MONASH University

Optimal Predictor Selection for High-dimensional Nonparametric Forecasting

Nuwani Kodikara Palihawadana
B.Sc. (Hons), University of Colombo, Sri Lanka

A thesis submitted for the degree of
Doctor of Philosophy
at Monash University in 2025
Department of Econometrics & Business Statistics

Table of contents

Copyright notice

Abstract

Forecasting in real-world settings often involves modelling time series data characterised by dynamic and evolving patterns. Achieving accurate forecasts typically requires incorporating diverse predictors that may interact with the target series in complex and non-linear ways. While deep learning methods have gained popularity for such tasks, these models often function as black boxes, offering little insight into the underlying mechanisms. This emphasises the need for statistical approaches that offer similar capabilities while providing greater transparency in model specification and estimation.

Nonparametric additive models have recently emerged as an effective and adaptable method for forecasting and prediction tasks. However, their practical application faces notable difficulties. One major challenge is identifying the relevant predictors to include in the model when working with high-dimensional data. Another important issue is determining how to group predictors to capture their interactions appropriately. To help bridge this gap, nonparametric additive index models are explored as a forecasting framework in this thesis, with a particular focus on automating predictor selection, predictor grouping, and model estimation, thereby reducing dependence on prior knowledge or domain expertise.

This thesis offers three main contributions. First, it introduces the Sparse Multiple Index (SMI) Modelling algorithm, which estimates nonparametric additive index models via an ℓ_0- and ℓ_2-regularised non-linear least squares optimisation problem with linear constraints, solved using mixed integer programming. The algorithm performs both predictor selection and grouping with minimal user input. Empirical applications to forecasting heat exposure-related daily mortality and daily solar intensity demonstrate the method's competitive forecasting performance. Owing to its objective and flexible estimation approach, the proposed SMI model accommodates a broad spectrum of additive structures—from single index models to fully additive models—making it a general estimation framework suitable for nonparametric additive models. While its application in this thesis focuses on forecasting, the model itself is not tailored to time series data and can readily be applied to cross-sectional prediction tasks, highlighting its versatility as a statistical tool.

Second, the thesis investigates methods for generating prediction intervals to quantify forecast uncertainty in nonparametric additive models. It evaluates the standard block bootstrap approach and three established conformal prediction methods. In addition, it introduces a novel method—Conformal Bootstrap (CB)—that brings together block bootstrap resampling and split conformal prediction in a cohesive procedure for constructing prediction intervals. Based on the two empirical applications presented, the CB method generally yields prediction intervals with improved calibration and sharpness over those generated by the block bootstrap approach, and performs comparably to conformal prediction methods in most scenarios.

As a third contribution, this thesis introduces an open-source R package, `smimodel`, to promote the practical use of the proposed methodologies. The package implements the SMI Modelling algorithm, along with tools for generating prediction intervals using both the block bootstrap and conformal bootstrap methods. The development of this software ensures that the proposed methods are accessible, reproducible, and extensible by the wider research and forecasting community.

Declaration

This thesis is an original work of my research and contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Student name: Nuwani Kodikara Palihawadana

Date: 17.06.2025

The contribution in Chapter 2 of this thesis was presented in the International Symposium on Forecasting held in Dijon, France, in July 2024.

Chapter 2 is ready for submission to an appropriate forecasting or computational statistics journal. As a result, it is self-contained and includes relevant material for completeness; therefore, readers may encounter some repetition with content presented in other chapters.

Reproducibility statement

This thesis is written using Quarto (Allaire et al. 2024). All materials (including the data sets and source files) required to reproduce this document can be found at the Github repository `github.com/nuwani-palihawadana/Nuwani-Thesis`.

Acknowledgements

This thesis reflects a journey of dedication, perseverance, and personal growth over the past four years; a journey that would not have been possible without the unwavering support of those closest to my heart, including some incredible individuals I was fortunate to meet along the way.

First and foremost, I express my sincere gratitude to my supervisor, Professor Rob J. Hyndman, for his incomparable mentorship. Throughout my PhD journey, one recurring comment I made to my husband was, "Rob can tell I'm struggling, but he waits until I'm nearly at the edge before stepping in." It became something I would say with a smile—a genuine expression of appreciation for how you challenged me to grow. Your approach to mentoring pushed me to develop resilience and independence. While it was challenging at times, the process taught me invaluable lessons that I would not have learnt otherwise. Thank you for believing in my ability to navigate difficulties, even when I doubted myself. I am immensely grateful and honoured to have worked under your guidance. Your expertise, way of thinking, and leadership will continue to inspire me now and always.

I am deeply grateful to my associate supervisor, Dr Xiaoqian Wang, for her generous and invaluable support. Although we started working together halfway through my journey, the impact you had on both my research and me during that time was extraordinary. Your knowledge, commitment, and academic insight will continue to inspire me for many more years to come. I would also like to extend my heartfelt gratitude to my external supervisor Professor Louise M. Ryan. Despite her busy schedule, she has always been just an email away whenever I needed guidance. I am especially grateful for the times she was there to be the best sounding board I could ask for, particularly when Rob was away on academic trips and conferences.

I gratefully acknowledge the constructive feedback, suggestions, and encouragement I received from my PhD milestone evaluation panel members: Emeritus Professor Gael Martin, Dr Patricia Menéndez, Professor Dianne Cook, Professor Catherine Forbes, Dr Didier Nibbering, Professor Xibin Zhang, and Dr Jack Jewson. Special thanks go to Emeritus Professor Gael Martin, Professor Catherine Forbes, and Professor Xibin Zhang for being outstanding PhD directors and for their immense support throughout

my candidature. I am also thankful to Professor Mervyn Silvapulle and Emeritus Professor Donald Poskitt, who taught our PhD coursework units during the first year of the programme.

I extend my sincere thanks to Dr Shanika Wickramasuriya, whose help during a particularly challenging time in my PhD is deeply appreciated. Her unbounded willingness to share knowledge and the generous time she devoted to helping me navigate difficult problems made a real difference when I needed it most. I remain incredibly grateful for her guidance and friendship.

I am thankful to Monash University, Australia, for this invaluable opportunity to pursue my PhD at such a prestigious institution. The exposure and experiences I gained over the years at Monash have been immense. I am particularly grateful for the financial support provided through the Co-funded Monash Graduate Scholarship and the Monash Business School Co-Funded Graduate Research Scholarship, which enabled me to concentrate fully on my research. I am also thankful for the financial assistance provided to attend international conferences.

I am deeply grateful for the kind support and assistance I received from the Graduate Research Team of the Faculty of Business and Economics, during particularly difficult periods of my candidature. I would especially like to thank Cornelius Ngau, who went above and beyond to support me, providing reassurance and handling administrative matters with such efficiency that it gave me invaluable peace of mind. I also gratefully acknowledge the support of the Monash eResearch Centre for providing access to the MonARCH (Monash Advanced Research Computing Hybrid) HPC Cluster. Special thanks to Simon Michnowicz and Phillip Chan for their patience and prompt assistance whenever I encountered technical issues. Many thanks to the NUMBATs research team for creating a supportive and enjoyable research environment. I would also like to thank the administrative staff of the Department of Econometrics and Business Statistics for their support and assistance. I also acknowledge with gratitude the travel award provided by the International Institute of Forecasters to attend the 44th International Symposium on Forecasting in Dijon, France.

I acknowledge the use of Grammarly and ChatGPT to check for grammatical and spelling errors. The output from these tools was used to improve the grammatical accuracy of my writing.

I remember with gratitude, the University of Colombo, Sri Lanka, for paving the way for my academic journey. I am especially grateful to my undergraduate research supervisor, Dr G.P. Lakraj, for sparking my interest in time series analysis and forecasting, and for broadening my academic horizons through his wisdom and mentorship. I am also grateful to St. Paul's Girls' School, Milagiriya, Sri Lanka, where I received my primary and secondary education. The life lessons and challenges I encountered there shaped me into a stronger individual, something that proved invaluable throughout my PhD journey.

I am thankful to my fellow PhD students for making my time at Monash a memorable experience. Special thanks to Dr Yangzhuoran (Fin) Yang for generously spending hours helping me debug my codes whenever I ran to him for help, and to Dr Heshani Madigasekara and Jayani Piyadi Gamage for constantly being my moral support system at the office. Thank you, Chaya, Himasha, and Hiruni, for being my partners in all the crazy explorations around campus, unforgettable trips, and countless chats. Thank you, Mitch, for being the most understanding and supportive head tutor I could have asked for, and for generously relieving me of tutoring duties when I needed extra time for my thesis. Thank you, Janith, for always being willing to answer my technical questions.

