Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis International Bachelor Econometrics and Operations Research

Discovering Latent Dependencies: Boosting the Nonexchangeable Conformal Prediction Paradigm

Timothy Lauren Nijhuis (611398)

Supervisor:	Meer, S van
Second assessor:	Hemerik, JBA
Date final version:	1st July 2024

The views stated in this thesis are those of the author and not necessarily those of the supervisor, second assessor, Erasmus School of Economics or Erasmus University Rotterdam.

Abstract

This study advances the literature on conformal prediction for time series data through a two-stage approach. First, we address the considerations that go into selecting an appropriate forecaster for conformal prediction in the context of time series. Specifically, a Bayesian Structural Time Series (BSTS) model is evaluated against several traditional forecasting methods. Second, we propose a novel technique to determine test-sample adaptive weights in nonexchangeable conformal prediction. Our approach leverages a Long Short-Term Memory (LSTM) network to estimate the conditional distributions of nonconformity scores. We then compute the Kullback-Leibler divergences between these estimated distributions and process them through a kernel function. This method has been tested and benchmarked across six different simulated settings. Our findings show that the proposed method generalizes well and autonomously identifies the relevant latent dependencies between the nonconformity scores. This significant contribution enhances the validity and reliability of nonexchangeable conformal prediction for time series data.

1 Introduction

Consider the scenario of forecasting electricity demand using a black-box machine learning model. The modeler is tasked with finding predictive intervals for tomorrow's electricity demand with a $1-\alpha$ probability, such that the electricity demand of households can be met with a specified confidence level. To achieve this, the modeler turns to conformal prediction, a technique that is compatible with any model and requires minor assumptions about the distribution of electricity demand.

Conformal prediction is a technique that constructs prediction intervals with guaranteed coverage for any model. Given a dataset of n points $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$ and a new data point (X_{n+1}, Y_{n+1}), conformal prediction calculates nonconformity scores $R_i = |Y_i - \hat{\mu}(X_i)|$ using the model's predictions $\hat{\mu}(X_i)$. Here, $\hat{\mu}$ represents the fitted model, X denotes the features, and Y is a real-valued target variable. These scores are then used to form a prediction interval $\hat{C}(X_{n+1})$ that includes Y_{n+1} with the specified probability $1-\alpha$. This ensures a marginal coverage guarantee: $P(Y_{n+1} \in \hat{C}(X_{n+1})) \geq 1-\alpha$.

The modeler uses this technique to create the desired prediction intervals $\hat{C}(X_{n+1})$ for electricity demand. However, they find that the prediction intervals do not achieve the target coverage level. This is because the required assumption for marginal coverage is the exchangeability of the nonconformity scores, which is violated in time series data. In practice, tomorrow's electricity demand can depend on trends, seasonal patterns, and autoregressive components. These dependencies lead to a coverage gap: $(1-\alpha) - P\{Y_{n+1} \in \hat{C}(X_{n+1})\}$, indicating the difference between the intended and actual coverage level.

To minimize the coverage gap, the modeler must enhance the exchangeability of the nonconformity scores. This can be achieved in two stages. First, selecting an appropriate forecaster that accounts for temporal dependencies such as trends, seasonal patterns, and autoregressive components will reduce uncaptured dependencies in the nonconformity scores. Second, the conformal prediction procedure can be improved by weighting the nonconformity scores based on their similarity to the test point.

In this paper, the first stage is addressed by using a Bayesian Structural Time Series (BSTS) model to forecast electricity demand. The BSTS model combines simpler Bayesian time series models to capture changing slopes, trends, and seasonal effects. Specifically, the model's time-dependent coefficients evolve according to a random walk with variance determined by prior beliefs, such that coefficients that better explain the data are considered more likely. This allows the BSTS model to adapt to distributional changes over time without needing to refit the model or discard samples. This makes the model particularly appropriate for handling data with evolving temporal dependencies.

The second stage is addressed by employing a Long Short-Term Memory (LSTM) network to weigh nonconformity scores. The LSTM estimates the moments of the conditional nonconformity score distribution for the test point using the features X_{n+1} and calculates the Kullback-Leibler (KL) divergence between this distribution and those of the sample points. These KL divergences serve as the distance metric for a kernel function that assigns adaptive weights to the utilized nonconformity scores. This method improves the validity of conformal prediction in the presence of distribution drift, changepoints, and conditional heteroskedasticity.

The rest of the paper is structured as follows. Section 2 reviews the literature on conformal prediction, emphasizing test-sample adaptive weighting schemes. Section 3 describes the data used to test the hypotheses: the BSTS model is applied to an electricity demand dataset in the first stage, while the LSTM model provides weights under six simulated datasets in the second stage. Section 4 details the methodology, including the mathematical underpinnings and the motivations for using the BSTS and LSTM models. Section 5 presents the numerical results and discusses their practical implications. Finally, Section 6 summarizes the main findings and offers suggestions for future research.

2 Literature Review

Conformal prediction techniques vary in their sample efficiency, with more efficient methods also being more computationally intensive. Nonconformity scores, derived from point forecasts, need to be exchangeable. Exchangeability is violated when nonconformity scores are more likely to belong to specific observations rather than being equally likely for all observations. For instance, using the absolute residual as the nonconformity measure, constructing prediction intervals for out-of-sample predictions based on in-sample predictions violates exchangeability. This is because the residual for X_{n+1} is more likely to belong to observation $n+1$ than to observations $1 : n$, as in-sample and out-of-sample residuals typically follow different distributions.

Therefore, one approach to conformal prediction, called split conformal prediction, involves using a holdout set to measure residuals and then taking an empirical quantile from these holdout residuals. Another method, known as full conformal prediction, hypothesizes a value for the test point's target variable, fits the model on the entire dataset, and treats each point as an in-sample prediction. However, this requires fitting the model for each possible value of Y_{n+1}, which is feasible for only a small set of models. Both of these methods were first comprehensively detailed in the book "Algorithmic Learning in a Random World" (Vovk et al., 2005).

Since then, algorithms that bridge these two extremes have been developed using a cross-validation framework. For example, the Jackknife+ constructs sample-efficient prediction intervals using leave-one-out predictions (Barber, Candes et al., 2020). Angelopoulos & Bates (2022) give a comprehensive introduction to the current field of conformal prediction and its methods. Additionally, Vovk et al. (2022) published an updated version of the aforementioned book, including improved algorithms and extensions.

The central assumption for these methods to be valid is that the nonconformity scores are exchangeable. However, this paper focuses on time series data where this condition does not hold. In Barber et al. (2023), a weighting method is proposed to deal with such nonexchangeable data. Here, each data point (X_i, Y_i) is assigned a weight w_i that relates to our prior expectations of its exchangeability with the test point. The prediction interval is then calculated using the weighted quantile $Q_{1-\alpha}\left(\sum_{i=1}^{n} w_i \delta_{R_i} + w_{n+1}\delta_{\infty}\right)$. This weighting method provides approximate coverage guarantees despite the violation of exchangeability.

These methods for conformal predictions aim to provide marginal coverage guarantees, meaning that coverage is obtained in the long run when averaging out all observations. In this paper, we seek to enhance the validity of conformal prediction by assigning higher weights to more exchangeable observations, such that these marginal coverage guarantees are more likely to hold.

This approach is connected to the literature on conditional coverage guarantees for conformal prediction. By making the weights adaptive to the test sample, we effectively implement a soft form of conditioning.

In this domain of the literature, it has been proven that it is not possible to construct finite-width prediction intervals that offer conditional coverage guarantees of the form

$$\mathbb{P}\left\{Y_{n+1} \in \hat{C}_n(X_{n+1}) \mid X_{n+1} = x\right\} \geq 1-\alpha, \qquad \text{for all distributions } P \text{ and almost all } x$$

when dealing with finite samples, without assuming smoothness of the distributions P (Lei & Wasserman, 2012; Vovk, 2012). However, Barber, Candès et al. (2020) find that by relaxing the conditional coverage definition to

$$\mathbb{P}\left\{Y_{n+1} \in \hat{C}_n(X_{n+1}) \mid X_{n+1} \in \mathcal{X}\right\} \geq 1-\alpha, \qquad \text{for all distributions } P \text{ and all } \mathcal{X} \subseteq \mathbb{R}^d \text{ with } P_X(\mathcal{X}) \geq \delta$$

nontrivial finite-width prediction intervals can be found. An intuitive 'soft conditioning' method related to this relaxed conditional coverage definition is localized conformal prediction, which assigns higher weights to samples within a local region around the test sample using a kernel function (Guan, 2022).

A more advanced method for finding weights, similar to the Long Short-Term Memory (LSTM) network procedure proposed in this paper, is described by Auer et al. (2023). In their approach, a Modern Hopfield Network (MHN) is used to identify segments of the time series with similar conditional absolute residual distributions. This technique reweights samples based on their similarity to the current test sample, effectively conditioning on identified 'error regimes.' Again, the focus is not on the conditional coverage guarantees; yet, state-of-the-art performance is achieved in both coverage and predicted interval widths through this soft conditioning.

The MHN encodes features and determines similarity to stored feature representations, which are processed through a softmax function to establish association strengths. In contrast, this paper uses an LSTM network to make point predictions for the moments of the conditional nonconformity score distribution. It then calculates the Kullback-Leibler (KL) divergences between the test sample's conditional distribution and those of the other samples. Finally, a kernel transforms these KL divergences into the utilized weights.

These adaptive reweighting methods can be further improved by superposing a method that adaptively changes the coverage target based on the actual miscoverage level relative to the desired coverage. The first method to do this, developed by Gibbs & Candes (2021), simply increases the coverage target after a miscoverage event and decreases it after a coverage event.

The most recent improvement in adaptive target selection is described by Angelopoulos et al. (2023), where a Proportional-Integral-Derivative (PID) controller is applied to conformal prediction. Their derivative control component addresses systematic trends in the nonconformity scores that were not captured by the initial forecaster. The LSTM network in this paper can be viewed through the lens of this 'scorecasting' component. However, the integration of the scorecaster here differs, as it does not influence the internal target coverage but rather the weighting scheme of the nonconformity scores.

3 Data

This section details the data used for the experiments. The first part covers the Elec2 dataset, which is utilized to fit the Bayesian Structural Time Series (BSTS) model (Harries, 1999). This dataset is also used to replicate the findings of Barber et al. (2023), who evaluated three methods for full conformal prediction in the context of nonexchangeable time series data. They compared the performance of a Least Squares (LS) model with both normal and nonexchangeable algorithms, and a Weighted Least Squares (WLS) model with the nonexchangeable algorithm.

The second part describes the Data-Generating Process (DGP) for six different simulated datasets, each corresponding to a unique scenario. These datasets are employed to evaluate the performance of the LSTM network in providing test-sample adaptive weights. Each scenario follows a distinct process, enabling us to assess how well the LSTM network can generalize across various settings.

3.1 Elec2 Dataset

The Elec2 Dataset is taken from Kaggle (The Elec2 Kaggle Link). The dataset is normalized to be in the range $[0, 1]$ and contains 45,312 instances from 7 May 1996 to 5 December 1998. Each instance represents a 30-minute period, with 48 instances per day. The records include the New South Wales electricity demand *nswdemand* and price *nswprice*, the Victoria electricity demand *vicdemand* and price *vicprice*, and the scheduled electricity transfer between these states *transfer*. Other features present in the dataset are dropped in this paper. For all models, the initial stretch of time ($t = 0 : 17760$) is removed because the target variable *transfer* is constant during this period.

To replicate the findings of Barber et al. (2023), we use their methods on a subset of the data from 9:00 AM to 12:00 PM. This was also done in their paper to minimize the presence of hourly seasonal effects. For fitting the BSTS model, we use the full Elec2 dataset because we are specifically interested in how well this forecaster can capture dependencies that make the data nonexchangeable, such as seasonal effects. An exposition of both the hourly and daily seasonal effects that are present in the dataset is given in Appendix A

3.2 Simulated Data

The first three scenarios we consider to evaluate the performance of the LSTM network are based on Barber et al. (2023), providing a reliable benchmark. Then, this paper introduces three additional scenarios to check the robustness of the LSTM network and its ability to generalize to different scenarios. These new scenarios are influenced by highly interpretable latent processes, as illustrated in Appendix B. Consequently, we recommend incorporating these scenarios in future research on test-sample adaptive weights. For each scenario, 3000 observations have been generated.

- **Setting 1: i.i.d. data.** The data points (X_i, Y_i) are generated i.i.d, with $X_i \sim \mathcal{N}(0, I_4)$ and $Y_i \sim X_i^T \beta + \mathcal{N}(0, 1)$ for a coefficient vector $\beta = (3, 1, 0, 0)$.

- **Setting 2: changepoints.** The data points (X_i, Y_i) are generated with $X_i \sim \mathcal{N}(0, I_4)$ and $Y_i \sim X_i^T \beta^{(i)} + \mathcal{N}(0, 1)$. The coefficient vector $\beta^{(i)}$ changes three times throughout the series.

$$\begin{aligned}
\beta^{(1)} &= \ldots = \beta^{(500)} = (3, 1, 0, 0), \\
\beta^{(501)} &= \ldots = \beta^{(1500)} = (0, -3, -1, 0), \\
\beta^{(1501)} &= \ldots = \beta^{(2500)} = (0, 0, 3, 1), \\
\beta^{(2501)} &= \ldots = \beta^{(3000)} = (0, 3, 1, 0),
\end{aligned}$$

- **Setting 3: distribution drift.** The data points (X_i, Y_i) are generated with $X_i \sim \mathcal{N}(0, I_4)$ and $Y_i \sim X_i^T \beta^{(i)} + \mathcal{N}(0, 1)$. Here, $\beta^{(i)}$ is the coefficient vector at time i and changes gradually through linear interpolation between the starting and ending coefficients.

$$\begin{aligned}
\beta^{(1)} &= (3, 1, 0, 0), \\
\beta^{(3000)} &= (0, 0, 3, 1),
\end{aligned}$$

- **Setting 4: stochastic volatility.** The data points (X_i, Y_i) are generated with $X_i \sim \mathcal{N}(0, I_4)$ and $Y_i \sim X_i^T \beta + \mathcal{N}(0, \sigma_i^2)$. Here, $\beta = (3, 1, 0, 0)$ is the coefficient vector, and σ_i^2 represents the stochastic volatility at time i, which evolves according to the exponent of the latent state h_i that follows an AR(1) process.

$$\sigma_i^2 = \exp(h_i), \qquad\qquad h_i = 0.9 h_{i-1} + \mathcal{N}(0, 0.25)$$

- **Setting 5: heteroskedasticity.** The data points (X_i, Y_i) are generated with $X_i \sim \mathcal{N}(0, I_4)$ and $Y_i \sim X_i^T \beta + \mathcal{N}(0, \sigma_i^2)$. Here, $\beta = (3, 1, 0, 0)$ is the coefficient vector, and σ_i^2 depends on a linear combination of X_{i2} and X_{i4}.

$$\log(\sigma_i^2) = \alpha_0 + \alpha_1 X_{i2} + \alpha_2 X_{i4}$$

  where $\alpha_0 = 0.1$, $\alpha_1 = 0.4$, and $\alpha_2 = 0.3$.

