty of our prediction.

3.4 Conformal Prediction

In order to improve upon previous results, we introduce conformal prediction to our quantile regression model. Conformal prediction is useful because it guarantees coverage of our prediction interval at our chosen significance level. Using conformal prediction, we guarantee that our desired prediction lies within our confidence interval with a fixed level of confidence. Additionally, conformal prediction is well suited to our needs for several reasons. It works with any data, without distributional assumptions, it does not require analyzing or retraining, as data is fed into a neural network, and it costs relatively little in terms of computation.

In order to incorporate conformal prediction into the model, a calibration data set must be partitioned in addition to training, testing and validation sets. This is data which will be used only after the model is trained, so that the prediction intervals may be adjusted using conformal prediction before finally evaluating our results. The adjustments made to our original quantile prediction intervals by the conformal prediction process are done in a few steps. First, a distribution of predictions is created, where the predicted values from our quantile regression are scored based on the confidence of the prediction. Then, a threshold value is calculated for which at least 95%, or another fixed confidence level, of our data have a score for the correct interval above that value. This is done by taking the 1 - 95%, or 5%, quantile from the scored predictions. Lastly, our prediction interval is created by including values whose scores exceed this threshold value.

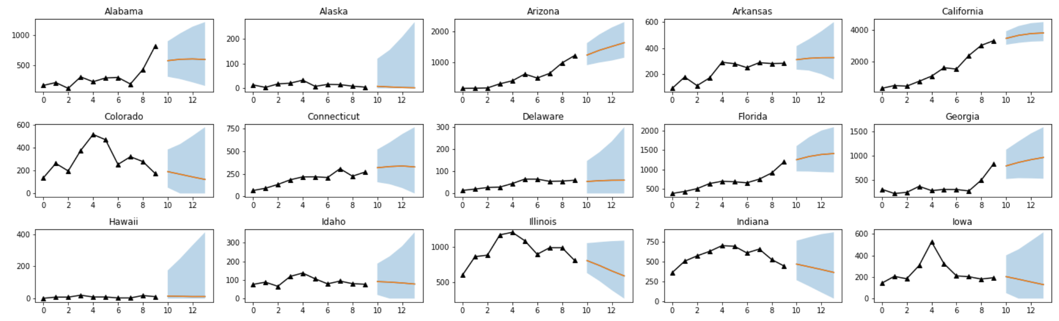
4 Model Implementation for torchTS

As part of our quarter 1 initiatives, we wanted to merge these uncertainty quantification methodologies into a usable suite of tools that focus on spatiotemporal analysis. TorchTS is a library built upon pytorch-lightning, a module that provides an organized framework for implementing custom models, which allows for it to provide specialized machine learning tools for spatiotemporal data.

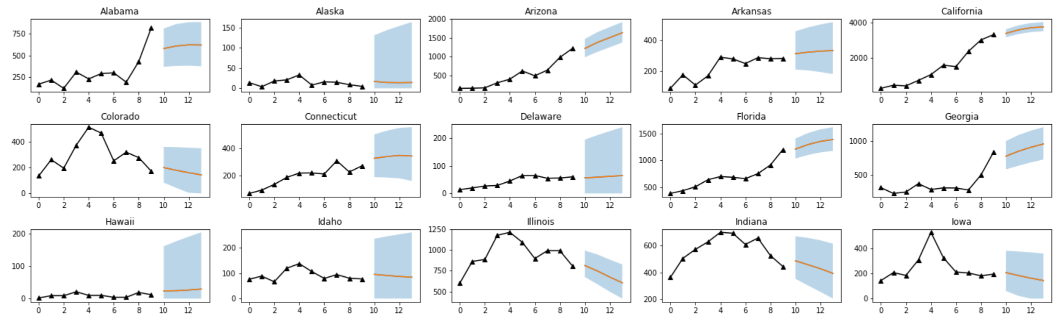
In quarter 1, we worked on an extension to the torchTS codebase that implements a quantile loss function, which provides uncertainty information in our forecasts by bounding our forecasts with models trained on 25%, 50%, and 75% confidence levels. The predictions from the 50% confidence level trained model provides the conventional prediction, while the 25% and 75% confidence level predictions serve as under and over-estimates that are represented in the variation in our model performance amongst various retrainings.