To my parents and my sister; your love and strength have been the driving force behind every step I have taken. Your endless support, quiet sacrifices, and unwavering belief in me have carried me through every challenge, both in this academic journey and in life. No matter how I stumble, you have always lifted me up and reminded me of my worth. Amma, Thaththa, and Nangi—I am endlessly grateful for you, beyond words, beyond life, and beyond everything.

Last but by no means least, my deepest gratitude goes to my husband, Rehan Mendis. Being thousands of miles away from my parents and sister, you became my home and my anchor. You carried the weight of my stress, mood swings, frustration, and the occasional midnight meltdown. Thank you for holding me through it all—and never letting go.

This journey was never mine alone, and for that, I am eternally grateful!

Chapter 1

Introduction

Time series forecasting is a key tool that supports strategic decision-making across diverse fields, including but not limited to preventive maintenance, finance, retail and power systems planning (Jakobs & Liebig 2024; Godahewa et al. 2021; Lim & Zohren 2021; Fan & Hyndman 2012). These real-world forecasting problems often involve time series with complex and dynamic patterns over time. Accurately forecasting these series typically requires the consideration of multiple predictors that have intricate non-linear relationships with the time series of interest. For example, temperature is typically an important predictor variable in forecasting electricity demand, and the relationship between electricity demand and temperature is often non-linear (Hyndman & Fan 2010; Fan & Hyndman 2012).

In recent years, research has increasingly shifted towards deep learning-based forecasting methods, moving away from classical statistical models for time series forecasting (Jakobs & Liebig 2024; Lim & Zohren 2021). This shift is driven by the argument that deep learning techniques are better able to capture intertemporal and non-linear dependencies in the data (Elsayed et al. 2021). Moreover, the estimation of traditional models relies heavily on the domain knowledge and experience of the forecaster. In contrast, machine learning approaches identify temporal patterns through data-driven techniques (Lim & Zohren 2021). This distinction has further facilitated the transition towards deep learning methods, particularly due to the increasing availability of data and the continuous advancement of computational capabilities.

However, researchers have begun to critically evaluate the use of deep learning methods in real-life forecasting problems. The increasing complexity of these methods makes it almost impossible to identify the underlying model-building process, or fully understand the rationale behind their

performance, which is particularly problematic in risk-intensive decision-making contexts (Jakobs & Liebig 2024; Rudin 2019).

These circumstances motivated this research, which aims to develop a statistical model capable of capturing complex temporal dynamics and non-linear predictor–response relationships in real-world time series forecasting problems, while providing transparency in its structure and estimation process. Although nonparametric additive index models are not specific to time series forecasting, our research has shown that they are capable of capturing non-linear relationships and handling complex interactions between variables, while maintaining the necessary clarity in model construction. However, we observed a lack of objectivity in the selection of predictor variables in high-dimensional scenarios and in the grouping of predictors to model interactions, particularly in predictive applications. Therefore, the broader aim of this thesis is to investigate nonparametric additive index models as a forecasting tool in high-dimensional settings, while automating predictor selection, predictor grouping, and model estimation where possible, thereby reducing the necessity to utilise domain expertise or prior knowledge in the process.

1.1 Background

This section provides an overview of relevant background work related to the overall focus of this thesis. While not all the statistical models and methodologies outlined here are directly utilised in the thesis, they are included here to provide context for the research presented in Chapters 2–4.

In the aforementioned real-world forecasting problems with omnipresent non-linear predictor–response relationships, it is typical to consider a long history of predictors; i.e. several lags of the predictors into the forecasting model in addition to their current values. For example, forecasting half-hourly electricity demand often requires at least a week of historical temperature measurements (Hyndman & Fan 2010).

In this case, a non-linear *transfer function model* is appropriate, which can be stated as

$$y_t = f(\boldsymbol{a}_t, \boldsymbol{a}_{t-1}, \ldots, \boldsymbol{a}_{t-s}, y_{t-1}, \ldots, y_{t-k}) + \varepsilon_t,$$

where y_t is the observation of the response variable at time t, f is any arbitrary function (typically non-linear and smooth), $\boldsymbol{a}_t$ is a vector of predictors at time t, and ε_t is an independent and identically distributed (i.i.d.) random error. Note that the above model formation includes lagged values of the response y_t along with the lagged predictors, which may help to capture any serial correlation that is present in the data.

Estimating the nonparametric function f becomes infeasible in high-dimensional settings due to the curse of dimensionality (Bellman 1957; Stone 1982). As a result, *nonparametric additive models*, which impose an additivity constraint on the estimation of f, have become increasingly popular.

Let $(y_i, \boldsymbol{x}_i), i = 1, \ldots, n$, be i.i.d. observations, and $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})^T$ be a p-dimensional vector of predictor values. Then a nonparametric additive model can be written as

$$y_i = \sum_{j=1}^{p} f_j(x_{ij}) + \varepsilon_i, \quad i = 1, \ldots, n, \tag{1.1}$$

where f_j, $j = 1, \ldots, p$ are unknown functions (usually non-linear and smooth), and ε_i is the random error (Lian 2012). Even when such an additivity condition is imposed, estimating the optimal predictive model will still be troublesome when p is very large (probably even larger than the sample size n) due to over-fitting (Lian 2012). Thus, it is natural to bring in the sparsity assumption, and assume that some of the f_j functions are zero, which raises the need for a variable selection method to differentiate between zero and non-zero components (Huang, Horowitz & Wei 2010).

1.1.1 Variable selection

Backward elimination

In the problem of forecasting long-term peak electricity demand, Hyndman & Fan (2010) used a backward elimination procedure for variable selection through cross-validation. For each half-hourly model fitted, the data is split into training and validation sets, and the predictors are selected in the model based on the Mean Squared Error (MSE) calculated for the validation set. Starting from the full model, the predictive power of each variable is evaluated by sequentially dropping one variable at a time. If the removal of a predictor leads to a decrease in the validation MSE, it is omitted from the model in subsequent steps (Hyndman & Fan 2010). Fan & Hyndman (2012) used a similar method except for the fact that they considered the Mean Absolute Percentage Error (MAPE) as the selection criterion. Therefore, these previous works use backward elimination for variable selection, relying on out-of-sample forecasting performance.

Penalisation methods

According to Huang, Horowitz & Wei (2010), there are numerous penalised methods for variable selection and parameter estimation in high-dimensional settings, including the *bridge estimator* proposed by Frank & Friedman (1993), the *Least Absolute Shrinkage and Selection Operator (LASSO)* by Tibshirani (1996), the *Smoothly Clipped Absolute Deviation Penalty (SCAD)* by Fan & Li (2001), and the *Minimum Concave Penalty (MCP)* by Zhang (2010). Among them, the LASSO and the SCAD penalties are quite popular in the literature.

Tibshirani (1996) introduced the regularisation method, LASSO, for estimating linear models, which minimises the sum of squared residuals subject to an ℓ_1 penalty on the coefficients. Consider the classical linear regression model $y_i = \sum_{j=1}^{p} \beta_j x_{ij} + \varepsilon_i$, where the model is fitted to the data $(y_i, \boldsymbol{x_i})$, $i = 1, \ldots, n$. Here, y_i is the response, $\boldsymbol{x_i} = (x_{i1}, \ldots, x_{ip})^T$ is a p-dimensional vector of predictors, $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_p)^T$ is the parameter vector corresponding to $\boldsymbol{x_i}$, and ε_i is the random error. Then, the LASSO estimator, $\hat{\boldsymbol{\beta}}_{LASSO}$, can be obtained by

$$\hat{\boldsymbol{\beta}}_{LASSO} = \min_{\boldsymbol{\beta}} \left\{ \left\| \boldsymbol{y} - \sum_{j=1}^{p} \boldsymbol{x_j} \beta_j \right\|_2^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\},$$

where $\boldsymbol{x_j} = \left(x_{1j}, \ldots, x_{nj}\right)^T$, and λ is a non-negative tuning parameter. The LASSO estimator reduces to the Ordinary Least Squares (OLS) estimator when λ is equal to zero (Konzen & Ziegelmann 2016). Due to the nature of the penalty applied, LASSO shrinks some of the coefficients towards zero and sets the others exactly to zero, where the estimation of the coefficients and the selection of the variables are performed simultaneously (Konzen & Ziegelmann 2016).