- **Setting 6: lagged effects.** The data points (X_i, Y_i) are generated with $X_i \sim \mathcal{N}(0, I_4)$ and $Y_i \sim X_i^T \beta + \mathcal{N}(0, \sigma_i^2)$. Here, $\beta = (3, 1, 0, 0)$ is the coefficient vector. The variances are initialized as follows:

$$\begin{aligned}
\sigma_{(1)}^2 &= \ldots = \sigma_{(10)}^2 = 3.5 && \text{(very high initial variance)} \\
\sigma_{(10)}^2 &= \ldots = \sigma_{(30)}^2 = 1 && \text{(low initial variance)} \\
\sigma_{(30)}^2 &= \ldots = \sigma_{(45)}^2 = 0.5 && \text{(very low initial variance)} \\
\sigma_{(45)}^2 &= \ldots = \sigma_{(50)}^2 = 2 && \text{(high initial variance)}
\end{aligned}$$

For the time steps beyond the initial 50 steps, the variance σ_i^2 is determined by:

$$\log(\sigma_i^2) = 0.95 \log(\sigma_{i-50}^2) + \mathcal{N}(0, 0.15),$$

Introducing lagged effects where the variance at any point depends on the variance from 50 steps earlier, modified by a normally distributed noise term.

4 Methodology

The methodology of this paper is structured into three main sections. The first section outlines the two conformal prediction techniques utilized in this paper: split conformal prediction and full conformal prediction. Next, the second section details the Bayesian Structural Time Series (BSTS) model. It includes both an introduction to the BSTS as an appropriate forecaster for time series data with changing temporal dependencies and a rigorous exposition of its mathematical structure. The third and final part of the methodology pertains to the Long Short-Term Memory (LSTM) network, the chosen model to estimate the moments of the conditional nonconformity score distributions. This section begins with a motivation for using test-sample adaptive weights, followed by an in-depth exposition of the LSTM model architecture. It concludes with the method for obtaining weights using the Kullback-Leibler (KL) divergences between the estimated distributions.

4.1 Conformal Prediction Methods

This section details the two main methods for conformal prediction that are used in this paper: split conformal prediction and full conformal prediction (Vovk et al., 2005). Both methods can be adapted for nonexchangeable data, but we specifically choose to use the split conformal method for exchangeable data with the BSTS model. This is because the resulting coverage gap can then be interpreted as a performance metric that indicates how effectively the BSTS model transforms nonexchangeable data into exchangeable nonconformity scores.

Three versions of the full conformal prediction method are used: for exchangeable data, for nonexchangeable data, and for nonexchangeable data using a nonsymmetric algorithm. The latter two methods were proposed in Barber et al. (2023). Full conformal prediction is feasible for only a limited set of models, including Weighted Ridge Regression (WRR), which also encompasses Weighted Least Squares (WLS) and Least Squares (LS).

Two algorithms for these versions of full conformal prediction are detailed in the appendix. In Appendix C, an extension of the algorithm described in Vovk et al. (2005) is presented, with a time complexity of $\mathcal{O}(n^2)$. An improved algorithm, based on Vovk et al. (2022), is detailed in Appendix D, which achieves a reduced time complexity of $\mathcal{O}(n \log n)$.

4.1.1 General Notation

For both methods, we are given the data points $Z_i = (X_i, Y_i)$, where X_i represents the feature vector and Y_i is the real-valued target variable for $i = 1, \ldots, n$, we aim to construct a prediction interval $\hat{C}_n(X_{n+1})$ for a new test sample (X_{n+1}, y) with an unknown y. This prediction interval should satisfy the condition $\mathbb{P}\left\{Y_{n+1} \in \hat{C}_n(X_{n+1})\right\} \geq 1-\alpha$, where $1-\alpha$ is the desired target coverage. To achieve this, we utilize a fitted regression function $\hat{\mu} : \mathcal{X} \to \mathbb{R}$, which maps from the predictor space $\mathcal{X}$ to a real-valued output.

A nonconformity function $S(X_i, Y_i)$ is chosen that measures how unusual a data point is relative to the other samples. Here, we choose to work with the absolute residuals $s_i = |Y_i - \hat{\mu}(X_i)|$ as nonconformity scores. These scores are then used to construct a prediction interval by taking an empirical quantile $Q_{1-\alpha}$ of the scores at the level of the desired target coverage.

This works because if the exchangeability of the nonconformity scores is assumed, each score is equally likely to belong to any observation, including the test sample. Consequently, the rank of the test sample's nonconformity score is uniformly distributed, ensuring that the empirical quantile at level $1-\alpha$ has an α probability of being exceeded.

4.1.2 Split Conformal Prediction

Let $\hat{\mu} : \mathcal{X} \to \mathbb{R}$ be a forecasting model fitted on an initial subset of the data. Now, define an additional holdout set $\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$, which was not used to train the fitted model. Let δ_a denote the the point mass at a. The prediction interval $\hat{C}_n(X_{n+1})$ for the test point with features X_{n+1} is now given as.

$$\hat{C}_n(X_{n+1}) = \hat{\mu}(X_{n+1}) \pm Q_{1-\alpha}\left(\sum_{i=1}^{n} \frac{1}{n+1} \cdot \delta_{s_i} + \frac{1}{n+1} \cdot \delta_{+\infty}\right)$$

4.1.3 Full Conformal Prediction

The method for full conformal prediction with nonexchangeable data and a nonsymmetric algorithm is described here. To adapt this method for full conformal prediction with exchangeable data, set the weights for each observation to one. For full conformal prediction with a symmetric algorithm, ensure all tags are set to one.

A concise description of the method will be given, for a more detailed exposition refer to Barber et al. (2023). Let $\mathcal{A}$ be an algorithm that maps a sequence of "tagged" data points $(X_i, Y_i, t_i) \in \mathcal{X} \times \mathbb{R} \times \mathcal{T}$ to a fitted regression function $f(X)$. Here, $\mathcal{X}$ denotes the predictor space and $\mathcal{T}$ is the tag space.

$$\mathcal{A} : \bigcup_{n \geq 0} (\mathcal{X} \times \mathbb{R} \times \mathcal{T})^n \to \{\text{measurable functions } \hat{\mu} : \mathcal{X} \to \mathbb{R}\},$$

The tags t_i can serve multiple purposes. In the context of fitting a WLS model, they denote the observation weights. Let $w_1, \ldots, w_n \in [0, 1]$ be the initial weights assigned to the nonconformity scores. Then, construct normalized weights as follows.

$$\tilde{w}_i = \frac{w_i}{w_1 + \cdots + w_n + 1}, \quad i = 1, \ldots, n, \quad \text{and} \quad \tilde{w}_{n+1} = \frac{1}{w_1 + \cdots + w_n + 1}.$$

These normalized weights will be used for both the swap step and for taking the empirical quantile at the target coverage level. Define π_k as the permutation swapping the tags k and $n+1$ and denote the vector of target variables with y unknown as.

$$Y_i^y = \begin{cases} Y_i, & i = 1, \dots, n, \\ y, & i = n+1. \end{cases}$$

Then, a prediction interval $\hat{C}_n(X_{n+1})$ is constructed for the test point using Algorithm 1.

Algorithm 1 General Full Conformal Prediction

```
Input:
    Observed data $Z_{1:n}$, $X_{n+1}$
    Miscoverage level $\alpha$
Draw an index $k$, where $K \sim \sum_{i=1}^{n+1} \tilde{w}_i \cdot \delta_i$
for each $y \in \mathbb{R}$ do
    Fit model $\hat{\mu} = \mathcal{A}\left(\{(X_{\pi_k(i)}, Y^y_{\pi_k(i)}, t_i) : i \in [n+1]\}\right)$
    Compute the nonconformity scores $s_{i:n+1} = S(X_{\pi_k(i)}, Y^y_{\pi_k(i)})_{1:n+1}$
end for
Return:
    $\hat{C}_n(X_{n+1}) = \left\{y : s_{n+1} \leq Q_{1-\alpha}\left(\sum_{i=1}^{n+1} \tilde{w}_i \cdot \delta_{s_i}\right)\right\}$
```

4.2 Bayesian Structural Time Series

In this section, a detailed account of the employed Bayesian Structural Time Series (BSTS) model will be provided. It will also be argued that this model is particularly suitable for conformal prediction using time series data with evolving temporal dependencies. The BSTS model is constructed as a sum of simpler Bayesian state-space models, each designed to capture a specific component of the process dynamics. These components include changing slopes, trends, and seasonal effects.

The flexibility of the BSTS model comes from its use of time-dependent coefficients that evolve according to a random walk. Through a Kalman Filtering algorithm, the coefficients that make the data more likely are given a higher probability. To illustrate the benefit of this approach, consider a simple example of a time series where the slope reverses midway.

$$y_t = \begin{cases} \mu + \beta t & \text{for } t \leq \frac{T}{2} \\ \mu + \beta(T - t) & \text{for } t > \frac{T}{2} \end{cases}$$

Non-state space models, such as traditional linear regression, assume that the coefficients are constant over time and would yield a β coefficient of zero in this scenario. The absolute residuals of such a model would therefore be affected by this unmodeled dependency shift. These types of unmodeled dependencies that remain in the nonconformity scores nullify the validity of conformal prediction. This makes the use of traditional models with constant coefficients an ill-informed choice for conformal prediction in time series contexts with potentially changing dependencies.

4.2.1 The BSTS Model

The BSTS model is defined as a linear Gaussian system, where the level, slope, and seasonal effects are assumed to be latent states (Harvey, 1990; Scott & Varian, 2014). Additionally, the model includes a regression component, which is simply an OLS regression. This component should be viewed as an augmentation to the model rather than a core part of the Bayesian framework. The mathematical equations of the BSTS model are given below.

$$\begin{aligned}
y_t &= \mu_t + \gamma_{t,i}^{(d)} + \gamma_{t,j}^{(h)} + \beta^\top x_t + \varepsilon_{1,t} && \text{(Observation Regression)}\\
\mu_t &= \mu_{t-1} + \delta_{t-1} + \varepsilon_{2,t} && \text{(Semi-Local Linear Trend)}\\
\delta_t &= D + \phi(\delta_{t-1} - D) + \varepsilon_{3,t} && \text{(Semi-Local Linear Trend)}\\
\gamma_{t,i}^{(d)} &= -\sum_{d \in D \setminus i} \gamma_{t-1,d}^{(d)} + \varepsilon_{4,t} && \text{(Day-of-Week Effect)}\\
\gamma_{t,j}^{(h)} &= -\sum_{h \in H \setminus j} \gamma_{t-1,h}^{(h)} + \varepsilon_{5,t} && \text{(Hour-of-Day Effect)}
\end{aligned}$$

For the Semi-Local Linear Trend, it holds that the coefficient μ_t is a level term that changes as a random walk plus a slope coefficient δ_t. The slope δ_t follows an AR(1) process with the AR parameter ϕ and a potentially non-zero mean D.

The Seasonal Effects are captured through the coefficients $\gamma_{t,i}^{(d)}$ for the day of the week $i \in \mathcal{D}$ (set of 7 days) and $\gamma_{t,j}^{(h)}$ for the hour of the day $j \in \mathcal{H}$ (set of 24 hours). Finally, β is the vector of fitted regression coefficients, with x_t being the observed covariates. Each process also includes a Gaussian noise term.

4.2.2 The Likelihood

For the linear Gaussian system, the interest is in finding a distribution over the model parameters given the observed data. This distribution is given by Bayes' Rule.

$$p(\theta \mid Y = y) = \frac{p(Y = y \mid \theta)p(\theta)}{p(y)}$$

This distribution of the model parameters given the observed data is referred to as the posterior. The posterior reflects the adjusted beliefs over the model parameters after having seen the data. That is, the proper Bayesian first defines the probability $p(\theta)$, also called the prior, that reflects the probability distribution of the model parameters prior to having seen any of the data. The initialization of these prior distributions is explained in Appendix E.

Given these prior beliefs, encapsulated as probabilities over the model parameters, the likelihood of observing the data $p(Y = y \mid \theta)$ is calculated. The final term $p(y)$ is the marginal and reflects the probability of observing the data after having integrated out all the latent states.

The likelihood $p(Y = y \mid \theta)$ is the probability of observing the data given the model parameters. To demonstrate the analytical computation of this likelihood, the exposition of Yi (2019) will be closely followed. First, the likelihood can be broken down as a chain of conditional probabilities.

$$\begin{aligned} p(Y = y \mid \theta) &= p(y_1, y_2, \ldots, y_n \mid \theta) \\ &= p(y_1|\theta)\, p(y_2|y_1, \theta)\, p(y_3|y_{1:2}, \theta) \cdots p(y_n|y_{1:n-1}, \theta) \end{aligned}$$

In the linear Gaussian system, we know that each of these y_t variables is normally distributed. Moreover, based on the transition and observation equations defined in Appendix F, we can model their inductive relationship via a two-step procedure. First, the update step computes the probability distribution of the next state estimate given the current state estimate.

$$\begin{aligned} p(z_{t-1}|y_{1:t-1}, \theta) &= \mathcal{N}(\mu_{t-1}, \Sigma_{t-1}) \\ p(z_t|y_{1:t-1}, \theta) &= \mathcal{N}(A\mu_{t-1} + b, A\Sigma_{t-1}A^\top + \Sigma_w) \end{aligned}$$

Since we have initialized the states z_0 as Gaussian, all subsequent states will also be Gaussian when we use the inductive step defined above. Then, since we have solved for the distribution of the state z_t we can use the observation equation for the prediction step. This allows us to write down the distribution of the observed variable y_t.

$$p(y_t|y_{1:t-1}, \theta) = \mathcal{N}(H(A\mu_{t-1} + b), H(A\Sigma_{t-1}A^\top + \Sigma_w)H^\top + \Sigma_v)$$

This algorithm, known as the Kalman filter (Kalman, 1960), underpins the implementation of the Bayesian Structural Time Series (BSTS) in TensorFlow (Abadi et al., 2015), which we use to fit the BSTS model in Python.