This quarter, we worked on further extend the torchTS by implementing conformal quantile regression. After the model trained on the original quantile regression model, we conformalized the predicted interval using the calibration set. First, we used our nonconformal predictor to predict the interval (upper bound and lower bound) for the calibration set, and then we calculated the difference between the true Y_i and our prediction and found the target residual value d as described in section 4.2. Finally, we updated our former prediction by adding d to previous upper bound and subtracting d from previous lower bound. The final result we obtained is the conformalized interval, which has the properties that it guarantee coverage with smallest length possible.

4.1 Quantile Regression Development



(a) Reproduced Quantile Regression Model



(b) Original Paper Quantile Regression Model

Figure 1: Quantile Regression Model Visualization Compression

During quarter 1, we replicated the results in SpatiotemporalUQ using quantile model. For the COVID19 dataset, we chose to follow the original uncertainty quantification paper by not predicting the death number, but forecasting the residual in order to better replicate the results and accuracy of the paper.

In our visualization, we picked the first 15 states from the United States in alphabetical order for comparison. For the Quantile regression model, we plot our results in Figure 1a.

By comparing with the original paper results in Figure 1b, we found out that our prediction from the trainings has the same overall shape as the original paper. However, our results appear to have a larger confidence interval. We hypothesize that this may be due to our insufficient training steps, and we will examine this by retraining using different patience.

4.2 Conformal Quantile Regression [CQR] Development

Our work during the second half of our research sought to add on to and improve our quantile regression from our prior work. To this end, we conformalize our quantile regression prediction. The expected result of adding conformal prediction to our model is a guarantee of valid coverage with high probability that prediction intervals hold the true value of a metric.

Conformal quantile regression can be implemented in a few different styles. Since we are focused on implementing code in our library that is able to run on computers with processing power that is increasingly accessible to the general public, we focus on *inductive conformal prediction (ICP)*, which is a type of conformal prediction that invokes a regression function a finite number of times. In order to perform ICP, we are required to split our training dataset equally into two new datasets: training and calibration [6].

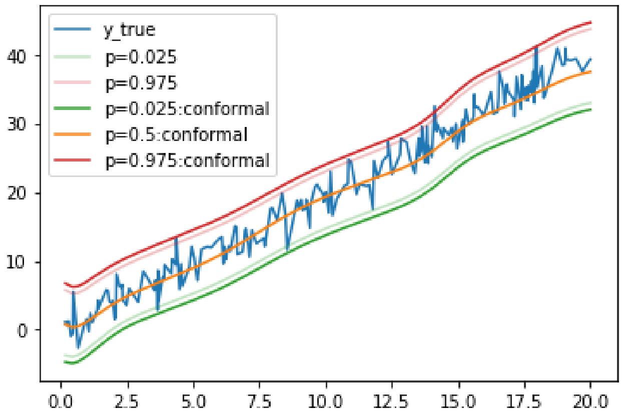
Once we obtain our new training dataset, we proceed to train our nonconformal predictor (NCP), in our case a quantile regressor, which trains a given model for an upper confidence bound (0.975 quantile), a median (0.5 quantile), and a lower confidence bound (0.025 quantile). These construct the confidence band that we will build on in our conformal quantile regressor. It is important to note that for this step, we have specified a fixed desired confidence level of 95%, which is where we get the range of (0.025, 0.975) for our confidence band.

Following the inital NCP training to obtain a confidence band, we perform the first step of the conformalization process. The conformalization process involves us using our new calibration set to update the NCP-predicted calibration bands to achieve enough coverage to satisfy a specified confidence level. We begin by calculating the difference between the true Y_i from our new calibration set and each of the upper and lower bands obtained through our NCP, which are the residuals of our model's predictions. For each entry of our calibration set, we select the smaller of the two residuals: the upper band residual, or the lower band residual. We then sort our selected residuals in descending order and find the residual where $1 - \alpha$ of the values pertaining to each entry in our calibration set are larger than it. We take this residual value, which we will call d, and use it to adjust our predicted NCP confidence bands by subtracting it from our former lower band, L_{prev} and add it to our former upper band, U_{prev} to obtain

$$L_{updated} = L_{prev} - d$$
$$U_{updated} = U_{prev} + d$$

This split conformal process provides conformalization with the performance cost equivalent to fitting our NCP and provides us with aforementioned coverage guarantee [4].