While showing that the LASSO is inconsistent for variable selection in certain situations, Zou (2006) introduced *Adaptive LASSO* (popularly known as *adaLASSO*); an extension of the LASSO method that uses adaptive weights to penalise coefficients with the ℓ_1 penalty. Thus, the adaLASSO objective function can be written as

$$\hat{\boldsymbol{\beta}}_{adaLASSO} = \min_{\boldsymbol{\beta}} \left\{ \left\| \boldsymbol{y} - \sum_{j=1}^{p} \boldsymbol{x_j} \beta_j \right\|_2^2 + \lambda \sum_{j=1}^{p} w_j |\beta_j| \right\},$$

where the vector of weights $\boldsymbol{w} = \left(w_1, \ldots, w_p\right)^T$ is estimated by $\hat{\boldsymbol{w}} = 1/|\hat{\boldsymbol{\beta}}|^\gamma$ for $\gamma > 0$, where γ is a tuning parameter, and $\hat{\boldsymbol{\beta}}$ is any consistent estimator of $\boldsymbol{\beta}$ (Zou 2006).

Yuan & Lin (2006) considered the problem of selecting groups of variables and discussed extensions of three variable selection and estimation methods namely, *LASSO* (Tibshirani 1996), *Least Angle Regression Selection* (*LARS*, Efron et al. 2004), and *Non-Negative Garrotte* (Breiman 1995). Consider an n-dimensional response vector $\boldsymbol{y}$, and an $n \times p$ matrix of predictor values $\boldsymbol{X}$. Then the *Group LASSO* estimator of the coefficient vector $\boldsymbol{\beta}$ is obtained by minimising

$$\frac{1}{2} \left\| \boldsymbol{y} - \sum_{\ell=1}^{L} \boldsymbol{X}_\ell \boldsymbol{\beta}_\ell \right\|_2^2 + \lambda \sum_{\ell=1}^{L} \|\boldsymbol{\beta}_\ell\|_{K_\ell},$$

where $\boldsymbol{X}_\ell$ is an $n \times p_\ell$ submatrix in $\boldsymbol{X}$ corresponding to the ℓ^{th} group of predictors (p_ℓ is the number of predictors in the ℓ^{th} group), $\boldsymbol{\beta}_\ell$ is the corresponding vector of coefficients, $\|\boldsymbol{\beta}_\ell\|_{K_\ell} = (\boldsymbol{\beta}_\ell^T \boldsymbol{K}_\ell \boldsymbol{\beta}_\ell)^{\frac{1}{2}}$ with

$\boldsymbol{K}_1, \ldots, \boldsymbol{K}_L$ being a set of given positive definite matrices, L is the number of predictor groups, and λ is a non-negative tuning parameter. In addition, Simon et al. (2013) proposed *Sparse-Group LASSO*, which is a convex combination of general LASSO and Group LASSO methods, focusing on both *groupwise sparsity* (the number of groups with at least one non-zero coefficient), and *within-group sparsity* (the number of non-zero coefficients within each non-zero group).

According to Fan & Li (2001), a penalty function used in penalised least squares approaches should have three properties. First, it should be singular at the origin to generate a sparse solution. Second, it should fulfil certain conditions to be stable in model selection. Finally, it should be able to produce unbiased estimates for large coefficients by being bounded by a constant. They argued that all these three conditions are not satisfied by several penalisation methods, such as the bridge regression (Frank & Friedman 1993) and the LASSO (Tibshirani 1996). Therefore, they proposed the SCAD penalty function, which is defined in terms of its first derivative as

$$p'_\lambda(\theta) = \lambda \left\{ I(\theta \leq \lambda) + \frac{(a\lambda - \theta)_+}{(a-1)\lambda} I(\theta > \lambda) \right\},$$

for some $a > 2$, and $\theta > 0$ (Fan & Li 2001). According to Fan & Li (2001), the SCAD penalty function retains the favourable properties of both best subset selection and ridge regression, while having all three desired properties, i.e. sparsity, stability, and unbiasedness.

Based on the above penalisation methods originally developed for linear models, Huang, Horowitz & Wei (2010) proposed a new penalisation method for variable selection in the nonparametric additive model (Equation 1.1) called *Adaptive Group LASSO*. They approximated the f_j functions using normalised B-spline bases so that a linear combination of B-spline basis functions is used to represent an individual nonparametric component f_j. The proposed method is a generalisation of the Adaptive LASSO method (Zou 2006) to the Group LASSO method (Yuan & Lin 2006).

When considering the nonparametric additive model in Equation 1.1, an obvious possibility is that some of the additive components (i.e. f_j) are linear. For example, when forecasting the electricity demand, Fan & Hyndman (2012) included calendar effects such as the day of the week and holiday effect as linear variables in the model, while lagged temperature and lagged demand variables were non-linear additive components. Such situations suggest the use of *semiparametric partially linear additive models*, which can be expressed mathematically as

$$y_i = \sum_{j=1}^{p} f_j(x_{ij}) + \sum_{k=1}^{b} u_{ik}\beta_k + \varepsilon_i, \quad i = 1, \ldots, n,$$

where $\boldsymbol{x}_j$, $j = 1, \ldots, p$, is a set of predictors entering the model as nonparametric components, while $\boldsymbol{u}_k$, $k = 1, \ldots, b$, is another set of predictors entering the model as linear components. While several studies have assumed that the number of nonparametric components is fixed, and performed variable selection only among the linear components of the model (Lian 2012; Guo et al. 2013; Liu, Wang & Liang 2011), Wang et al. (2014) introduced a methodology for selecting both linear and non-linear components simultaneously, in the context of correlated, longitudinal data. They proposed the use of a *Penalised Quadratic Inference Function (PQIF) with double SCAD penalties* for variable selection and model estimation, where the correlation structure of the data was incorporated into the estimation method (see Wang et al. (2014) for details).

Time series aspect

It is worth mentioning briefly that there are extensions to the penalisation methods discussed above that have been specifically proposed to account for autocorrelation and lag structures in time series data.

Wang, Guodong & Tsai (2007) proposed an extension of LASSO to *Regression with Autoregressive Error (REGAR)* models. Park & Sakaori (2013) and Konzen & Ziegelmann (2016) proposed modifications to the Adaptive LASSO method to incorporate the lag structures presented in *Autoregressive Distributed Lag (ADL)* models into the variable selection and estimation methodology. The *Ordered LASSO* was introduced by Tibshirani & Suo (2016) to deal with time-lagged regression problems, where we forecast the response value at time t using the predictor values from a few previous time points, assuming that the magnitude of the regression coefficients decreases as the lagged predictor moves away from time t.

However, it is important to note that all the models considered in the above time series-related work are linear; none of them include nonparametric terms.

Mathematical optimisation for variable selection

Mathematical optimisation

Optimisation plays an important role in both decision science and physical systems evaluation. *Mathematical Optimisation* or *Mathematical Programming* can be defined as the minimisation (or maximisation) of a function subject to restrictions on the unknowns/parameters of that function (Nocedal & Wright 2006). Thus, a mathematical optimisation problem can be written as

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f_0(\boldsymbol{x}) \\ \text{s.t.} \quad & f_i(\boldsymbol{x}) \leq b_i, \quad i = 1, \ldots, m \end{aligned} \tag{1.2}$$

where the vector of unknowns or parameters of the problem is given by $\boldsymbol{x} = (x_1, \ldots, x_n)^T$, the *objective function* is given by $f_0 : \mathbb{R}^n \to \mathbb{R}$, the *constraint functions* are given by $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$, and the bounds of the constraints are given by $\boldsymbol{b} = (b_1, \ldots, b_m)^T$. A vector of values $\boldsymbol{x}^*$ that results in the smallest value for the objective function among all vectors that satisfy the stated constraints is called the *optimal* value or the *solution* to the problem (Boyd & Vandenberghe 2004). After mathematically formulating the optimisation problem as above (Equation 1.2), an appropriate *optimisation algorithm* is used to obtain the solution $\boldsymbol{x}^*$ (Nocedal & Wright 2006).

Based on the form of the objective function and the constraints, different types of optimisation problems are identified.

An optimisation problem is identified to be a *Linear Programming (LP)* if both the objective function and the constraints in Equation 1.2 (i.e. all f_i, $i = 0, \ldots, m$) are linear. Thus, an LP can be written as

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{a_0}^T \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \end{aligned} \tag{1.3}$$

where $\boldsymbol{x}$ is the vector containing the parameters to optimise, and $\boldsymbol{a_0}$ and $\boldsymbol{b}$ are given vectors, and $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is a given matrix. All LPs are *convex* optimisation problems (Theußl, Schwendinger & Hornik 2020).

The LP problem given in Equation 1.3 can be generalised to include a quadratic term in the objective function, in which case it is called *Quadratic Programming (QP)*. A QP can be written as

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \frac{1}{2} \boldsymbol{x}^T \boldsymbol{Q_0} \boldsymbol{x} + \boldsymbol{a_0}^T \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \end{aligned}$$

where $\boldsymbol{Q_0} \in \mathbb{R}^{n \times n}$. Unless the matrix $\boldsymbol{Q_0}$ is positive semidefinite, a QP is non-convex (Theußl, Schwendinger & Hornik 2020).