To estimate the posterior, a Variational Inference (VI) procedure was chosen. VI is often faster than Hamiltonian Monte Carlo (HMC) for high-dimensional problems. However, whether HMC might have been more effective for this specific problem has not been tested. A detailed description of the VI procedure is provided in Appendix G

4.3 Long Short-Term Memory

This section elucidates the application of the Long Short-Term Memory (LSTM) network to determine test-sample adaptive weights. It starts with a motivation for using test-sample adaptive weights within the nonexchangeable conformal prediction framework. Next, it outlines the mathematical framework of the LSTM and its suitability for finding latent dependencies in the Data-Generating Process (DGP).

Finally, the complete approach for determining the test-sample adaptive weights is detailed. The proposed method parametrizes a distribution for each conditional nonconformity score. To assess the degree of exchangeability between these scores, the Kullback-Leibler (KL) divergences between the estimated distributions are calculated. These divergences are then smoothed using a kernel to obtain the final weights.

4.3.1 Adaptive Weights Motivation

This section serves as the motivation for using test-sample adaptive weights. For ease of exposition, the Stochastic Volatility (SV) model will be assumed (Taylor, 1994). Here, the price process follows a random walk, and the second central moment is conditionally dependent. As we assume that the innovations are normally distributed around zero with dependent variance, we are not interested in modeling dependencies between other moments. The equations for the DGP are given below.

$$
\begin{aligned}
y_t &= \mu + y_{t-1} + \varepsilon_t, && \varepsilon_t \sim \mathcal{N}(0, \sigma_t^2) \\
h_t &= \phi h_{t-1} + \eta_t, && \eta_t \sim \mathcal{N}(0, \sigma_\eta^2) \\
\sigma_t^2 &= \sigma^2 \exp(h_t)
\end{aligned}
$$

The core idea behind conformal prediction is to use the empirical distribution of nonconformity scores and trim the tails of this distribution to obtain a forecasted nonconformity score interval at a new time step. This method relies on the assumption that the new nonconformity score belongs to the same distribution as the previous ones. However, in time series contexts, this assumption is often violated due to the dependency of observations on prior sequences.

For the given DGP, observing a relatively high innovation term ϵ_t when σ_t is relatively low is clearly less likely than observing a lower innovation term. In the nonexchangeable setting for conformal prediction, a weighted quantile of the observed nonconformity measures is taken. Let us take the residual as the nonconformity measure. Then, higher weights need to be assigned to residuals that are approximately exchangeable with the residual of the forecasted observation, while making sure that the effective sample size of the empirical distribution is still sufficient.

The foremost method described in the literature fixes weights in advance in relation to our prior expectations of the DGP. In the case of the presently described SV model, one could assign exponentially decaying weights, where the ratio of weight i to weight j, with $j > i$ is given as λ^{j-i}. This approach integrates the expected autoregressive covariance structure that is inherent in the assumed DGP. In essence, we are embedding our expectations of the dependency structure of the DGP into the utilized weights, while also considering the smoothness of their distribution.

However, this method can be improved on a fundamental level by making the weights dependent on the observed values of the target variable. This is because assigning fixed weights only allows one to model an expectation of future similarities between observations. By incorporating the previously observed data $y_{1:t-1}$, the state estimates $h_{1:t}$ can be refined. This enables the modeler to form a more accurate expectation of the similarities between observations.

In the SV model, this is reflected by virtue of the fact that the residual at time step t is exchangeable with the residual at time step $t-i$ to the degree that $h_t \approx h_i$ (considering the joint likelihood $p(e_t, e_i)$). Therefore, the data-uninformed weights model the dependency structure suboptimally and unnecessarily reduce the effective sample size when latent states repeat.

4.3.2 Long Short-Term Memory

An LSTM model is a specific type of Recurrent Neural Network (RNN) that includes an additional cell state to retain long-term information (Hochreiter & Schmidhuber, 1997). This makes it particularly effective for DGPs with long-range dependencies. The model is therefore chosen for its potential to generalize across different settings. However, this ability to generalize comes with the drawback of increased training complexity. Simpler models might perform better on straightforward DGPs without significant long-term dependencies. The LSTM model consists of three main components that integrate together in a recursive manner.

1. *Cells*: Each cell in the LSTM network corresponds to a time step in the input sequence. These cells are the fundamental units that process the input data x_t and maintain the internal state of the network. At each time step, an LSTM cell takes the input data and the previous states, the hidden state h_{t-1} and cell state c_{t-1}, to compute the current states h_t and c_t.

2. *States*: Each LSTM cell takes the input vector x_t along with the previous hidden state and cell state as inputs. It outputs an updated hidden state and cell state.

   - *Cell State* (C_t): Acts as the long-term memory of the LSTM. It is updated based on the forget gate and input gate, carrying forward important information through the sequence.
   - *Hidden State* (h_t): Represents the short-term memory and is used as the output of the LSTM cell. It is updated based on the output gate and the current cell state.

3. *Gates*: LSTM cells contain three types of gates that control the flow of information. Each gate takes the current input vector x_t and the previous hidden state h_{t-1} as input.

   - *Forget Gate* (f_t): This gate determines what information from the previous cell state should be discarded.
   - *Input Gate* (i_t): This gate decides which new information should be added to the cell state.
   - *Output Gate* (o_t): This gate controls which information from the cell state is passed to the hidden state.

A forward pass in the LSTM network involves taking the input vector x_t along with the previous hidden state h_{t-1} and cell state C_{t-1} as inputs to the cell. Inside the cell, the forget gate, input gate, and output gate perform internal computations to determine which information to discard, update, and output, respectively. The outputs of the cell are the updated cell state C_t, which carries forward important information, and the new hidden state h_t. The compact forms of the equations for the forward pass of an LSTM cell are given below (Gers et al., 1999).

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{1}$$
$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{2}$$
$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{3}$$
$$\tilde{c}_t = \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{4}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{5}$$
$$h_t = o_t \odot \sigma_h(c_t) \tag{6}$$

At each time step t, the forget gate (1) is computed first. Similarly, the input (2) and output gates (3) are calculated. All of the gates use the sigmoid activation function σ_g. A new cell state (4) is then proposed based on the input vector x_t and the previous hidden state h_{t-1} using a hyperbolic tangent activation function σ_c. This function normalizes the values of the cell state between -1 and 1, making sure that the network is stable over time (Mishra, 2023).

Let the superscripts d and h represent the number of input features and hidden units, respectively. Then $\mathbf{W} \in \mathbb{R}^{h \times d}$, $\mathbf{U} \in \mathbb{R}^{h \times h}$, and $\mathbf{b} \in \mathbb{R}^{h}$ are the weight matrices and bias vector parameters that need to be learned during training. In (5) the new cell state is created by combining the element-wise product $\odot$ of the forget gate with the previous cell state and the input gate with the proposed cell state. Finally, the output gate determines the new hidden state by applying the element-wise product to the cell state (6). Initially, both c_0 and h_0 are set to zero. A schematic representation of the LSTM cell can be found in Appendix H.

4.3.3 Model Features

Due to time constraints and practical considerations, the LSTM model fitted in this paper is only used to estimate the latent variance process. The model estimates the absolute residual as a proxy for the standard deviation following the approach of Auer et al. (2023). The base forecaster that is used is the Least Squares (LS) forecaster.

To accommodate various potential variance processes, we use three types of features derived from the residuals: (1) *lagged_abs_residual*, which represents the absolute residual with lags from 1 to 10, resulting in ten features; (2) *averaged_abs_residual*, which averages the absolute residual over lookback windows of 10, 20, 30, 40, and 50, yielding five features; and (3) *garch_sd_estimate*, a Generalized Autoregressive Conditional Heteroskedasticity (GARCH) estimate of the conditional standard deviation (Bollerslev, 1986).

The GARCH(1,1) model is refitted for each observation, such that all observations before the test point are used in the model fitting. Incorporating a GARCH input feature in the LSTM model can significantly enhance the variance modeling performance (Kim & Won, 2018).

Finally, the original covariates are added to the input features to model conditional heteroskedasticity coming from these covariates. The LSTM is implemented using the Keras API from TensorFlow in Python (Chollet, 2015), with model parameters detailed in Appendix I.

4.3.4 Weight Calculation

The weights are test-sample adaptive, meaning that the entire vector of weights is recalculated for each test sample. These weights are derived from the conditional moment predictions of the LSTM model. For each observation, we estimate a conditional nonconformity score distribution. This distribution is parameterized as a normal distribution with mean zero and a standard deviation equal to the absolute residual predicted by the LSTM.

Next, the KL divergences for each observation relative to the test point are calculated. This metric measures the statistical distance between distributions, making it a suitable proxy for the degree of exchangeability among the observations. The KL divergence ranges from $[0, \infty)$, where a value of 0 indicates identical distributions. To derive the weights, we therefore pass the KL divergences through an appropriate kernel function.

$$w_i = e^{-\rho D_{\mathrm{KL}}(p_i \| p_{n+1})}, \qquad i = 1, \ldots, n+1$$

Here, $D_{\mathrm{KL}}(P \parallel Q)$ denotes the KL divergence of the distribution P to the reference distribution Q. The specific nonconformity score distributions are represented as p_i. In this study, the parameter ρ, which acts as a smoothing parameter, is set to 100 without any tuning.

However, finding an optimal value for ρ is possible through cross-validation and grid search. Generally, ρ should be selected to ensure that each test point has a reasonable effective sample size. The appropriate choice of ρ depends on several factors, including the number of modeled moments, the complexity of the parameterized distribution, and the target coverage level $1 - \alpha$.

5 Results

This section presents the numerical results of the experiments conducted in this study. The aim of the study was to enhance the two stages of the nonexchangeable conformal prediction framework. Both stages are described in a separate section. The first stage focused on selecting an appropriate forecaster for nonexchangeable time series data; a Bayesian Structural Time Series (BSTS) model was chosen. The performance of this model is discussed and a detailed account of its coverage properties is given.

The second stage aimed to improve the nonexchangeable conformal prediction framework through a novel method of obtaining test-sample adaptive weights using a Long Short-Term Memory (LSTM) network. The proposed procedure is benchmarked against three other methods. Following this, we describe how the obtained weights respond to various scenarios. This description includes (1) the uniformity of the weights for i.i.d data, (2) the behavior of the weights after a changepoint, and (3) the parabolic shape of the weights exhibited for data with distribution drift.

5.1 The BSTS Forecaster

The BSTS model was selected to analyze the Elec2 Dataset. It was trained on an initial 2000 observations, while an additional 2000 observations formed the holdout set. The holdout set employed an expanding window approach, where predictions were incorporated into the holdout set after their true values were observed. The model delivered a stellar performance, achieving 89.3% coverage for the subsequent 23352 observations. The average interval width obtained by the model was 0.12. Figure 1 below depicts the coverage and interval width over time.

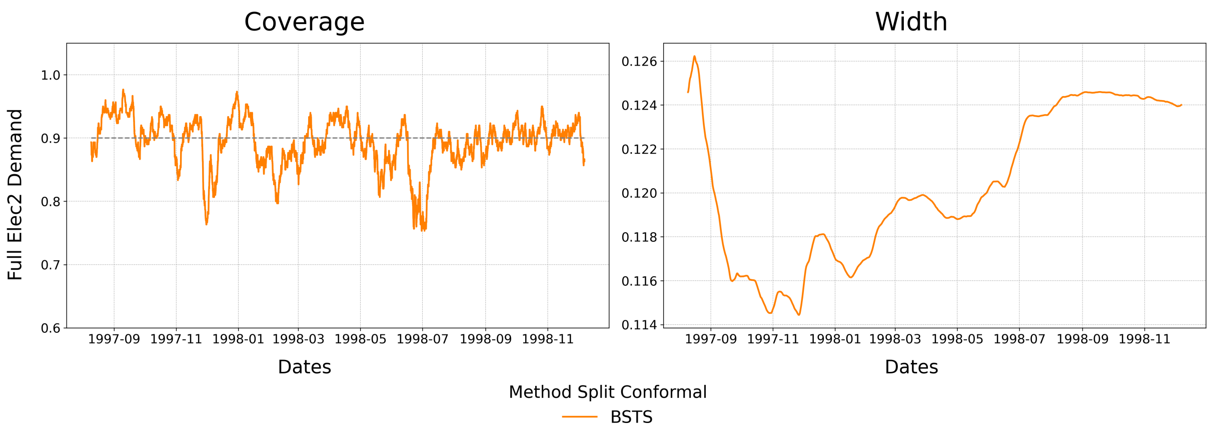

Figure 1: BSTS properties

The BSTS model significantly outperforms the methods used in Barber et al. (2023) for modeling the Elec2 dataset. The results from that paper are replicated in Appendix J. The BSTS model achieves an interval width that is more than four times narrower than the best method from the cited paper. Furthermore, the BSTS model is applied to predict the entire Elec2 dataset, while the other methods only used a subset of the data to reduce seasonality effects. This demonstrates the advantage of using models capable of inherently handling evolving temporal dependencies. A plot of the predicted intervals, the predictions, and the true values are given in Figure 2 below.

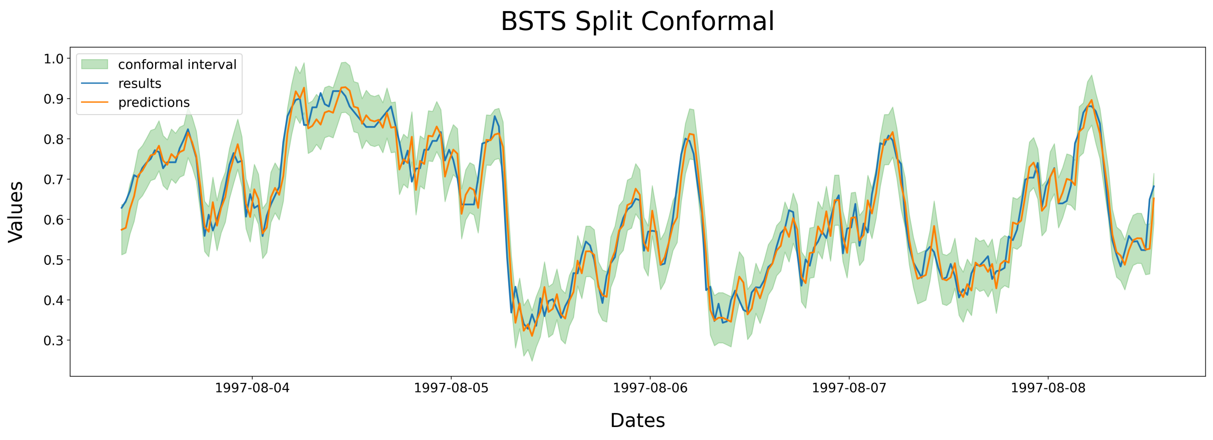

Figure 2: BSTS predictions

The plot clearly shows that the BSTS model accurately captures the seasonal effects inherent in the data. The predictions closely align with the true values, underscoring the model's effectiveness for time series data. The narrow predicted intervals indicate that the model has learned an adequate representation of the underlying process.