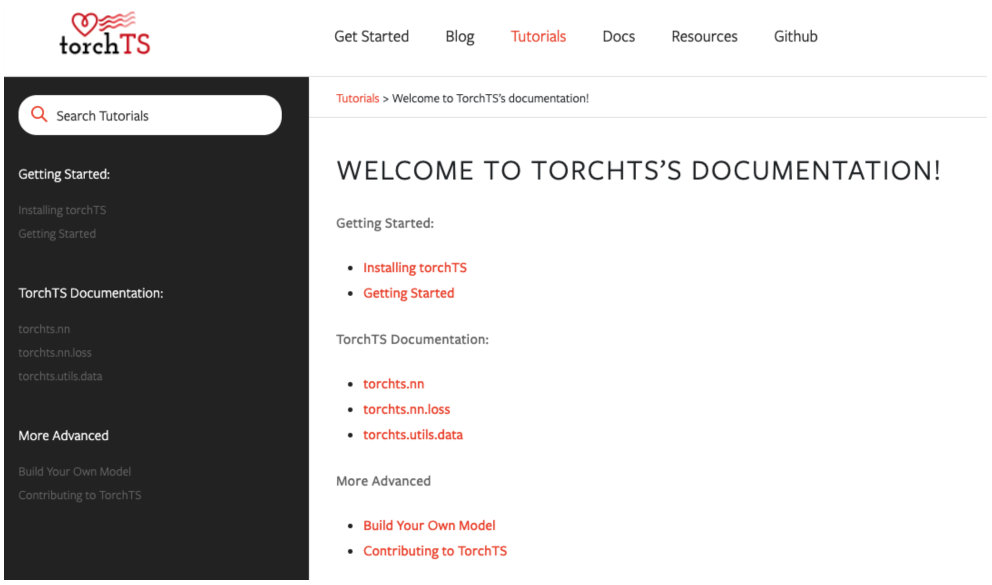
Our resulting conformal quantile regression code before adaptation to the respective torchTS format provides us with modified bands that guarantee 95% coverage when trained and predicted over a randomly generated trend line with noise.



(a) Conformal Prediction on Random Data

5 TorchTS' User Experience Enhancement

Users can now easily view the library's API documentation on the official website. We used Sphinx to auto-generate torchTS API documentation based on class and function docstrings with pytorch-theme. We also used Docusaurus V2 to easily generate torchTS' official website. Then, we merged Sphinx into Docusaurus V2 so that users can view both torchTS tutorials and API docs on the official website. We implemented customized Github Action workflows such that every commit to the main branch will trigger a new build for the package and the official website, and thus, every PR merge will automatically update our API docs and tutorials.



Tutorials > Welcome to TorchTS's documentation!

WELCOME TO TORCHTS'S DOCUMENTATION!

Getting Started:

- Installing torchTS
- Getting Started

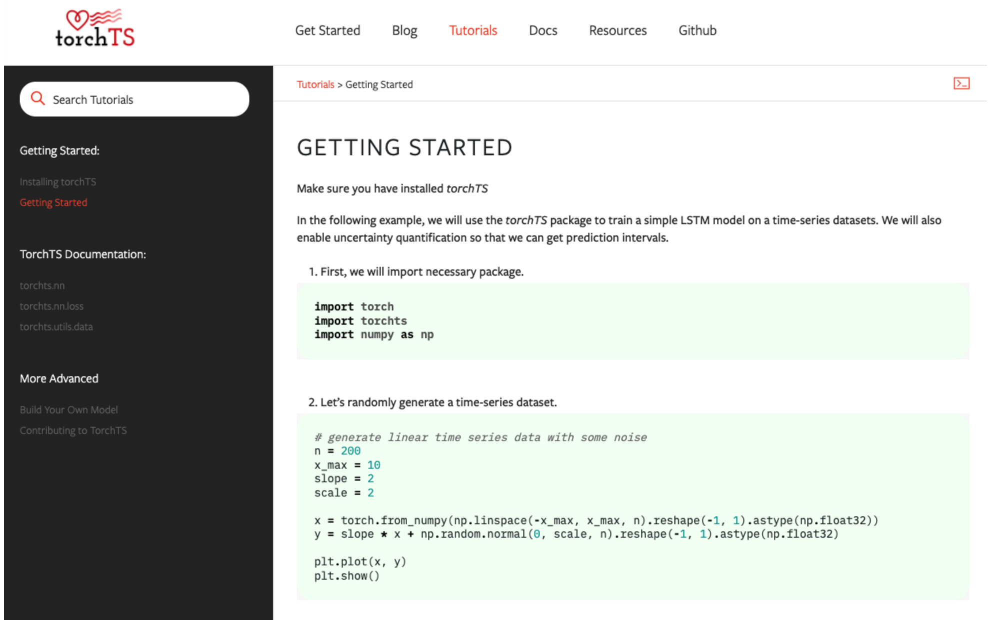
TorchTS Documentation:

- torchts.nn
- torchts.nn.loss
- torchts.utils.data

More Advanced

- Build Your Own Model
- Contributing to TorchTS

Also, we are working on the Getting Started section so that users can quickly understand how to use this library and how they can benefit from it.



Tutorials > Getting Started

GETTING STARTED

Make sure you have installed *torchTS*

In the following example, we will use the *torchTS* package to train a simple LSTM model on a time-series datasets. We will also enable uncertainty quantification so that we can get prediction intervals.

1. First, we will import necessary package.

```
import torch
import torchts
import numpy as np
```

2. Let's randomly generate a time-series dataset.

```
# generate linear time series data with some noise
n = 200
x_max = 10
slope = 2
scale = 2

x = torch.from_numpy(np.linspace(-x_max, x_max, n).reshape(-1, 1).astype(np.float32))
y = slope * x + np.random.normal(0, scale, n).reshape(-1, 1).astype(np.float32)

plt.plot(x, y)
plt.show()
```

6 Discussion and Future Work

An important aspect of maintaining and growing the torchTS library is to make it easy for others to contribute code to the repository. TorchTS is built with the intention of making spatiotemporal analysis more accessible and open. In order to do this, more descriptive docstrings and more guides on contributing would greatly improve the quality of users' experiences. Additionally, as noted above there are many approaches to the same problem such as with uncertainty quantification. TorchTS would greatly benefit from the option of choosing from a variety of different model implementations, while maintaining the elegance of making methods simple and intuitive for the common user. We hope that the groundwork laid within documentation efforts and implementations of uncertainty quantification helps in guiding the project to becoming more widely utilized in the future.

7 Contribution

We first brainstormed the project together and performed individual studies and experiments. Next, we delegated the work based on personal interests and specializations. Miles, Derek, and Judy contributed to most of the research study and coding on conformal quantile regression. Kai contributed to most of the documentation-keeping and making the tutorial more user-friendly. We also helped each other with other parts to accelerate the process.

8 Acknowledgements

Our contributions to the TorchTS library were supported by Rose Yu who provided domain knowledge and expertise that greatly assisted in the research, development, and guidance driving our efforts. We thank Kevin Lane for their assistance and feedback towards developing and pushing code into the torchTS library. We would also like to show our gratitude to Allen Wu for their help with our understanding of various aspects of Uncertainty Quantification. Lastly, we'd like to thank Aaron Fraenkel for his direction and information during the course of this research project.

References

Paul Fearnhead Christopher Nemeth. Stochastic gradient markov chain monte carlo. 2019.

Rana A. Moyeed Keming Yu. Bayesian quantile regression. 2001.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NIPS*, 2017.

Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J. Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression, 2017.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting, 2018.

Yaniv Romano, Evan Patterson, and Emmanuel J. Candès. Conformalized quantile regression. In *NeurIPS*, 2019.

Chen Xu and Yao Xie. Conformal prediction interval for dynamic time-series. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11559–11569. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/xu21h.html`.

Yanrong Yang Yuan Gao, Han Lin Shang. High-dimensional functional time series forecasting: An application to age-specific mortality rates. 2018.