When a linear objective function is minimised over a *convex cone*, such an optimisation problem is called a *Conic Programming (CP)*, which can be written as

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{a_0}^T \boldsymbol{x} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{x} + s = \boldsymbol{b}, \quad s \in \mathscr{K}, \end{aligned}$$

where $\mathscr{K}$ denotes a non-empty closed convex cone. CPs are designed to model convex optimisation problems (Theußl, Schwendinger & Hornik 2020).

If we restrict some of the unknowns/parameters in an optimisation problem to take only integer values, then the optimisation problem is called a *Mixed Integer Programming (MIP)*. For example, if we constrain $x_k \in \mathbb{Z}$ for at least one k, $k \in \{1, \ldots, n\}$ in the optimisation problem given by Equation 1.2, then the optimisation problem becomes an MIP. If all unknowns of an optimisation problem are constrained to be integers, such a problem is called a pure *Integer Programming (IP)*, whereas if all the unknowns are bounded between zero and one (i.e. $\boldsymbol{x} \in \{0, 1\}^n$), the optimisation problem is called a *Binary (Integer) Programming* (Theußl, Schwendinger & Hornik 2020). MIPs are difficult to solve as they are non-convex due to the integer constraints. However, a growing number of commercial and non-commercial MIP solvers have made it possible to solve MIP problems conveniently and directly.

Variable selection using optimisation techniques

Mathematical optimisation is of fundamental importance in statistics, as many statistical problems, including regression, classification, and other estimation/approximation problems, can be reinterpreted as optimisation problems (Theußl, Schwendinger & Hornik 2020). For example, the problem of variable selection—a long-standing interest of statisticians—has recently benefited from the use of optimisation concepts, in particular MIP and convex optimisation.

For example, Bertsimas, King & Mazumder (2016) used a mixed integer optimisation procedure to solve the classical best subset selection problem in linear regression. They developed a discrete optimisation method by extending modern first-order continuous optimisation techniques. The method can produce near-optimal solutions, which would serve as a warm start for an MIP algorithm that would select the best k features from p predictors. Similarly, Hazimeh & Mazumder (2020) developed fast and efficient algorithms based on coordinate descent and local combinatorial optimisation to solve the same best subset selection (or ℓ_0-regularised least squares) problem by reformulating local combinatorial search problems as structured MIPs.

In addition, Hazimeh, Mazumder & Radchenko (2023) proposed a groupwise variable selection method based on discrete mathematical optimisation, which is applicable to both ℓ_0-regularised linear regression and nonparametric additive models in a high-dimensional setting. They formulated the group ℓ_0-based estimation problem as a *Mixed Integer Second Order Cone Programming (MISOCP)*, and proposed a new customised *Branch-and-Bound (BnB)* algorithm (Land & Doig 1960; Little et al. 1963) to obtain the global optimal solution to the MISOCP.

Through the study of the above literature, we found that the mathematical optimisation-based algorithms reduce the computational cost of variable selection procedures in high-dimensional settings. This is largely due to the availability of efficient commercial solvers such as *Gurobi* and

CPLEX. This motivated us to focus on a mathematical optimisation based approach for the development of our variable selection methodology.

1.1.2 Index models

Single index model

The *Single Index Model*, a generalisation of the linear regression model where the linear predictor is replaced by a semiparametric component, is popularly used in the literature (Radchenko 2015). Let y_i be the response, and $\boldsymbol{x}_i$ be a p-dimensional predictor vector. Then the single index model can be written as

$$y_i = g\left(\boldsymbol{\alpha}^T \boldsymbol{x}_i\right) + \varepsilon_i, \quad i = 1, \ldots, n,$$

where $\boldsymbol{\alpha}$ is a p-dimensional vector of unknown coefficients (i.e. parameters), g is an unknown univariate function, and ε_i is the random error (Stoker 1986; Härdle, Hall & Ichimura 1993). The linear combination $\boldsymbol{\alpha}^T \boldsymbol{x}_i$ is called the *index*. The single index model is considered a viable alternative to the nonparametric additive model because it offers more flexibility and interpretability (Radchenko 2015).

According to Radchenko (2015), single index models have been widely used in scenarios with fairly low and moderate dimensionality, where the corresponding estimation and variable selection techniques are not directly applicable to the high-dimensional setting. The sum of squared error of the model, which is non-convex with respect to the index coefficients, is the main reason for the existence of a very limited number of methods in the high-dimensional case (Radchenko 2015). For a comprehensive summary of the available methods, we refer to Radchenko (2015).

Multiple index models

The nonparametric additive model (Equation 1.1) estimates the relationship between the response and the predictors using a sum of univariate non-linear functions corresponding to each predictor variable. It is therefore unable to handle the interactions among predictors that are ubiquitous in real-world problems (Zhang et al. 2008).

Projection pursuit regression

Friedman & Stuetzle (1981) introduced *Projection Pursuit Regression (PPR)* by extending the nonparametric additive model (Equation 1.1) to enable the modelling of interactions among predictor variables. PPR can also be seen as an extension of the single index model to an "additive index model", given by

$$y_i = \sum_{j=1}^{q} g_j(\boldsymbol{\alpha}_j^T \boldsymbol{x}_i) + \varepsilon_i, \quad i = 1, \ldots, n,$$

where y_i is the response, $\boldsymbol{x}_i$ is a p-dimensional predictor vector, $\boldsymbol{\alpha}_j = \left(\alpha_{j1}, \ldots, \alpha_{jp}\right)^T, j = 1, \ldots, q$ are p-dimensional projection vectors, $g_j, j = 1, \ldots, q$ are unknown univariate functions, and ε_i is the random error.

Instead of estimating a single index, PPR estimates multiple indices and relates them to the response through a sum of univariate non-linear functions. These indices are constructed through a *Projection Pursuit (PP)* (Kruskal 1969; Friedman & Tukey 1974) algorithm, which aims to find "interesting" low-dimensional projections of high-dimensional feature space, obtained by maximising an appropriate objective function or a "projection index" (Huber 1985).

According to Zhang et al. (2008), PPR increases the power of additive models in high-dimensional settings, but it has two major drawbacks. First, since PPR increases the freedom of the additive model, it tends to overfit in a situation, where there are many unimportant predictors. Second, the interpretation of the model estimated by PPR will be troublesome as there will be many non-zero elements in each projection vector $\boldsymbol{\alpha}_j$. To overcome these issues, Zhang et al. (2008) introduced an ℓ_1-regularised projection pursuit algorithm, where the resulting regression model is called *Sparse Projection Pursuit Regression (SpPPR)*. In SpPPR, an ℓ_1 penalty (i.e. a LASSO penalty) on the index coefficients is added to the cost function (the squared error) at each iteration of the PP algorithm, thereby performing variable selection and model estimation simultaneously. See Zhang et al. (2008) for further details.

Although Zhang et al. (2008) claims that the SpPPR algorithm can detect important predictors even in a noisy data set, our experiments show that it is not particularly scalable for large data sets with both a higher number of predictors and observations.

Groupwise additive index model

Although PPR introduces flexibility and the ability to model interactions among predictors in additive models, the indices obtained by PPR contain all available predictors. Hence, even with a variable selection mechanism such as SpPPR, PPR may produce indices possibly by mixing heterogeneous variables in a single linear combination, which makes little sense in terms of interpretability (Masselot et al. 2023).

Typically, in many real-world problems, natural groupings in predictor variables can be identified. For example, naturally interacting variables can be grouped together, such as multiple lags of a predictor, weather-related variables, and genes or proteins that are grouped by biological pathways in a biological study (Masselot et al. 2023; Wang, Xu & Zhu 2015).

This suggests the use of a *Groupwise Additive Index Model (GAIM)*, which can be written as

$$y_i = \sum_{j=1}^{q} g_j(\boldsymbol{\alpha}_j^T \boldsymbol{x}_{ij}) + \varepsilon_i, \quad i = 1, \ldots, n,$$

where y_i is the univariate response, $\boldsymbol{x}_{ij} \in \mathbb{R}^{l_j}$, $j = 1, \ldots, q$ are naturally occurring q groups of predictors, which are q non-overlapping subsets of $\boldsymbol{x}_i$, the vector of all predictors, $\boldsymbol{\alpha}_j$ is an l_j-dimensional vector of index coefficients corresponding to the index $h_{ij} = \boldsymbol{\alpha}_j^T \boldsymbol{x}_{ij}$, g_j is an unknown (possibly non-linear) component function, and ε_i is the random error, which is independent of $\boldsymbol{x}_i$ (Wang et al. 2015; Masselot et al. 2023).