5.2 The LSTM Weights

An LSTM model was utilized to determine the test-sample adaptive weights by calculating the KL divergences between the estimated conditional nonconformity score distributions and passing them through a kernel. This method, paired with the Least Squares (LS) forecaster, will be referred to as **LstmCP + LS**. The effectiveness of this method is compared to three approaches used by Barber et al. (2023). These approaches are: **CP + LS** using an exchangeable algorithm with an LS model, **nexCP + LS** using a nonexchangeable algorithm with weights $w_i = 0.99^{n+1-i}$ and an LS model, and **nexCP + WLS** using a nonexchangeable algorithm with a Weighted Least Squares (WLS) model with weights and tags $w_i = t_i = 0.99^{n+1-i}$. The performance of these methods across the six simulated scenarios is graphically presented in Figure 3 and Figure 4 below. The numerical results can be found in Appendix K.

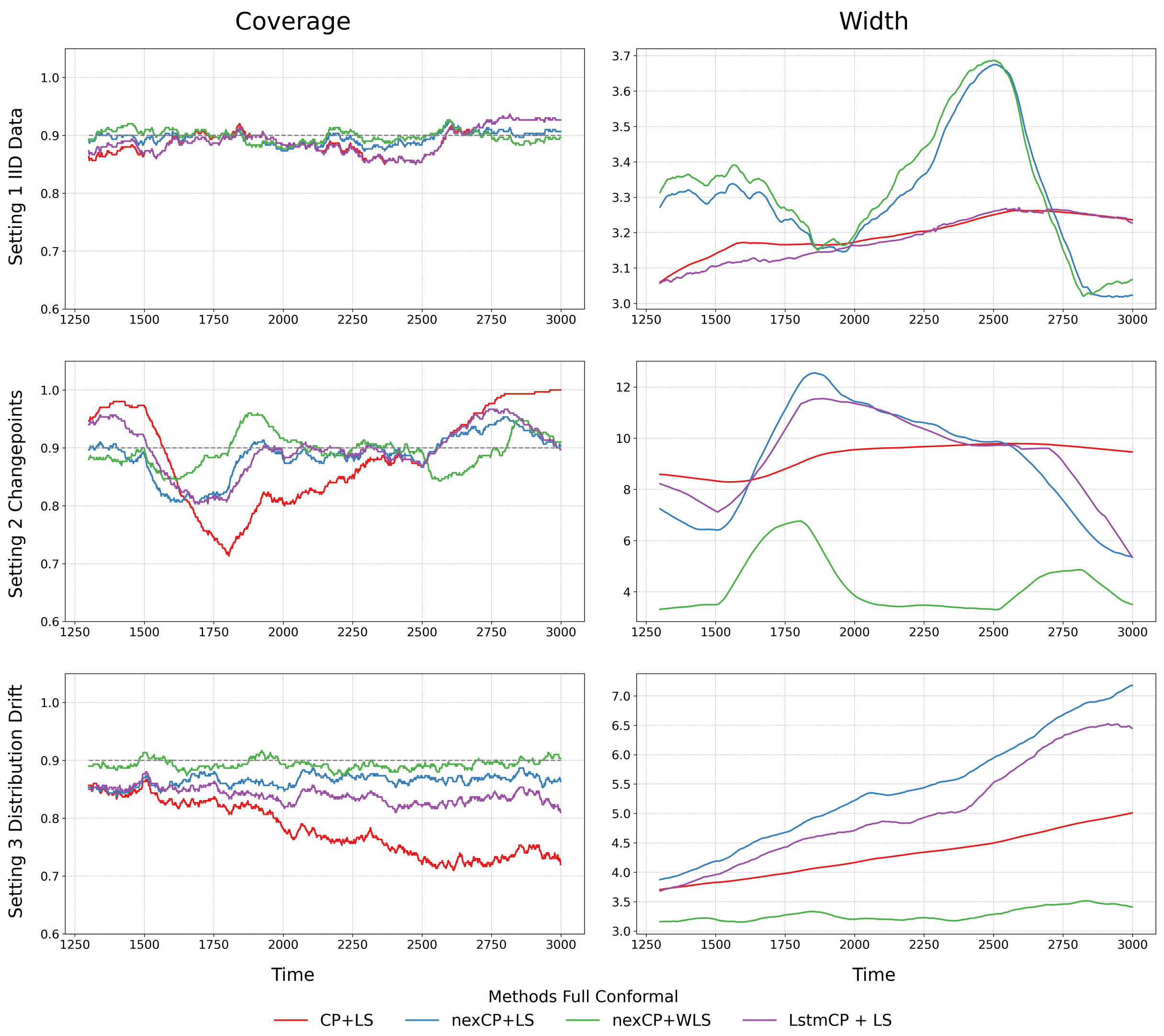

Figure 3: Coverage Properties Setting 1-3

For i.i.d. data, we observe that the LstmCP + LS method performs similarly to the LS + CP method. This similarity is desirable because it indicates that the proposed method for obtaining weights does not introduce additional variability into the system for i.i.d. data. In the context of changepoints and distribution drift, the method behaves similarly to the nonexchangeable nexCP + LS algorithm. This indicates that the LstmCP + LS method can autonomously determine the suitable treatment of the time series data. In essence, it adapts its approach by using an exchangeable algorithm for exchangeable data and a nonexchangeable algorithm for nonexchangeable data. This represents a significant advancement in developing a reliable method for conformal prediction in time series data.

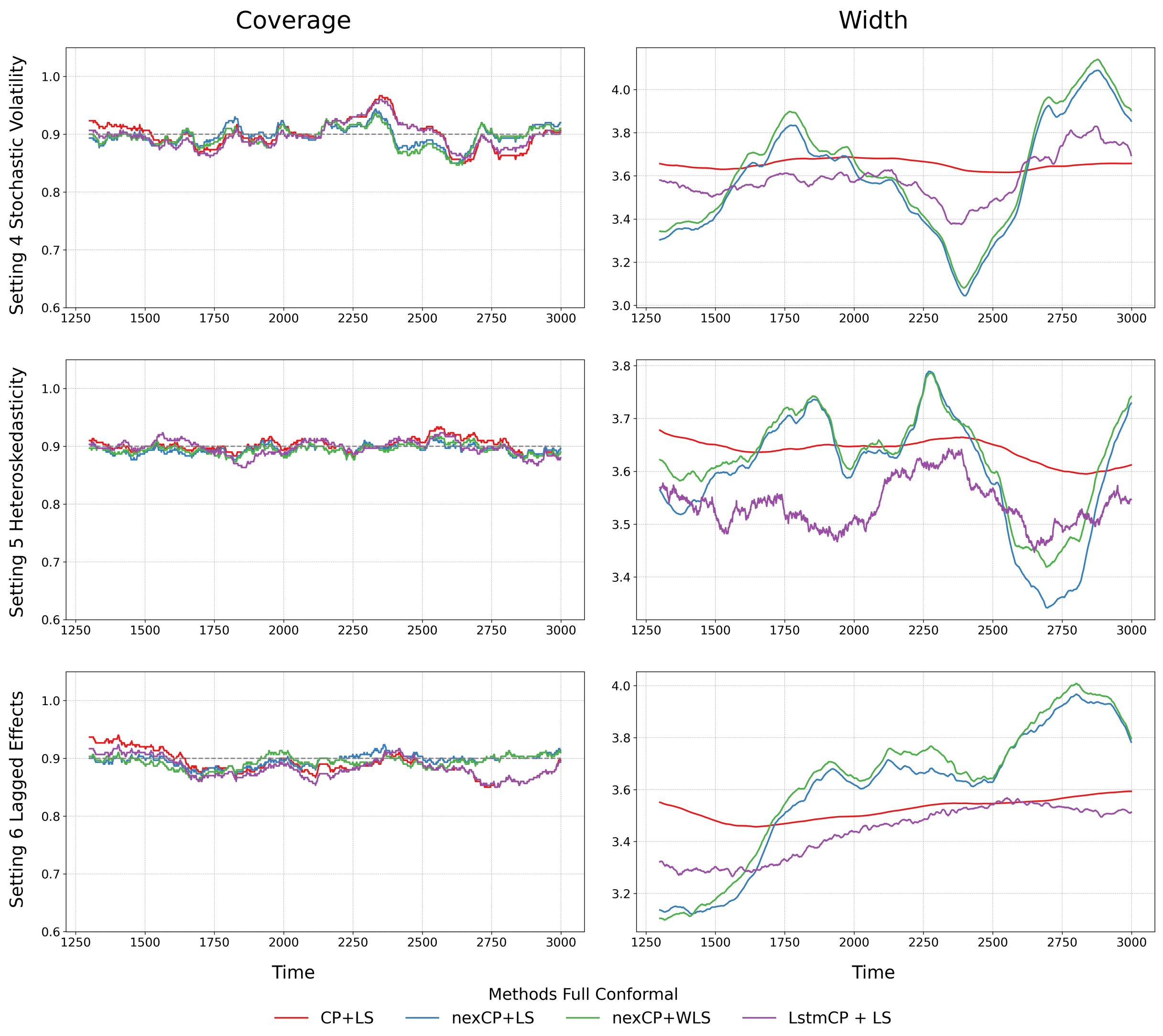

Figure 4: Coverage Properties Setting 4-6

For the final three settings, providing an appropriate interpretation of the interval widths is more challenging. This difficulty arises because we plot the widths using a rolling average of 300 observations. Therefore, for the stochastic volatility, heteroskedasticity, and lagged effects settings, we expect the influence of the conditional variance on the interval width to even out. Despite this, it is noteworthy that the interval width of the LstmCP + LS consistently dips below that of the CP + LS method. This suggests that the soft conditioning allows the LstmCP + LS method to achieve lower interval widths for observations where it has higher certainty.

This effect is particularly evident in the heteroskedasticity setting, likely because the model predicts narrower intervals when X_2 and X_4 are low, as these covariates are responsible for the heteroskedasticity. Moreover, since the interval widths for these settings should approximately even out when using a rolling average, the drawback of using fixed exponentially decaying weights becomes clear. These weights cause sharp peaks and troughs in the predicted interval widths, which is unjustified when averaging over 300 observations for these scenarios.

To gain a deeper understanding of the test-sample adaptive weights provided by the LSTM model, we will examine three specific settings. First, we will analyze the weights for i.i.d. data to confirm that the model assigns uniform weights when the data is exchangeable. The plot of the weights for i.i.d data is given in Figure 5 below.

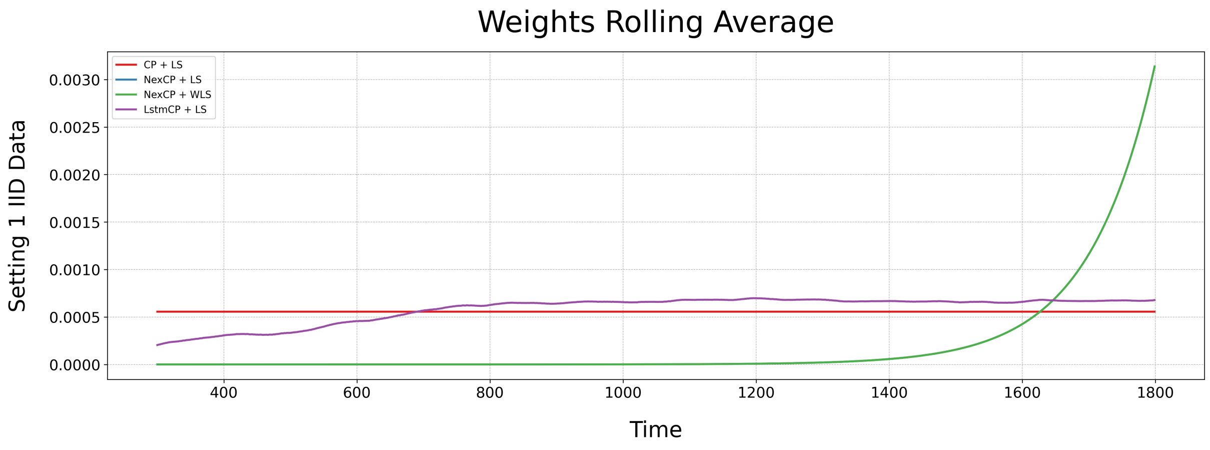

Figure 5: Weights for IID Data

In this plot, we can see that the weights are uniform, resembling the CP+LS method. The increasing weights in the first 500 observations are due to the LstmCP+LS method setting the first 200 weights to zero (due to the lagged features and practical considerations). This persists up to observation 500 because of the 300-observation rolling window. The same pattern applies to the other two plots. The next plot, Figure 6, shows how the test-sample adaptive weights react to a changepoint.

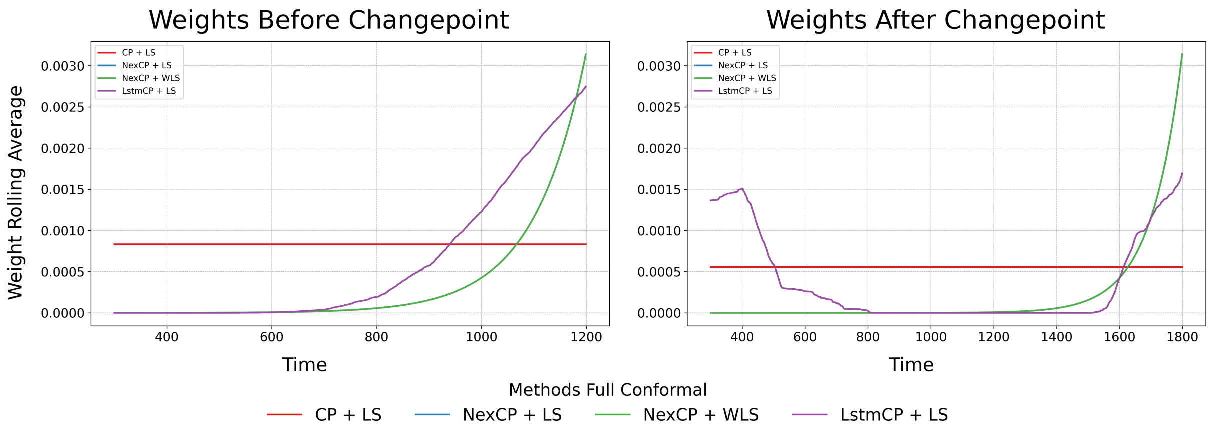

Figure 6: Response of Weights to a Changepoint

It is important to note that the first changepoint occurs at time step 500. In the first plot, the LstmCP + LS method already assigns more weight to observations after this changepoint. This occurs because the LS model is fitted on a larger number of observations following the changepoint compared to before it. Therefore the relations that it has learned are biased towards the second regime.