Since GAIM uses groups of predictors that naturally or logically belong together to construct indices, such derived indices will be more expressive and interpretable. However, this also introduces a degree of subjectivity into the model formulation, as different users may group the available predictors in different ways based on different logical considerations.

1.1.3 Uncertainty estimation

The uncertainty of a point forecast is most commonly measured by a *prediction interval*. A prediction interval is an interval or a range of values, where we anticipate a future value of a time series to fall, with a given probability (Hyndman & Athanasopoulos 2021). In other words, a prediction interval is a probabilistic statement about an unknown future value, which can be considered a random variable at the forecast origin (Chatfield 1993).

Since the focus of this thesis is on nonparametric additive index models, this section outlines methods for generating prediction intervals that do not rely on any distributional assumptions about the data or model estimation.

Block bootstrap

The method of resampling from the empirical distribution of historical model residuals is known as *Bootstrap* (Chatfield 1993). The bootstrap method closely resembles the theoretical error distribution using the empirical distribution of observed residuals. It is therefore considered to be an approach that is free of distributional assumptions, which makes it appropriate for use in the context of nonparametric models.

When constructing prediction intervals in time series forecasting problems, it is crucial to retain the serial correlations present in the data, while performing bootstrapping. The conventional method for bootstrapping stationary time series is called *Block Bootstrap (BB)* (Politis 2003; Hyndman & Fan 2010). In block bootstrap, randomly resampled blocks from the historical time series are combined to create new simulated series. When applying this technique, the most important parameter choice is

the *block length*. The length of a block should be chosen to be adequate to capture the autocorrelation patterns within the time series, but short enough to permit the construction of many simulated series (Hyndman & Fan 2010).

If the time series to be resampled is seasonal, then the block length is often set to be a multiple of the seasonal period length, and this variant of block bootstrapping is called *seasonal* or *single season* block bootstrap (Politis 2001; Hyndman & Fan 2010). Seasonal block bootstrap assumes the stationarity of the seasonal subseries, implying that blocks, each of length equal to a multiple of the seasonal period and starting in the season, follow the same stationary distribution.

Conformal prediction

Conformal Prediction (CP), introduced by Vovk, Gammerman & Shafer (2005), is a distribution-free approach for generating prediction intervals from any arbitrary predictive model, relying only on the assumption of data exchangeability. Most importantly, CP provides theoretical guarantees on the marginal coverage of the constructed prediction intervals. As a result, CP has experienced a wave of popularity in recent years as a method to quantify the uncertainty of predictions obtained by any statistical learning or black-box machine learning model, including nonparametric models (Shafer & Vovk 2008; Papadopoulos 2008; Lei & Wasserman 2014). For theoretical details, readers are referred to Vovk, Gammerman & Shafer (2005) and Shafer & Vovk (2008).

The original CP method, which is known as *Full Conformal Prediction* or *Transductive Conformal Prediction* (Vovk, Gammerman & Shafer 2005), is a computationally intensive method. A widely used computationally efficient alternative is the *Split Conformal Prediction (SCP)*, which is also known as *Inductive Conformal Prediction* (Papadopoulos et al. 2002; Lei et al. 2018).

SCP is a holdout method for generating prediction intervals. That is, SCP divides the data set into non-overlapping training, calibration, and test sets, and trains a prediction model on the training set. Then the prediction errors (or *non-conformity scores*) are calculated on the calibration set (i.e. the *holdout* set) using the predictions obtained by the estimated model. Finally, the calculated quantiles of these calibration set errors are used to construct the prediction intervals of desired coverage for the new observations in the test set. It is important to note, however, that while SCP is computationally efficient, this advantage comes at the cost of reduced statistical efficiency due to data partitioning. For a comprehensive summary, see Barber et al. (2023), and for algorithmic details, see Lei et al. (2018).

As previously stated, the original theory of CP is based on the assumption that the data are exchangeable. Hence, it cannot be applied to scenarios where the underlying distribution of the data changes across the data set, or where the data are simply not exchangeable; for example time series data

with autocorrelation. A number of previous studies have therefore extended CP theory to deal with non-exchangeable data.

Tibshirani et al. (2019) proposed a weighted CP method for the case where the training and test sets have different covariate (X) distributions, but have the same conditional distribution of the response Y given X, which they called the "covariate shift" assumption. In this method, the calculated non-conformity scores are weighted using the ratio of the likelihoods of the training and test covariate distributions, where this likelihood ratio is assumed to be known or accurately estimated. Moreover, Barber et al. (2023) introduced a method of weighting quantiles to avoid the assumption of exchangeability. In contrast to Tibshirani et al. (2019), where the weights for the non-conformity scores are data dependent, in Barber et al. (2023), the weights are considered fixed. The proposed method has been shown to handle violations of the assumption of exchangeability, as long as these violations are marginal. We refer to this method as *Weighted Split Conformal Prediction (WSCP)* in Chapter 3.

Using an online updating procedure to handle the distribution shifts in the data, Gibbs & Candès (2021) introduced *Adaptive Conformal Prediction (ACP)*. Here, they update the nominal significance level, α, at each updating step by comparing the achieved coverage at the previous step with the target coverage, $1-\alpha$. Informally, depending on whether the achieved coverage is greater or less than the target coverage in the previous iteration, ACP will increase or decrease the target level α respectively, for the next iteration. Building on this idea, subsequent research, including Zaffran et al. (2022) and Gibbs & Candès (2024), introduced time-dependent step sizes to account for arbitrary distribution shifts.

In addition, Angelopoulos, Candes & Tibshirani (2024) proposed *Conformal PID Control (PID)*, another online updating framework designed to adapt to seasonality, trends, and general distributional shifts in the data. While this method has been shown to provide favourable results in time series settings, it was originally developed for one-step-ahead forecasting. Furthering this work, Wang & Hyndman (2024) introduced *Autocorrelated Multi-step Conformal Prediction (AcMCP)*, which generates statistically efficient multi-step prediction intervals by taking into account the inherent autocorrelations in multi-step forecast errors.

1.2 Objectives and thesis outline

The primary aim of this thesis is to explore nonparametric additive index models as a forecasting tool in a high-dimensional context, where predictor variables exhibit non-linear and complex interactions with the response variable of interest. In pursuit of this goal, the objectives and the structure of the thesis can be outlined as follows.

The **first** objective of this thesis is to develop an optimal predictor selection methodology for high-dimensional nonparametric additive index models, which will potentially minimise the need to use domain expertise or prior knowledge in the model estimation process. Chapter 2 addresses this objective by proposing a novel algorithm for nonparametric additive index model estimation, named *Sparse Multiple Index (SMI)*[1] *Modelling* algorithm, which is based on solving an ℓ_0- and ℓ_2-regularised non-linear least squares optimisation problem with linear constraints. The proposed methodology estimates nonparametric additive index models by algorithmically selecting and grouping predictors into indices, with minimal input information supplied, while resulting in a model with competitive forecasting accuracy compared to existing nonparametric additive models.

In forecasting, it is necessary to produce prediction intervals to quantify the uncertainty of the point forecasts generated. Typically, the constructed prediction intervals are underestimated due to variable selection processes used in the model estimation. Therefore, the **second** objective of this thesis is to study and explore methods to produce prediction intervals for uncertainty estimation in nonparametric additive index models, while addressing the aforementioned problem of underestimation. The Chapter 3 of the thesis will focus on addressing this objective. The standard block bootstrap method and three existing conformal prediction techniques are tested for constructing prediction intervals. In addition, we introduce a novel approach, *Conformal Bootstrap (CB)*, which combines *block bootstrap* with *split conformal prediction* in a unified framework for generating prediction intervals.

Chapter 2 and Chapter 3 have established the core workflow for time series forecasting with predictor variables using nonparametric additive index models, by focusing on the generation of point forecasts through the development of an estimation algorithm and exploring techniques to quantify the uncertainty of these forecasts. The next step is to make these proposed methods available to potential users.

We believe that open-source software is the best way to make these methodologies accessible to a wider audience. Open-source software gives users access to the source code, allowing them to freely use, modify, extend, debug, and redistribute the software as desired (Nelson, Sen & Subramaniam 2006). More importantly, it facilitates reproducibility and knowledge sharing for further research and

[1] The acronym *SMI* is pronounced as *smi* (as in 'smile').

development. Hence, the **third** objective of this thesis is to automate the methodologies presented in Chapter 2 and Chapter 3 by developing an open-source R package, given the popularity of R in the field of forecasting (Hyndman & Athanasopoulos 2021; Shmueli & Polak 2024). This third objective forms the Chapter 4 of the thesis, which introduces the R package `smimodel` that implements the SMI Modelling algorithm proposed in Chapter 2, and the block bootstrap and conformal bootstrap methods for generating prediction intervals discussed in Chapter 3. (Note that we used the R package `conformalForecast` (Wang & Hyndman 2025) to apply existing conformal prediction methods in Chapter 3.)