When a new changepoint occurs, the weights invert. At this point, the model is still primarily fitted on data from the second regime. The LstmCP + LS method identifies that its uncertainty in modeling the first and third regimes is similar. Thus, it assigns high weights to observations coming from both of these regimes. This approach increases the effective sample size compared to the NexCP methods while maintaining a comparable level of validity.

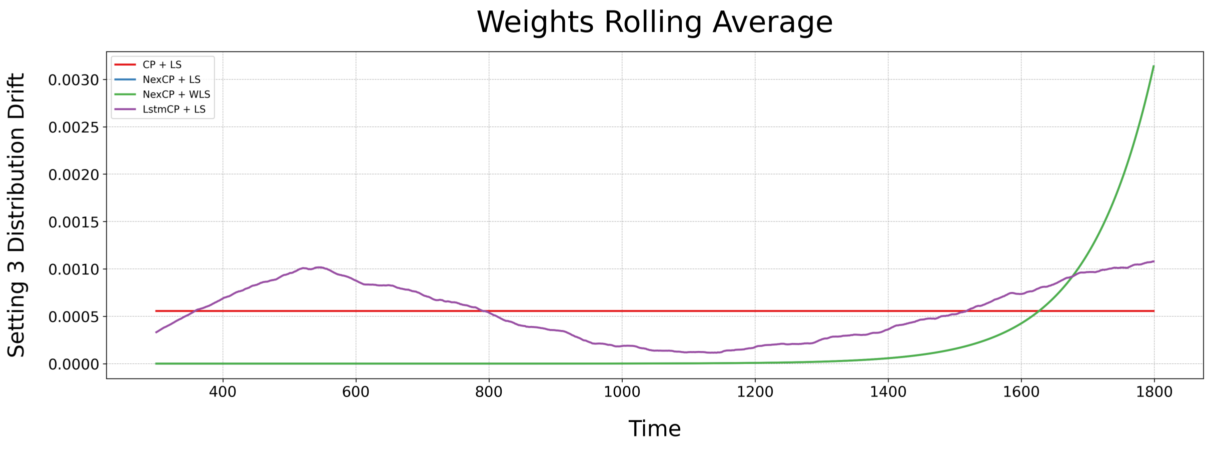

Figure 7: Weights for Distribution Drift

The final plot, Figure 7, displays the weights in the context of distribution drift. The pattern of the weights is as expected. Note again that the increasing weights up until observation 500 are a result of the aforementioned rolling window. After this point, the weights exhibit a parabolic shape with the minimum centered around observation 1100. This is logical since it roughly represents the midpoint of the observations. This means the LS model is most effective at modeling these observations and its performance gradually declines as we move outward from the center. Therefore, the Lstm + CP method also appropriately determines the weights for this scenario. The weights for the other three settings can be found in Appendix L.

6 Conclusion

This study aimed to enhance the conformal prediction framework for time series data through a two-stage approach. Initially, a Bayesian Structural Time Series (BSTS) model was utilized to manage evolving temporal dependencies inherent in time series data. Subsequently, a novel method for deriving test-sample adaptive weights was introduced using a Long Short-Term Memory (LSTM) network.

The BSTS model demonstrated significantly narrower prediction intervals, over four times tighter than those of conventional forecasting models. Additionally, despite using only a small subset of the data for training, the model came very close to achieving the target coverage level. This finding underscores the effectiveness of state-space models as forecasters for time series data within the conformal prediction framework.

The proposed LSTM method for determining test-sample adaptive weights proved effective across six different simulated scenarios. The method demonstrated excellent performance despite not having inherent expectations of the potential time series dependencies. For exchangeable data, it produced uniform weights. For nonexchangeable data, the method quantified the uncertainty in predicting the test point and assigned higher weights to nonconformity scores for which the used forecaster had a similar uncertainty.

This was further confirmed by a more detailed examination of the weight behavior. Following a changepoint, the method inverted the weights it had previously assigned. This inversion occurred because the test point now belonged to a new regime characterized by high uncertainty, contrasting with the previous regime of high certainty. For data showing distribution drift, the weights followed a parabolic pattern. This makes sense as the uncertainty of the used forecaster grows with the distance to the temporal midpoint.

In conclusion, our study demonstrated substantial improvements in both stages of the conformal prediction framework. By selecting an appropriate forecaster for time series data, we achieved a performance that is over four times more efficient than that of conventional forecasters. Most importantly, the LSTM method for deriving test-sample adaptive weights marked a significant advancement in the understanding and application of nonexchangeable conformal prediction algorithms.

Future research could investigate various LSTM network architectures. In particular, testing a deeper network could be beneficial, as this study employed only a single LSTM layer. Additionally, since the current LSTM model still fits a significant amount of noise, enhanced feature engineering and regularization techniques could further improve the method. Finally, other models for generating conditional moment forecasts could be investigated. A promising candidate is the N-BEATS model, which employs a deep neural architecture to achieve state-of-the-art performance in univariate time series modeling (Oreshkin et al., 2020).

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., . . . Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* `https://www.tensorflow.org/`. (Accessed: 2024-05-16. TensorFlow Structural Time Series (STS) library.)

Angelopoulos, A. N. & Bates, S. (2022). *A gentle introduction to conformal prediction and distribution-free uncertainty quantification.* Retrieved from `https://arxiv.org/abs/2107.07511`

Angelopoulos, A. N., Candes, E. J. & Tibshirani, R. J. (2023). *Conformal pid control for time series prediction.*

Auer, A., Gauch, M., Klotz, D. & Hochreiter, S. (2023). *Conformal prediction for time series with modern hopfield networks.*

Barber, R. F., Candes, E. J., Ramdas, A. & Tibshirani, R. J. (2020). *Predictive inference with the jackknife+.* Retrieved from `https://arxiv.org/abs/1905.02928`

Barber, R. F., Candès, E. J., Ramdas, A. & Tibshirani, R. J. (2023). Conformal prediction beyond exchangeability. *The Annals of Statistics*, *51*(2), 816 – 845. Retrieved from `https://doi.org/10.1214/23-AOS2276` doi: 10.1214/23-AOS2276

Barber, R. F., Candès, E. J., Ramdas, A. & Tibshirani, R. J. (2020). *The limits of distribution-free conditional predictive inference.*

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, *31*(3), 307-327. Retrieved from `https://www.sciencedirect.com/science/article/pii/0304407686900631` doi: https://doi.org/10.1016/0304-4076(86)90063-1

Chollet, F. (2015). *Keras.* `https://keras.io`.

Gers, F., Schmidhuber, J. & Cummins, F. (1999). Learning to forget: continual prediction with lstm. In *1999 ninth international conference on artificial neural networks icann 99. (conf. publ. no. 470)* (Vol. 2, p. 850-855 vol.2). doi: 10.1049/cp:19991218

Gibbs, I. & Candes, E. (2021). Adaptive conformal inference under distribution shift. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang & J. W. Vaughan (Eds.), *Advances in neural information processing systems* (Vol. 34, pp. 1660–1672). Curran Associates, Inc. Retrieved from `https://proceedings.neurips.cc/paper_files/paper/2021/file/0d441de75945e5acbc865406fc9a2559-Paper.pdf`

Guan, L. (2022). *Localized conformal prediction: A generalized inference framework for conformal prediction.*

Harries, M. (1999). *Splice-2 comparative evaluation: Electricity pricing* (Technical report). Sydney, Australia: University of New South Wales.

Harvey, A. C. (1990). *Forecasting, structural time series models and the kalman filter.* Cambridge University Press.

Hochreiter, S. & Schmidhuber, J. (1997, 12). Long short-term memory. *Neural computation*, *9*, 1735-80. doi: 10.1162/neco.1997.9.8.1735

Ingolfsson, T. M. (2021). *Insights into lstm architecture.* Retrieved from `https://thorirmar.com/post/insight_into_lstm/` (Accessed: 2024-06-25)

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, *82*(1), 35. Retrieved from `http://dx.doi.org/10.1115/1.3662552` doi: 10.1115/1.3662552

Kim, H. Y. & Won, C. H. (2018). Forecasting the volatility of stock price index: A hybrid model integrating lstm with multiple garch-type models. *Expert Systems with Applications*, *103*, 25-37. Retrieved from `https://www.sciencedirect.com/science/article/pii/S0957417418301416` doi: https://doi.org/10.1016/j.eswa.2018.03.002

Kingma, D. P. & Welling, M. (2022). *Auto-encoding variational bayes.*

Köhler, F. (2021). *Variational inference — evidence lower bound (elbo) — intuition & visualization.* Retrieved from `https://www.youtube.com/watch?v=HxQ94L8n0vU&list=PLISXH-iEM4JloWnKysIEPPysGVg4v3PaP` (Accessed: 2024-06-13)

Lei, J. & Wasserman, L. (2012). *Distribution free prediction bands.*

Mishra, P. (2023). Understanding and implementing lstm networks. *Medium*. Retrieved from `https://medium.com/@palashm0002/understanding-and-implementing-lstm-networks-41ca52495108` (Accessed: 2024-06-28)

Oreshkin, B. N., Carpov, D., Chapados, N. & Bengio, Y. (2020). *N-beats: Neural basis expansion analysis for interpretable time series forecasting.*

Scott, S. & Varian, H. (2014, 01). Predicting the present with bayesian structural time series. *Int. J. of Mathematical Modelling and Numerical Optimisation*, *5*, 4 - 23. doi: 10.1504/IJMMNO.2014.059942

Taylor, S. J. (1994). Modeling stochastic volatility: A review and comparative study. *Mathematical Finance*, *4*(2), 183-204. Retrieved from `https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-9965.1994.tb00057.x` doi: https://doi.org/10.1111/j.1467-9965.1994.tb00057.x

Vovk, V. (2012, 04–06 Nov). Conditional validity of inductive conformal predictors. In S. C. H. Hoi & W. Buntine (Eds.), *Proceedings of the asian conference on machine learning* (Vol. 25, pp. 475–490). Singapore Management University, Singapore: PMLR. Retrieved from `https://proceedings.mlr.press/v25/vovk12.html`

Vovk, V., Gammerman, A. & Shafer, G. (2005). *Algorithmic learning in a random world.* United States: Springer US. doi: 10.1007/b106715

Vovk, V., Gammerman, A. & Shafer, G. (2022). *Algorithmic learning in a random world* (2nd ed.). Springer Cham. Retrieved from `https://link.springer.com/book/10.1007/978-3-031-06649-8` doi: https://doi.org/10.1007/978-3-031-06649-8

Yi, W. (2019). *Demystifying tensorflow time series: Local linear trend.* Retrieved from `https://towardsdatascience.com/demystifying-tensorflow-time-series-local-linear-trend-9bec0802b24a` (Accessed: 2024-06-14)

Appendix

The Appendix covers various extra details of the paper for the interested reader.

A Elec2 Seasonality

Figure 8 below illustrates the hourly and daily seasonal effects present in the Elec2 dataset. The hourly mean transfer shows notable peaks between 20:00 and 06:00. The daily mean transfer indicates two peaks on Sunday and Monday. Additionally, the graph demonstrates that selecting an hourly subset from 09:00 to 12:00 removes most of the hourly seasonal effects.

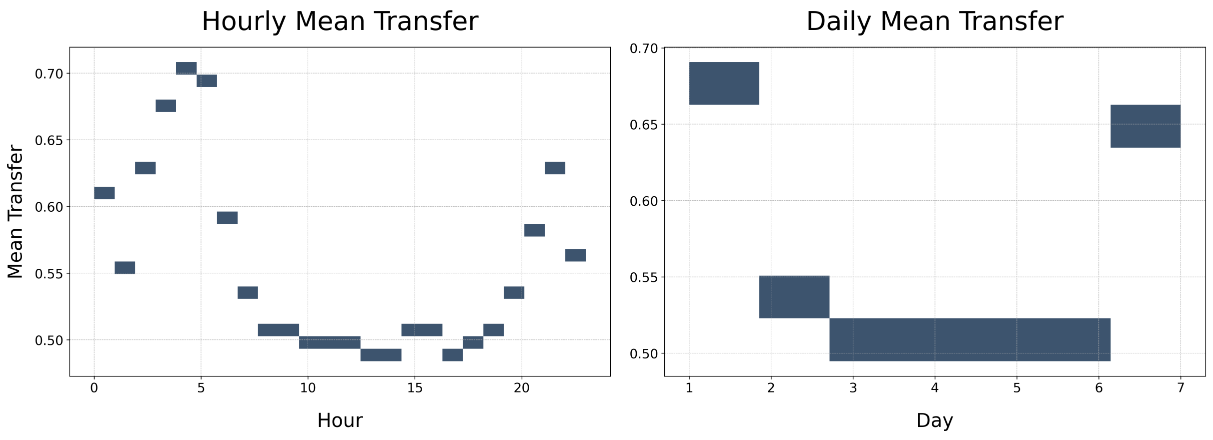

Figure 8: Transfer Rates Seasonal Effects

B Simulation Graphs

The graphs below illustrate the different types of nonexchangeability exhibited by the newly introduced scenarios. Figure 9 shows a process with clear volatility clustering. Periods of high and low volatility occur randomly but persist over time. Next, Figure 10 shows variance mainly determined by the two covariates X_2 and X_4. As these covariates are generated randomly, the graph does not exhibit a clear pattern for the variance. However, we know that the variance fluctuates depending on the values of the relevant X variables. Finally, Figure 11 displays a repeating pattern with a wavelength of 50, which is expected due to the introduced lag-50 in the AR component. A well-generalizing model should be able to detect all three of these latent processes.

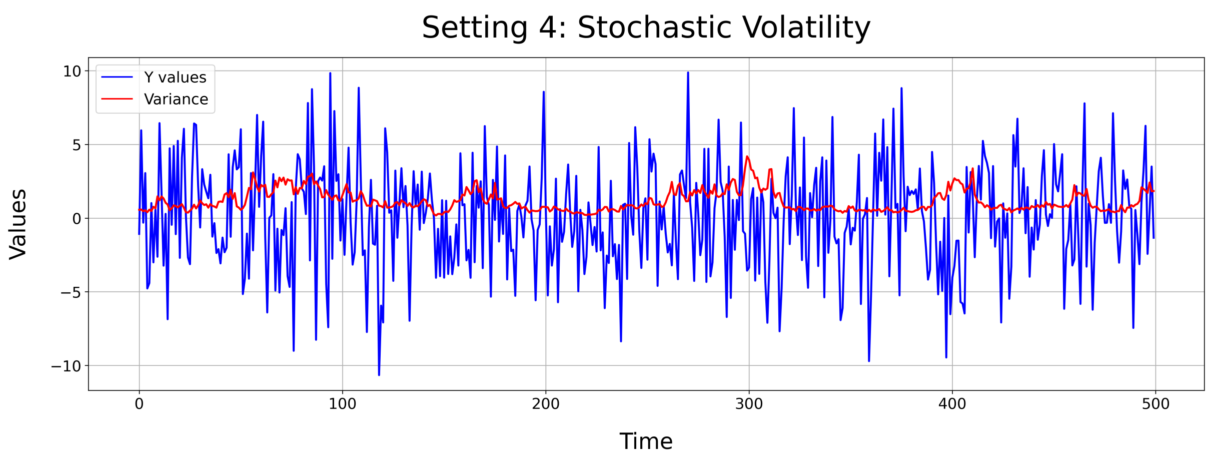

Figure 9: Simulation Setting 4

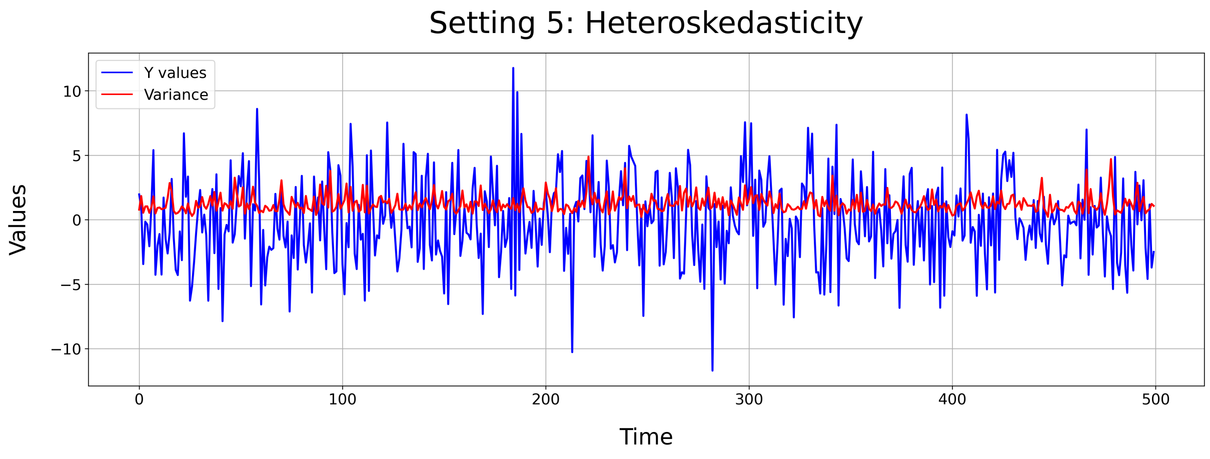

Figure 10: Simulation Setting 5

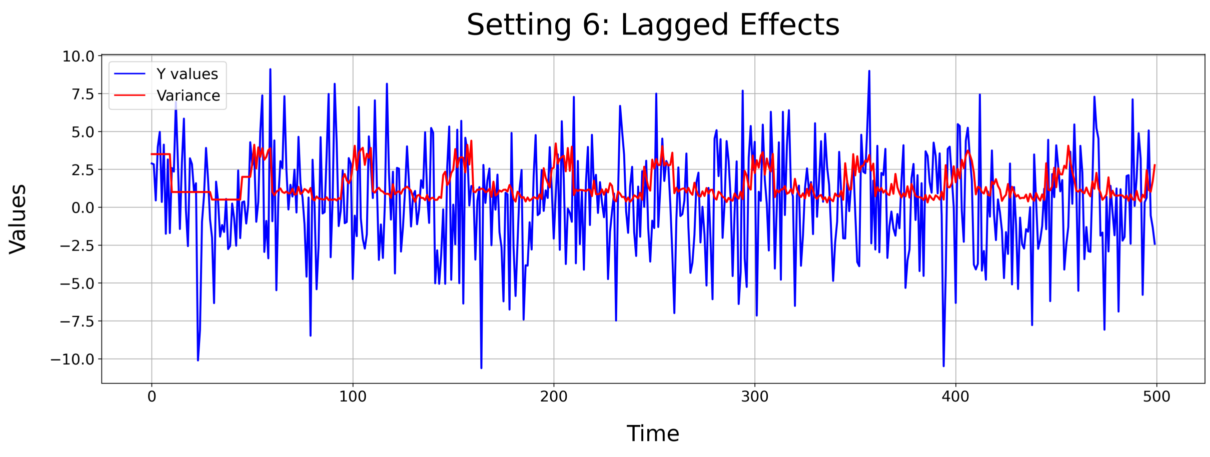

Figure 11: Simulation Setting 6

C 2005 Algorithm

Vovk et al. (2005) originally presents the algorithm implemented by Barber et al. (2023) as the Ridge Regression Confidence Machine (RRCM). The RCCM uses the absolute error as the nonconformity measure. This algorithm can be further adapted to Weighted Ridge Regression (WRR), while simultaneously being extended to handle nonexchangeable data. The WRR predictions can be obtained with the following formula.

$$\hat{Y}_n = X_n \left(X_n' W_n X_n + z I_p\right)^{-1} X_n' W_n Y_n$$

Here, when $z = 0$, this clearly reduces to Weighted Least Squares (WLS) regression. To simplify the notation of the vector of the nonconformity scores $(s_1, s_2, \ldots, s_n)'$, we can make use of the hat matrix $H = X_n \left(X_n' W_n X_n + z I_p\right)^{-1} X_n'$. Note that the vector of nonconformity scores can now be given as.

$$|Y_n - H_n Y_n| = |\left(I_n - H_n\right) Y_n|$$

Let y be a possible label for x_n. Then, we can represent the label vector as $Y = (y_1, y_2, \ldots, 0)'$ $+ (0, 0, \ldots, y)'$. This allows us to write the vector of nonconformity scores as a true label component plus a hypothesized label component $|A + By|$. This deconstruction makes use of.

$$A = \left(I_n - H_n\right)(y_1, y_2, \ldots, 0)'$$
$$B = \left(I_n - H_n\right)(0, 0, \ldots, 1)'$$

In conformal prediction, we are looking for hypothesized labels that conform to the data. To do this, we select the y labels that have a p-value greater than α, a pre-specified miscoverage parameter. Under exchangeability of the data, this corresponds to selecting the labels y for which it holds that a fraction equal or greater than α has a greater or equal nonconformity score.

To extend this method to the nonexchangeable setting, the p-value needs to be redefined. In particular, we have a vector of normalized weights $(\tilde{w}_1, \tilde{w}_2, \ldots, \tilde{w}_n)$ that represents our belief in the degree of exchangeability of the observations with our test point, while simultaneously ensuring that the effective sample size is great enough. Intuitively, a measure of nonconformity is only valid when the observations can be compared. Hence, we redefine the p-value as follows.

$$p^y = \sum_{i=1}^{n} \left(\tilde{w}_i I_i^y\right)$$

Here, we make use of the indicator function I_i^y that indicates whether the label y is in the set S_i. The label is in the set S_i when the nonconformity measure given the hypothesized label of observation i is greater than or equal to that of the test point n.

$$I_i^y = \begin{cases} 1 & \text{if } y \in S_i \\ 0 & \text{if } y \notin S_i \end{cases}$$

$$S_i = \{y : s_i(y) \geq s_i(y)\} = \{y : |a_i + b_i y| \geq |a_n + b_n y|\}$$

In light of the previous explanation, it effectively measures whether observation i is more or equally nonconformal as the test point n. In the weighted case, we need to adjust the relevance of this measure according to the normalized weight, which serves the purpose discussed before. Note that the p-value can only change if the sign of $s_i(y) - s_n(y)$ changes. Since we are interested in $|a_i + b_i y|$, we can assume $b_i \geq 0$, for $i = 1, \ldots, n$. As y increases, the possible points at which the p-value might change are given by.

$$\begin{aligned} -\frac{a_i - a_n}{b_i - b_n} \quad \text{and} \quad & -\frac{a_i + a_n}{b_i + b_n}, && \text{if } b_i = b_n \neq 0 \\ & -\frac{a_i + a_n}{2b_i}, && \text{if } b_i = b_n \neq 0 \text{ and } a_i \neq a_n \end{aligned}$$

Hence we can find the explicit representation of the set for which $p^y \geq \alpha$ as the union of finitely many intervals and rays. The algorithm is schematically laid out below in Algorithm 2.

Algorithm 2 Weighted RCCM Nonexchangeable Setting 2005

```
**Input:**
    $A = (I_n - H_n)(y_1, y_2, \ldots, 0)'$
    $B = (I_n - H_n)(0, 0, \ldots, 1)'$
for $i = 1, \ldots, n$ do
    if $b_i < 0$ then
        $a_i = a_i; b_i = -b_i$
    end if
end for
$\mathcal{P} = \emptyset$
for $i = 1, \ldots, n$ do
    if $b_i \neq b_n$ then
        add $\frac{a_i - a_n}{b_i - b_n}$ and $\frac{a_i + a_n}{b_i + b_n}$ to $\mathcal{P}$
    end if
    if $b_i = b_n \neq 0$ and $a_i \neq a_n$ then
        add $\frac{a_i + a_n}{2b_i}$
    end if
end for
add $-\infty$ and $\infty$ to $\mathcal{P}$
sort $\mathcal{P}$ in ascending order obtaining $y_{(0)}, \ldots, y_{(m+1)}$
$N(j) = 0; j = 1, \ldots, m$
for $i = 1, \ldots, n$ do
    for $j = 0, \ldots, m$ do
        if $|a_i + b_i y| \geq |a_n + b_n y|$ for $y \in \{y_{(j)}, y_{(j+1)}\}$ then
            $N(j) = N(j) + \tilde{w}_i$
        end if
    end for
end for
$M(j) = 0; j = 1, \ldots, m$
for $i = 1, \ldots, n$ do
    for $j = 0, \ldots, m$ do
        if $|a_i + b_i y| \geq |a_n + b_n y|$ then
            $M(j) = M(j) + \tilde{w}_i$
        end if
    end for
end for
Return:
    $C_n = \{\cup_{j:N(j)\geq\alpha}(y_{(j)}, y_{(j+1)})\} \cup \{y_{(j)} : M(j) \geq \alpha\}$
```

D 2022 algorithm

Vovk et al. (2022) later developed a simpler and more efficient approach for doing conformal prediction for Ridge Regression. The main logic is the same as discussed in Appendix C. However, instead of defining a single set S_i that uses the absolute error as the nonconformity measure, a combination of sets $S_i^{(l)}$ and $S_i^{(u)}$ is used. Here (l) denotes lower and (u) denotes upper. The lower set takes the form $(-\infty, t_i]$ and the upper set takes the form $[t_i, \infty)$.

The sets follow by taking both a lower and upper nonconformity measure. These are defined as $s_i^{(l)} = \hat{y}_i - y_i$ and $s_i^{(u)} = y_i - \hat{y}_i$. The lower nonconformity measure assesses how much smaller the true label is than the predicted label. Conversely, the upper nonconformity measure evaluates how much larger the true label is than the predicted label. Since the sets $S_i^{(l)}, S_i^{(u)}$ do not make use of an absolute value measure, the points for which the sign $s_i^{(j)} - s_n^{(j)}$, with $j \in \{l, u\}$ might change is simply given as $\dfrac{a_i - a_n}{b_n - b_i}$.

Now, we define both a lower $l = (l_1, l_2, \ldots, l_n)'$ and upper $u = (u_1, u_2, \ldots, u_n)$ vector of points. Since t_i is both a lower bound of $S_i^{(l)}$ as an upper both of $S_i^{(u)}$, we simply set $t_i = l_i = u_i$ with one exception; In the anomalous case that $b_n \leq b_i$ we set $l_i = -\infty$ and $u_i = \infty$. This is because the slope of the inequality $b_i - b_n$ would be negative, thereby pointing to y values in the wrong direction. Intuitively, we want to avoid cases where the nonconformity measure of point i is more sensitive to the hypothesized label y than the point n itself.

Now, as each point l_i corresponds to the lower bound of a set S_i^l, and each point u_i corresponds to the upper bound of a set $S_i^{(u)}$, the conformal prediction set in the exchangeable scenario can easily be found. First, sort l and u in ascending order. Then, take the $(\alpha/2)$ quantile of the lower vector and the $1 - (\alpha/2)$ quantile of the upper vector as the lower and upper bound of your predicted set, respectively.

In the setting of nonexchangeable data, we similarly have to take into account the adjusted p-value defined in Appendix C. In the adapted algorithm, this is implemented by defining two new vectors of ordered normalized weights $\tilde{w}^{(l)}$ and $\tilde{w}^{(u)}$ that follow the same sorting arguments as the lower and upper vector. Then, the index of the lower bound in the lower vector, and the upper bound in the upper vector, is found by finding the smallest index in the normalized weights $\tilde{w}^{(l)}$ and $\tilde{w}^{(u)}$ such that the cumulative sum of the weights up until the lower and upper index is greater than $(\alpha/2)$ and $1 - (\alpha/2)$, respectively. The upper index needs to be adjusted by -1 so that it covers more than a fraction $(\alpha/2)$ of sets. The algorithm is schematically laid out below in Algorithm 3.