Conclusions are drawn in Chapter 5, along with a discussion of the results and directions for potential future research.

Chapter 2

Sparse Multiple Index Models for High-dimensional Nonparametric Forecasting

Forecasting often involves high-dimensional predictors that have non-linear relationships with the outcome of interest. Nonparametric additive index models can capture these relationships while addressing the curse of dimensionality. This chapter introduces a new algorithm, ***S***parse ***M***ultiple ***I***ndex (***SMI***) ***Modelling***, tailored for estimating high-dimensional nonparametric/semiparametric additive index models while limiting the number of parameters to be estimated by optimising predictor selection and predictor grouping. The SMI Modelling algorithm uses an iterative approach based on mixed integer programming to solve an ℓ_0- and ℓ_2-regularised non-linear least squares optimisation problem with linear constraints. The performance of the proposed algorithm is demonstrated through a simulation study and two empirical applications: forecasting heat exposure-related daily mortality and daily solar intensity.

2.1 Introduction

Forecasts are often contingent on a long history of predictors, which are non-linearly related to the variable of interest. For example, when forecasting half-hourly electricity demand, it is common to use at least a week of historical half-hourly temperatures and other weather observations (Hyndman & Fan 2010). The relationships between lagged temperatures and electricity demand are non-linear (due to both heating and cooling effects) and involve complex interactions due to thermal inertia in buildings (Fan & Hyndman 2012). Similarly, when forecasting bore levels, rainfall data from up to a thousand days earlier can impact the result (Peterson & Western 2014; Bakker & Schaars

2019; Rajaee, Ebrahimi & Nourani 2019) due to the complex non-linear flow dynamics of rainfall in aquifers.

These examples suggest a possible non-linear ***transfer function model*** of the form

$$y_t = f(\boldsymbol{a}_t, \boldsymbol{a}_{t-1}, \ldots, \boldsymbol{a}_{t-s}, y_{t-1}, \ldots, y_{t-k}) + \varepsilon_t,$$

where y_t is the observation of the response variable at time t, $\boldsymbol{a}_t$ is a vector of predictors at time t, and ε_t is an i.i.d. random error. By including lagged values of y_t along with the lagged predictors, the model can capture the serial correlation present in the data.

The form of f is typically non-linear and involves complicated interactions with a high value of s (and possibly a large k). It is infeasible to estimate f in high-dimensional settings (where s and k are large) due to the curse of dimensionality (Bellman 1957; Stone 1982). Therefore, we normally impose some additivity constraint and ignore interactions of more than two or three variables. There are also numerous ad hoc model choices in selecting the appropriate predictors to include.

For example, Fan & Hyndman (2012) proposed a *semiparametric additive model* to obtain short-term forecasts of half-hourly electricity demand for power systems in the Australian National Electricity Market. In this model, f is assumed to be fully additive and is used to capture the effects of recent predictor values on the demand. The main objective is to allow nonparametric components in a regression-based modelling framework, particularly to address serially correlated errors. The model fitted for each half-hourly period (q) can be written as

$$\log(y_{t,q}) = h_q(t) + f_q(\boldsymbol{w}_{1,t}, \boldsymbol{w}_{2,t}) + \sum_{j=1}^{k} a_{q,j}(y_{t-j}) + \varepsilon_t.$$

The term $h_q(t)$ models several calendar effects as either linear or smooth terms. Temperature effects are modelled using the nonparametric component $f_q(\boldsymbol{w}_{1,t}, \boldsymbol{w}_{2,t})$, where $\boldsymbol{w}_{i,t} = (w_{i,t}, \ldots, w_{i,t-s})^T$ is a vector of lagged temperatures at site i. The terms $a_{q,j}(y_{t-j})$ capture the lagged effects of the response. Note that the error term ε_t is serially uncorrelated within each half-hourly model because the serial correlation is eliminated by the inclusion of lagged responses in the model. However, some correlation may still exist between residuals from various half-hourly models (Fan & Hyndman 2012).

Similarly, a *distributed lag model* was proposed by Wood (2017) to forecast daily death rates in Chicago using measurements of several air pollutants. The response variable is modelled as a sum of smooth functions of lagged predictor variables, similar to the semiparametric additive model used by Fan & Hyndman (2012). However, unlike Fan & Hyndman (2012), Wood (2017) suggested allowing the smooth functions for lags of the same covariate to vary smoothly over lags, preventing large

differences in estimated effects between adjacent lags. Thus, the model is of the form

$$\log(y_t) = f_1(t) + \sum_{s=0}^{S} f_2(p_{t-s}, s) + \sum_{s=0}^{S} f_3(o_{t-s}, w_{t-s}, s),$$

where y_t is the death rate on day t, f_1 is a nonparametric term capturing the time effect, and p_t, o_t, and w_t are various predictor variables. The model incorporates the current value ($s = 0$) and several lagged values ($s = 1, \ldots, S$) of the predictors. The distributed lag effect of a single predictor variable and the interaction of two predictor variables are captured by the sums $\sum_{s=0}^{S} f_2(p_{t-s}, s)$ and $\sum_{s=0}^{S} f_3(o_{t-s}, w_{t-s}, s)$, respectively.

Further examples include Du, Cheng & Liang (2012), Goude, Nedellec & Kong (2014), Ho, Chen & Hwang (2020), and Boente & Martínez (2023), who applied semiparametric additive models to predict household gasoline consumption, local short and medium term electricity demand, ground-level $PM_{2.5}$ concentrations, and housing prices, respectively. Ibrahim et al. (2023) considered nonparametric additive models to predict census survey response rates. Additionally, Ravindra et al. (2019) provided a comprehensive review of the applications of additive models in environmental data, with a special focus on studies related to air pollution, climate change, and human health.

While nonparametric and semiparametric additive models have been used in diverse applications, several unresolved issues remain in their implementation. First, estimating these models in high-dimensional settings is challenging due to the large number of nonparametric components that need to be estimated. Second, there is noticeable subjectivity in selecting predictor variables (from the available predictors) and identifying which terms should be grouped to model interactions. In most of the applications discussed above, the choices are based on empirical exploration or domain expertise.

In this chapter, we focus on high-dimensional applications that exhibit complex interactions among predictors, particularly in the presence of a large number of lagged variables. In such situations, *index models* are beneficial in enhancing the flexibility of the broader class of nonparametric additive models (Radchenko 2015), while mitigating the difficulty of estimating a nonparametric component for each individual predictor.

Therefore, we propose addressing the above-mentioned issues using a ***Sparse Multiple Index (SMI) model*** with automatic variable selection and grouping. This semiparametric model can be written as

$$y_i = \beta_0 + \sum_{j=1}^{q} g_j(\boldsymbol{\alpha}_j^T \boldsymbol{x}_{ij}) + \sum_{k=1}^{d} f_k(w_{ik}) + \boldsymbol{\theta}^T \boldsymbol{u}_i + \varepsilon_i, \quad i = 1, \ldots, n, \tag{2.1}$$

where y_i is the univariate response, β_0 is the model intercept, $\boldsymbol{x}_{ij} \in \mathbb{R}^{l_j}$, $j = 1, \ldots, q$ are q subsets of predictors entering indices, and $\boldsymbol{\alpha}_j$ is a vector of index coefficients corresponding to the index $h_{ij} = \boldsymbol{\alpha}_j^T \boldsymbol{x}_{ij}$. The function g_j is a smooth non-linear function, possibly estimated by a spline. The model also allows for predictors that do not enter any indices, including covariates w_{ik}, $k = 1, \ldots, d$, which relate to the response through non-linear functions f_k, $k = 1, \ldots, d$, and linear covariates $\boldsymbol{u}_i$. The error term ε_i may exhibit autocorrelation, as the proposed SMI model is a simplified alternative to the previously discussed transfer function model and may not adequately capture the full autocorrelation structure inherent in the data. Although our focus is on forecasting time series data, the model has broader applicability; therefore, we have not included any notation specific to time series in the model formulation.

The SMI model encompasses the previously discussed models, including fully additive models (Wood 2011, 2017), where each predictor has its own index, and single index models (Stoker 1986; Härdle, Hall & Ichimura 1993; Radchenko 2015), where all predictors are included in a single index. This greater generality allows us to address the two issues mentioned earlier. First, the number of parameters to be estimated is reduced by combining variables using linear indices (i.e. grouping predictors into indices) to manage the complexity of interactions. In our model formulation, the number of indices q and the grouping of predictors within indices are unknown. We propose an algorithmic selection process to determine which predictors should be included in each index, thereby reducing subjectivity in model formulation. Here, we assume that no predictor enters more than one index, i.e. overlapping of predictors among indices is not allowed.

To our knowledge, no previous research has explored objective and principled predictor selection in nonparametric additive index models. Hence, our goal is to develop a methodology for optimal predictor selection and grouping in the context of high-dimensional nonparametric additive index models. Recent computational advancements have sparked significant interest in using mathematical optimisation to solve statistical problems (Theußl, Schwendinger & Hornik 2020). This has motivated us to develop a variable selection algorithm based on mathematical optimisation techniques.

It is crucial to note that any variable selection methodology naturally renders inferential statistics invalid, as we do not assume that the resulting model accurately represents the true Data Generating Process (DGP). Therefore, our objective is solely to improve forecast accuracy, rather than to draw inferences about parameter estimates.

The rest of this chapter is organised as follows. Section 2.2 presents the proposed Sparse Multiple Index Model and describes the algorithm for variable selection and grouping, as well as the estimation procedure. In Section 2.3, the functionality and characteristics of the proposed algorithm are demon-

strated through a simulation experiment. Section 2.4 illustrates two empirical applications of the proposed estimation and variable selection methodology, related to forecasting heat exposure-related daily mortality and daily solar intensity. Some benchmark comparison methods are also briefly introduced in this section. Concluding remarks are given in Section 2.5.

2.2 Sparse multiple index model

2.2.1 Optimisation problem formulation

We implement variable selection for the proposed ***Sparse Multiple Index (SMI) model*** in Equation 2.1 by allowing for zero index coefficients for predictors. Suppose we observe $y_1, \ldots, y_n$, along with a set of potential predictor vectors, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$, where each vector $\boldsymbol{x}_i$, $i = 1, \ldots, n$, contains p predictors that may enter indices. The optimisation problem we seek to address is of the form below, where the sum of the squared errors of the model (Equation 2.1) is minimised together with an ℓ_0 penalty term and an ℓ_2 (ridge) penalty term:

$$\min_{\beta_0, q, \boldsymbol{\alpha}, \boldsymbol{g}, \boldsymbol{f}, \boldsymbol{\theta}} \quad \sum_{i=1}^{n} \left[y_i - \beta_0 - \sum_{j=1}^{q} g_j(\boldsymbol{\alpha}_j^T \boldsymbol{x}_i) - \sum_{k=1}^{d} f_k(w_{ik}) - \boldsymbol{\theta}^T \boldsymbol{u}_i \right]^2 + \lambda_0 \sum_{j=1}^{q} \sum_{m=1}^{p} \mathbb{1}(\alpha_{jm} \neq 0) + \lambda_2 \sum_{j=1}^{q} \|\boldsymbol{\alpha}_j\|_2^2 \qquad (2.2)$$

$$\text{such that} \quad \sum_{j=1}^{q} \mathbb{1}(\alpha_{jm} \neq 0) \in \{0, 1\} \quad \forall m,$$

where $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1^T, \ldots, \boldsymbol{\alpha}_q^T]^T$, $\boldsymbol{g} = \{g_1, g_2, \ldots, g_q\}$, $\boldsymbol{f} = \{f_1, f_2, \ldots, f_d\}$, $\mathbb{1}(\cdot)$ is the indicator function, $\lambda_0 \geq 0$ is a tuning parameter that controls the number of selected predictors entering indices, and $\lambda_2 \geq 0$ is another tuning parameter that controls the strength of the additional shrinkage imposed on the estimated index coefficients. The constraint ensures that every predictor can only have a non-zero coefficient in at most one index.

Applying an ℓ_2 penalty in addition to the ℓ_0 penalty is motivated by related literature (Hazimeh & Mazumder 2020; Mazumder, Radchenko & Dedieu 2023; Hazimeh, Mazumder & Radchenko 2023), where it is suggested that the prediction performance of best subset selection can be enhanced by the inclusion of an additional ridge penalty, especially when there is a low signal-to-noise ratio.

Here, we use a combination of ℓ_0 and ℓ_2 penalties for variable selection rather than an ℓ_1 (LASSO) penalty, which can have a similar effect. The ℓ_0 penalty directly enforces subset selection by forcing some coefficients to be exactly zero, while the ℓ_2 penalty shrinks the remaining coefficients without forcing them to zero. Thus, the $\ell_0 + \ell_2$ combination gives us more control over the trade-off between

selecting important features (ℓ_0) and preventing over-fitting by controlling the magnitude of coefficients (ℓ_2), which is more suitable for our research goals. In contrast, the ℓ_1 penalty combines both sparsity and shrinkage in a single term, but it may not offer the same degree of flexibility in managing these two aspects independently.

To solve the optimisation problem in Equation 2.2, we present a big-M based *Mixed Integer Quadratic Programming (MIQP)* formulation:

$$\begin{aligned}
\min_{\beta_0,q,\boldsymbol{\alpha},\boldsymbol{g},\boldsymbol{f},\boldsymbol{\theta},\boldsymbol{z}} \quad & \sum_{i=1}^{n}\left[y_i-\beta_0-\sum_{j=1}^{q}g_j(\boldsymbol{\alpha}_j^T\boldsymbol{x}_i)-\sum_{k=1}^{d}f_k(w_{ik})-\boldsymbol{\theta}^T\boldsymbol{u}_i\right]^2+\lambda_0\sum_{j=1}^{q}\sum_{m=1}^{p}z_{jm}+\lambda_2\sum_{j=1}^{q}\sum_{m=1}^{p}\alpha_{jm}^2 \\
\text{s.t.} \quad & |\alpha_{jm}|\le Mz_{jm} \quad \forall j,\forall m, \\
& \sum_{j=1}^{q}z_{jm}\le 1 \quad \forall m, \\
& z_{jm}\in\{0,1\},
\end{aligned} \tag{2.3}$$

where $j=1,\dots,q$, and $m=1,\dots,p$. Here, we have introduced binary variables $z_{jm}=\mathbb{1}(\alpha_{jm}\neq 0)$ to indicate the index (if any) into which each predictor is selected. The ℓ_0 penalty is now applied to these binary variables, which encourages sparsity by driving some of them to zero, thereby performing variable selection. The big-M constraint ensures that α_{jm} is zero if and only if z_{jm} is zero, and if $z_{jm}=1$, then $|\alpha_{jm}|\le M$. This constraint enforces the relationship between the binary selection variables and the continuous model coefficients, reflecting the variable selection induced by the ℓ_0 penalty in the estimated model. Here, $M<\infty$ denotes the pre-specified *big-M parameter*, which should be sufficiently large. If $\boldsymbol{\alpha}^*$ is the optimal solution to the problem defined in Equation 2.3, then the big-M parameter should satisfy $\max\left(\{|\alpha_{jm}^*|\}_{j\in[q],m\in[p]}\right)\le M$. Finally, while the ℓ_0 penalty performs direct subset selection, the ℓ_2 penalty term provides additional shrinkage of the estimated coefficients, helping to prevent over-fitting by controlling their magnitudes, as previously discussed.

2.2.2 Estimation algorithm

We now show how to efficiently find a minimiser for the problem given in Equation 2.3. Since the number of indices q, the vector of index coefficients $\boldsymbol{\alpha}$, and the set of nonparametric functions $\boldsymbol{g}$ are all unknown, it is impossible to solve the above MIQP given in Equation 2.3 directly. Hence, we propose an iterative algorithm to solve the problem.

Initialising index structure and index coefficients

We first need to provide a feasible initialisation for the index structure (i.e. the number of indices q and the grouping of predictors among indices), as well as for the index coefficients (i.e. $\boldsymbol{\alpha}$) of the model.

Based on several experiments, we propose four possible methods for initialising the SMI model as follows.

1. **PPR: Projection pursuit regression based initialisation**

   A Projection Pursuit Regression model (Friedman & Stuetzle 1981) is a multiple index model, where each index includes all the available predictors. Since the proposed SMI model requires that there are no overlapping indices, it is impossible to use an estimated PPR model directly as a starting model for the algorithm. Thus, we follow the steps presented below to come up with a feasible initialisation for the index structure and the index coefficients.

   a. Scale the predictors in the data set that are entering indices by dividing each variable by its standard deviation (so that it is possible to compare the estimated coefficients among predictors).
   b. Fit a PPR model using the scaled predictors entering indices and obtain the estimated index coefficients. (The user can decide the number of initial indices q^* to be estimated; we use $q^* = 5$ in our simulations and applications.)
   c. Calculate a threshold $\tau = 0.1 \times \max(\text{abs}(\text{PPR index coefficients}))$.
   d. Set all coefficients with absolute values below the calculated threshold to zero.
   e. For predictors appearing in multiple indices, assign them to the index with the maximum absolute coefficient and set their coefficients in other indices to zero.
   f. After performing the above steps a–e, if any originally estimated index has all zero coefficients, it will be excluded from the model.
   g. Back-transform each index coefficient to the original scale of the respective predictor variable, reversing the scaling effect applied at the beginning.