Algorithm 3 Weighted RCCM Nonexchangeable Setting 2022

```
Input:
    $A = (I_n - H_n)\,(y_1, y_2, \ldots, 0)'$
    $B = (I_n - H_n)\,(0, 0, \ldots, 1)'$
for $i = 1, \ldots, n-1$ do
    if $b_n - b_i > 0$ then
        set $l_i = u_i = \dfrac{a_i - a_n}{b_n - b_i}$
    else
        set $l_i = -\infty$ and $u_i = \infty$
    end if
end for
sort $l = (l_1, \ldots, u_{l-1})$ in ascending order, then get $\tilde{w}^{(l)}$
find lower index $g = \min\{g \mid \sum_{i=1}^{g} \tilde{w}_i^{(l)} \geq \alpha\}$
sort $u = (u_1, \ldots, u_{n-1})$ in ascending order, then get $\tilde{w}^{(u)}$
find upper index $h = \min\{h \mid \sum_{i=1}^{h} \tilde{w}_i^{(u)} \geq 1 - \alpha\} - 1$
Return:
    $C_n = [l_g, u_h]$
```

E The BSTS Priors

The BSTS model in this paper is a sum of three Bayesian time series models, plus a regression model. The three time series models comprise one model that captures changes in the level and the slope of the overall time series, while the other two models serve to capture hour-of-day and day-of-week seasonality.

In the Bayesian framework, we are required to define prior probability distributions for all coefficients. This section serves to explain the definitions and initialization of the two time series architectures that are employed. These are referred to as the *Semi-Local Linear Trend* model and the *Seasonal Effects* model. The latter is deployed both for the Day-of-Week and the Hour-of-Day seasonality.

1. *Semi-Local Linear Trend*: The Semi-Local Linear Trend is a simple time series model with a level that evolves according to a random walk and a slope component that moves according to an AR(1) process centered on a potentially nonzero value D.

$$\begin{aligned} \mu_{t+1} &= \mu_t + \delta_t + \epsilon_t & \epsilon_t &\sim \mathcal{N}(0, \sigma_\mu) \\ \delta_{t+1} &= D + \phi(\delta_t - D) + \eta_t & \eta_t &\sim \mathcal{N}(0, \sigma_\delta) \end{aligned}$$

   To have valid priors for the components μ_t and δ_t as they evolve over time, their initial priors need to be defined first. These serve to initialize the first variables μ_0 and δ_0

- *Initial Level Prior*: A prior distribution for the initial level. μ_0
- *Initial Slope Prior*: A prior distribution for the initial slope δ_0.

To ensure valid prior distributions for the time-dependent parameters, we must specify four additional prior distributions that condition on our estimates of the parameters' realized values from the previous time step.

- *Level Scale Prior*: A prior distribution for the level scale parameter σ_μ.
- *Slope Mean Prior*: A prior distribution for the slope mean D.
- *Slope Scale Prior*: A prior distribution for the slope scale σ_δ.
- *Autoregressive Coefficient Prior*: A prior distribution for the autoregressive coefficient ϕ.

2. *Seasonal Effects*: The Seasonal Effects model represents the current observation as the seasonal effect corresponding to that observation. The model maintains a set of seasonal effects equal in number to the defined seasons S. Each of these seasonal effects evolves according to a random walk.

$$\gamma_{t+1} = -\sum_{s=0}^{S-2} \gamma_{t-s} + \epsilon_t \qquad\qquad \epsilon_t \sim \mathcal{N}(0, \sigma_\gamma)$$

To have valid priors for the parameter γ_t as it evolves over time, two priors need to be defined.

- *Initial Effect Prior*: A prior distribution for the initial effect of each season. It is also possible to have an independent prior distribution for each season.
- *Drift Scale Prior*: A prior distribution for the drift scale σ_γ.

The model is constructed in Python using TensorFlow STS (Abadi et al., 2015), which handles the entire process of fitting the surrogate posterior. This means that the priors are heuristically formed based on the observed time series (which is not truly Bayesian but is often done in practice to avoid model misspecification). Additionally, the approximated posterior is mean-field, meaning it assumes the posterior factorizes into independent distributions for each parameter. Thus, the surrogate posterior is built by training an independent distribution for each parameter. Specifically, the surrogate posterior consists of independent normal distributions with trainable mean and scale parameters, which are then transformed into the appropriate distribution approximation for that parameter using bijective functions.

F State-Space Representation BSTS Model

In essence, the BSTS is a linear state-space model plus a regression model. To understand the state space model, the regression model that is added on top of this model is disregarded in this section. To get a grip on the model, we will write it in terms of the fundamental equations for any state space model.

$$
\begin{aligned}
z_t &= Az_{t-1} + w_t, &\qquad w_t &\sim \mathcal{N}(b, \Sigma_w) \\
y_t &= Hz_{t-1} + v_t, &\qquad t_t &\sim \mathcal{N}(0, \Sigma_v)
\end{aligned}
$$

Here the first equation is called the transition equation since it models how the latent states evolve over time. The latent states are denoted by z, they evolve linearly according to the transition matrix A and a Gaussian noise component w. The second equation is the observation equation, as it links the observed values y to the latent states x by means of the linear relations defined in the observation matrix H with a Gaussian noise component v. For additional clarity, we can define the matrices using the notation for the coefficients defined in Section 4.2.1.

$$
\begin{bmatrix} \mu_t \\ \delta_t \\ \gamma_t^{(d)} \\ \gamma_t^{(h)} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & \phi & 0 & 0 \\ 0 & 0 & A_t^{(d)} & 0 \\ 0 & 0 & 0 & A_t^{(h)} \end{bmatrix} \begin{bmatrix} \mu_{t-1} \\ \delta_{t-1} \\ \gamma_{t-1}^{(d)} \\ \gamma_{t-1}^{(h)} \end{bmatrix} + \begin{bmatrix} w_{1,t} \\ w_{2,t} \\ w_{3,t} \\ w_{4,t} \end{bmatrix}
$$

$$
z_t = A_t z_{t-1} + w_t, \quad w_t \sim \mathcal{N}(b, \Sigma_w)
$$

The latent states for the seasonal effects are organized such that the current seasonal effect of the relevant seasonal effects model is always in the first dimension. The seasonal effects transition matrices $A_t^{(d)}$ and $A_t^{(h)}$ are ruled by the following logic.

$$
A_t^{(g)} = \begin{cases} P_g & \text{if the season changes at time step } t \\ I_g & \text{otherwise} \end{cases}
$$

Here, the index $g \in \{d, h\}$ selects the relevant seasonal effects model. Moreover $|g|$ denotes the cardinality of their respective sets $\mathcal{D}$ (set of 7 days) and $\mathcal{H}$ (set of 24 hours). The purpose of the permutation matrix P_g is to rotate the ordering of the seasonal effects vector. The matrix I_g is the identity matrix.

$$
P_g = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & 0 & 0 & \cdots & 0 \end{bmatrix} \qquad I_g = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}
$$

The dimensions of P_g and I_g are $|g| \times |g|$. The observation equation can then be denoted as follows.

$$\begin{bmatrix} y_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & H^{(d)} & H^{(h)} \end{bmatrix} \begin{bmatrix} \mu_t \\ \delta_t \\ \gamma_t^{(d)} \\ \gamma_t^{(h)} \end{bmatrix} + \begin{bmatrix} v_t \end{bmatrix}$$

$$y_t = H z_{t-1} + v_t, \quad v_t \sim \mathcal{N}(0, \sigma_v)$$

Here, $H^{(g)}$ is a $1 \times |g|$ vector with a one as the first element followed by all zeros. The vector selects the appropriate seasonal effect. Finally, the Gaussian noise terms are defined as.

$$w_t = \begin{bmatrix} w_{1,t} \\ w_{2,t} \\ w_{3,t} \\ w_{4,t} \end{bmatrix} \sim \mathcal{N}\left(b = \begin{bmatrix} 0 \\ (1-\phi)D \\ 0 \\ 0 \end{bmatrix}, \Sigma_w = \begin{bmatrix} \sigma_\mu, 0, 0, 0 \\ 0, \sigma_\delta, 0, 0 \\ 0, 0, \sigma_{d,t}, 0 \\ 0, 0, 0, \sigma_{h,t} \end{bmatrix} \right), \qquad v_t \sim \mathcal{N}(0, \sigma_v)$$

Here, the drift scales $\sigma_{d,t}$ and $\sigma_{h,t}$ follow a similar logic as their transition matrices.

$$\sigma_{g,t} = \begin{cases} \sigma_g & \text{if the season changes at time step } t \\ 0 & \text{otherwise} \end{cases}$$

G Variational Inference

The problem in finding the true posterior is that we can not evaluate the marginal $p(y)$.

$$\begin{aligned} p(y) &= \int p(y, \theta)\, d\theta \\ &= \int p(Y = y \mid \theta) p(\theta)\, d\theta \end{aligned}$$

Integrating out the model parameters for the specified BSTS is an intractable task (Kingma & Welling, 2022). The Variational Inference (VI) procedure solves this problem by finding a surrogate posterior that best approximates the true posterior. The explanation in this section closely follows the demonstration of Köhler (2021), but it is adapted to consider the model parameters instead of the latent states.

The surrogate posterior is constructed from a set of distributions that are easy to work with and is defined in such a way that it best captures the features of the true posterior. The intuition behind this is that even though the true posterior is complex, it might have characteristic attributes that can be well represented by other, simpler distributions.

This means that we are looking for a function $q(\theta)$ that best approximates the true distribution $p(\theta \mid Y = y)$. In essence, we want to minimize the distance between the approximating distribution and the target distribution. Hence, the metric that we need to minimize is the

Kullback-Leibler (KL) divergence. The problem can then be set up as.

$$q^*(\theta) = \arg\min_{q(\theta)\in\mathcal{Q}} \mathrm{KL}(q(\theta) \parallel p(\theta \mid Y = y))$$

For a given set of families of distributions $\mathcal{Q}$. However, this term still involves the true posterior. Hence, the KL divergence needs to be rewritten so that we can find the optimal distribution.

$$\begin{aligned}
\mathrm{KL}(q(\theta) \parallel p(\theta \mid Y = y)) &= \int q(\theta)\log\left(\frac{q(\theta)}{p(\theta \mid Y = y)}\right) d\theta \\
&= \int q(\theta)\log\left(\frac{q(\theta)\cdot p(y)}{p(\theta, y)}\right) d\theta \\
&= \int q(\theta)\log\left(\frac{q(\theta)}{p(\theta, y)}\right) d\theta + \int q(\theta)\log(p(y))\, d\theta \\
&= \mathbb{E}_{\theta\sim q(\theta)}\left[\log\left(\frac{q(\theta)}{p(\theta, y)}\right)\right] + \mathbb{E}_{\theta\sim q(\theta)}\left[\log p(y)\right] \\
&= -\mathbb{E}_{\theta\sim q(\theta)}\left[\log\left(\frac{p(\theta, y)}{q(\theta)}\right)\right] + \log p(y) \\
&= -\mathcal{L}(\theta) + \log p(y)
\end{aligned}$$

The log-likelihood $\log p(y)$ of the data is fixed and is called the evidence. Since the probability is between zero and one, the logarithm of the probability will be negative. Here, $\mathcal{L}(q)$ is called the Evidence Lower Bound (ELBO), and it can not be greater than the evidence. Having rewritten the KL divergence, we now have a solvable optimization problem.

$$\begin{aligned}
q^*(\theta) &= \arg\min_{q(\theta)\in\mathcal{Q}} \mathrm{KL}(q(\theta) \parallel p(\theta \mid Y = y)) \\
&= \arg\max_{q(\theta)\in\mathcal{Q}} \mathcal{L}(q)
\end{aligned}$$

This is the optimization problem that TensorFlow solves when approximating the posterior distribution of the BSTS.

H Schematic Representation LSTM

The schematic representation of an LSTM cell that is given below is taken from Ingolfsson (2021).

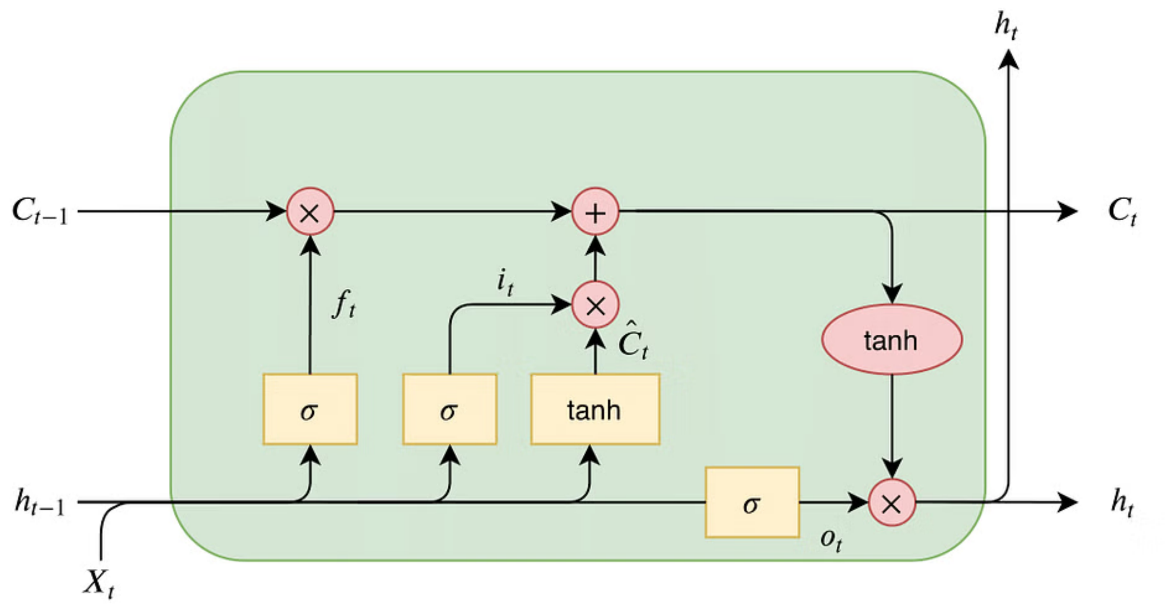

Figure 12: Schematic Representation of an LSTM cell

I LSTM Parameters

The LSTM network in this study consists of a single LSTM layer with 200 units. To prevent overfitting, a dropout layer with a rate of 0.5 is applied after the LSTM layer. This dropout layer randomly sets half of the hidden state outputs to zero during training. Finally, a dense layer is used to compute the prediction. The parameters used to train the model using the Keras API are given below. Additionally, early stopping with a patience of 10 was implemented to train the model.

- *n_neurons: 200*
- *stateful*: True
- *kernel_regularizer*: l1_l2(l1 = 0.0001, l2 = 0.01)
- *kernel_initializer*: glorot uniform
- *n_epochs*: 500
- *n_batch*: 100

J Replication Results

The 2005 algorithm ran in 1:26:25. The numerical results are presented in Table 1 below. Following that, a graphical display of the coverage properties is given in Figure 13.

	Regular Elec2 Data		Permuted Elec2 Data	
	Coverage	Width	Coverage	Width
CP+LS	0.852	0.565	0.899	0.639
NexCP+LS	0.890	0.606	0.908	0.652
NexCP+WLS	0.884	0.549	0.908	0.663

Table 1: 2005 Algorithm Results

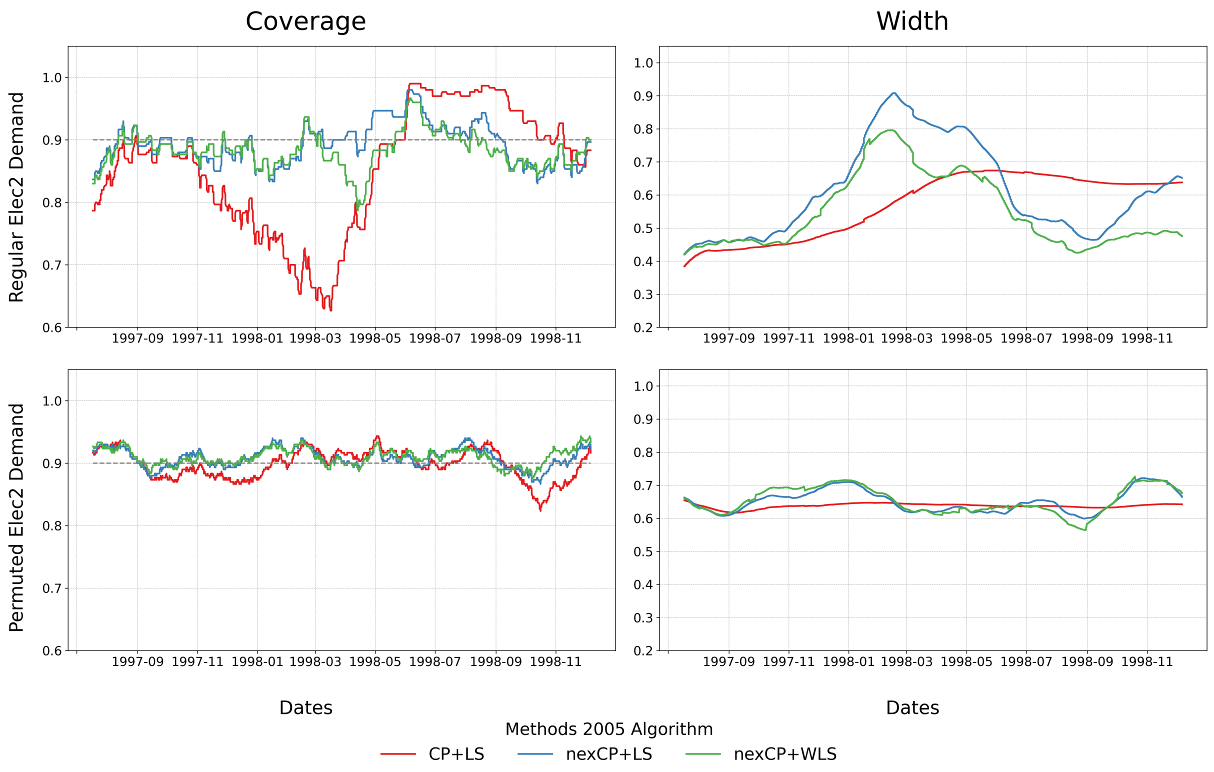

Figure 13: 2005 Algorithm Results

The 2022 algorithm ran in 15:38. The numerical results are presented in Table 2 below. Following that, a graphical display of the coverage properties is given in Figure 14.

	Regular Elec2 Data		Permuted Elec2 Data	
	Coverage	Width	Coverage	Width
CP+LS	0.874	0.566	0.899	0.627
NexCP+LS	0.891	0.496	0.898	0.633
NexCP+WLS	0.885	0.509	0.899	0.612

Table 2: 2022 Algorithm Results

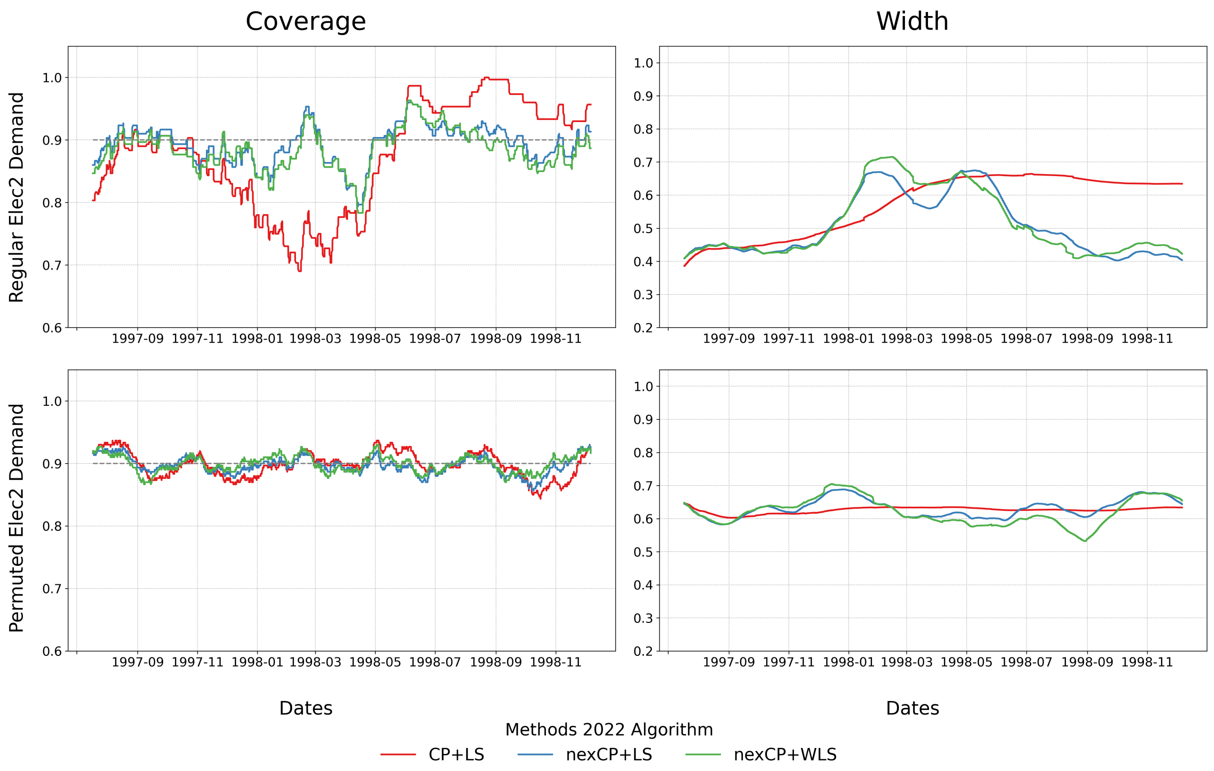

Figure 14: 2022 Algorithm Results

K LstmCP + LS Benchmark

The numerical results for the LstmCP + LS method together with the three methods detailed by Barber et al. (2023) are given here. Table 3 shows the results for the first three settings. Following that, Table 4 displays the results for the final three settings.

	Setting 1 IID Data		Setting 2 Changepoints		Setting 3 Distribution Drift	
	Coverage	Width	Coverage	Width	Coverage	Width
CP + LS	0.892	3.188	0.897	9.251	0.781	4.303
NexCP + LS	0.897	3.277	0.888	8.726	0.864	5.430
NexCP + WLS	0.899	3.307	0.894	4.117	0.895	3.275
LstmCP + LS	0.893	3.176	0.899	8.956	0.837	5.021

Table 3: Prediction Results for Settings 1-3

	Setting 4		Setting 5		Setting 6	
	Stochastic Volatility		Heteroskedasticity		Lagged Effects	
	Coverage	Width	Coverage	Width	Coverage	Width
CP + LS	0.905	3.653	0.904	3.642	0.894	3.532
NexCP + LS	0.900	3.567	0.898	3.609	0.897	3.563
NexCP + WLS	0.897	3.607	0.897	3.637	0.895	3.591
LstmCP + LS	0.900	3.584	0.898	3.539	0.887	3.439

Table 4: Prediction Results for Settings 4-6

L Adaptive Weight Plots

As mentioned in Section 5.1, for the stochastic volatility, heteroskedasticity, and lagged effects settings, we expect the influence of the conditional variance to even out in plots. This is due to the fact that a rolling average of 300 observations is used for plotting. Therefore, the weight plots for these settings are here just for completeness and should not be unduly interpreted.

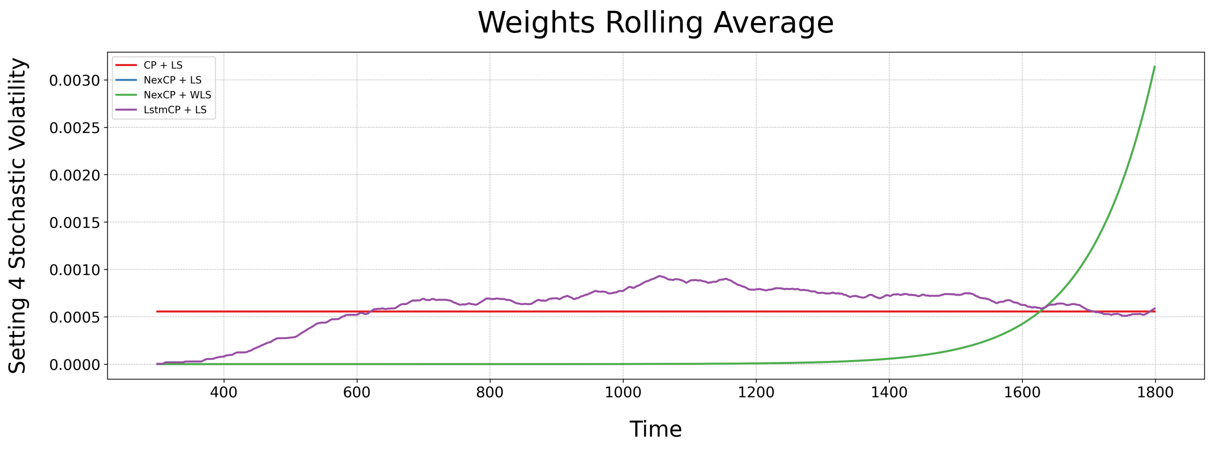

Figure 15: Weights for Stochastic Volatility

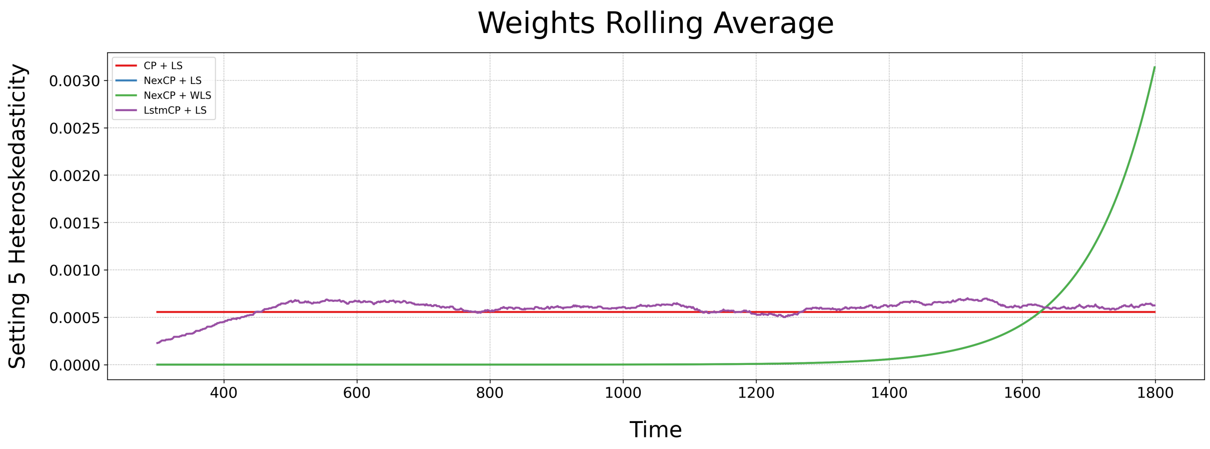

Figure 16: Weights for Heteroskedasticity

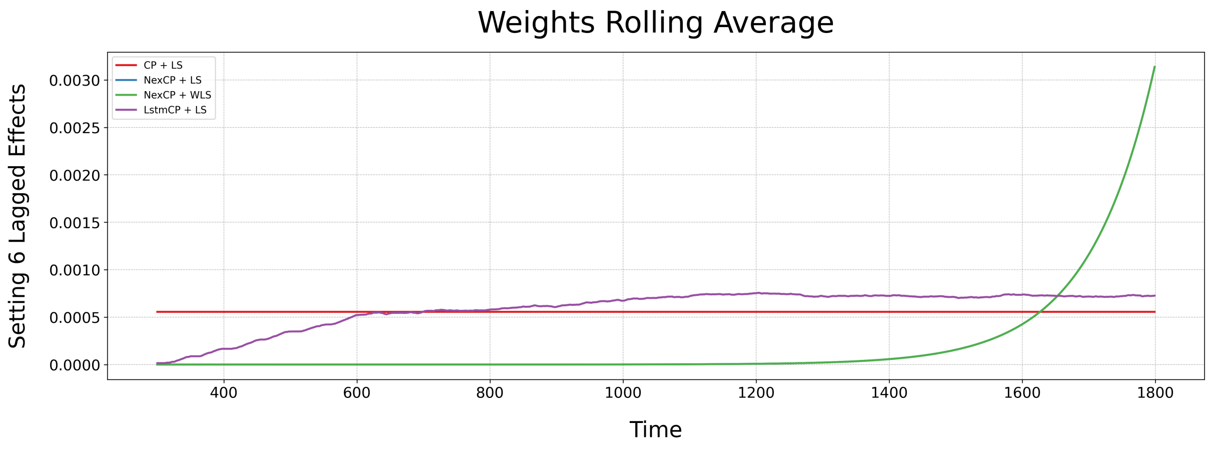

Figure 17: Weights for Lagged Effects