   Now, the index structure and the index coefficients obtained through the above steps are considered to be a feasible initialisation for the proposed algorithm.

2. **Additive: Nonparametric additive model based initialisation**

   As a fully additive model is a special case of the SMI model, we can set $q = p$ and assign each predictor to its own index.

3. **Linear: Linear regression based initialisation**

We first regress the response variable on the predictors using multiple linear regression. Then, we construct a single index (i.e. $q = 1$) using the estimated regression coefficients as the initial index coefficients of the predictors.

4. **Multiple: Picking one from multiple initialisations**

   We use multiple random instances of a model (i.e. the grouping of predictors into indices and the initial index coefficients are generated randomly), with a user-specified number of indices, as initial models. We then optimise the SMI model for each instance and select the initial model that results in the lowest loss for the MIQP problem. The number of random instances of the model to be evaluated is also specified by the user.

Of course, it is also possible for a user to specify an initialisation based on their domain expertise or prior knowledge when initialising the algorithm. For each of the above initialisation options, once the estimate for $\boldsymbol{\alpha}$ is obtained, the estimated initial index coefficients for each index are scaled to have a unit norm to ensure identifiability.

Estimating non-linear functions

Once we have an estimate for $\boldsymbol{\alpha}$, estimating the SMI model is equivalent to estimating a Generalised Additive Model (GAM) given by

$$y_i = \beta_0 + \sum_{j=1}^{q} g_j(\hat{h}_{ij}) + \sum_{k=1}^{d} f_k(w_{ik}) + \boldsymbol{\theta}^T \boldsymbol{u}_i + \varepsilon_i, \quad i = 1, \ldots, n,$$

where y_i is the response, and $\hat{h}_{ij} = \hat{\boldsymbol{\alpha}}_j^T \boldsymbol{x}_i$ is the estimated index. The R packages `mgcv` (Wood 2023, 2011) and `gam` (Hastie 2024), for example, can be used to fit GAMs. In all experiments presented in this chapter, we estimate the sets of nonparametric functions $\boldsymbol{g}$ and $\boldsymbol{f}$ using penalised cubic regression splines through the `mgcv` package.

Updating index structure and index coefficients

We obtain the updated index coefficients $\boldsymbol{\alpha}^{\text{new}}$ through an MIQP:

$$\begin{aligned}
&\min_{\boldsymbol{\alpha}^{\text{new}}, \boldsymbol{z}^{\text{new}}} (\boldsymbol{\alpha}^{\text{new}} - \boldsymbol{\alpha}^{\text{old}})^T \boldsymbol{V}^T \boldsymbol{V} (\boldsymbol{\alpha}^{\text{new}} - \boldsymbol{\alpha}^{\text{old}}) - 2(\boldsymbol{\alpha}^{\text{new}} - \boldsymbol{\alpha}^{\text{old}})^T \boldsymbol{V}^T \boldsymbol{r} + \lambda_0 \sum_{j=1}^{q} \sum_{m=1}^{p} z_{jm}^{\text{new}} + \lambda_2 \sum_{j=1}^{q} \sum_{m=1}^{p} (\alpha_{jm}^{new})^2 \\
&\text{s.t. } |\alpha_{jm}^{\text{new}}| \leq M z_{jm}^{\text{new}} \quad \forall j, \forall m, \\
&\quad z_{jm}^{\text{new}} \in \{0, 1\}, \\
&\quad \sum_{j=1}^{q} z_{jm}^{\text{new}} \leq 1 \quad \forall m,
\end{aligned} \tag{2.4}$$

where $j = 1, \ldots, q$, $m = 1, \ldots, p$, $\boldsymbol{\alpha}^{\text{old}}$ is the current value of $\boldsymbol{\alpha}$, z_{jm}^{new} are the updated set of binary variables to be estimated, and $\boldsymbol{V}$ is the matrix of partial derivatives of the right-hand side of Equation 2.1 with respect to $\boldsymbol{\alpha}_j$. The i^{th} row of $\boldsymbol{V}$ contains $[\boldsymbol{v}_{i1}, \ldots, \boldsymbol{v}_{iq}]$, where $\boldsymbol{v}_{ij} = \boldsymbol{x}_i g_j'(h_{ij})$. The current residual vector, which contains $r_i = y_i - \beta_0 - \sum_{j=1}^{q} g_j\big((\boldsymbol{\alpha}_j^{\text{old}})^T \boldsymbol{x}_i\big)$, is denoted by $\boldsymbol{r}$. It is important to note that the additional covariates w_{ik} and $\boldsymbol{u}_i$ are not required to update $\boldsymbol{\alpha}_j$, because they are constants with respect to $\boldsymbol{\alpha}_j$, and thus they disappear from $\boldsymbol{V}$.

Similar to the explanation given by Masselot et al. (2023), the MIQP objective function in Equation 2.4 ignores the Hessian (the matrix of second derivatives of Equation 2.1 with respect to $\boldsymbol{\alpha}_j$), and considers only the matrix of first derivatives, which is a quasi-Newton step (Peng 2022). Therefore, the $\boldsymbol{\alpha}$ updating step given in Equation 2.4 is assured to be in a descent direction.

After obtaining $\boldsymbol{\alpha}^{\text{new}}$, any estimated individual index coefficient vectors $\boldsymbol{\alpha}_j^{\text{new}}$ that contain only zeros will be dropped from the model. Then, we scale each estimated index coefficient vector $\hat{\boldsymbol{\alpha}}_j = \boldsymbol{\alpha}_j^{\text{new}}$ to have a unit norm.

Next, the algorithm alternates between updating the index coefficients $\boldsymbol{\alpha}$ and estimating the non-linear functions $\boldsymbol{g}$ until it meets one of the three criteria: (i) the reduction ratio of the objective (loss) function value in Equation 2.3, calculated between consecutive iterations, reaches a pre-specified convergence tolerance; (ii) the loss increases consecutively for three iterations; or (iii) a pre-specified maximum number of iterations is reached. The selection of convergence tolerance and maximum iterations depends on the specific problem or data.

Finally, we consider adjusting the model's index structure to exploit any potential benefit in further reducing the loss function in Equation 2.3. Since indices can be automatically reduced by dropping all-zero indices at each optimisation iteration, this step focuses on the possible addition of new indices to the current model. Specifically, we examine adding a new index by identifying predictors that have been dropped. If applicable, a new index is constructed from all such dropped predictors, and the alternating update process described previously is repeated. This step continues until one of the following termination criteria is met: (i) the number of indices reaches p, in which case the current model is selected as the final SMI model; (ii) the loss increases after the addition, whereby the model from the previous iteration is chosen as the final SMI model; or (iii) the number of indices remains unchanged from the previous iteration and the absolute difference in index coefficients between two successive iterations does not exceed a pre-specified tolerance, with the model exhibiting the smaller loss selected as the final SMI model.

The following algorithm briefly summarises the key steps of the SMI Modelling algorithm.

Algorithm 1: SMI modelling algorithm

1. Initialise q, the predictor grouping among indices, and obtain the initial estimate of $\boldsymbol{\alpha}$ using one of the options given in Section 2.2.2. Then, scale each $\hat{\boldsymbol{\alpha}}_j$, $j = 1, \ldots, q$, to have a unit norm.
2. Estimate g_j, $j = 1, \ldots, q$, using a GAM, taking y_i as the response and $\hat{\boldsymbol{\alpha}}_j^T \boldsymbol{x}_i$ as predictors.
3. Update $\boldsymbol{\alpha}$ through the MIQP in Equation 2.4, and again scale each $\hat{\boldsymbol{\alpha}}_j$ to have a unit norm.
4. Iterate steps 2 and 3 until convergence, loss increases for three consecutive iterations, or the maximum number of iterations is reached.
5. If there are no dropped predictors, stop the algorithm. Otherwise, include a new index consisting of all dropped predictors, and repeat step 4.
6. Increase q by one in each iteration of step 5 until meeting one of the termination criteria below:
   - The number of indices in the model reaches p; select the final fitted model as the final SMI model.
   - Loss increases after the increment; select the previous iteration model as the final SMI model.
   - The solution maintains the same number of indices as the previous iteration, and the absolute difference of index coefficients between two successive iterations is not larger than a pre-specified tolerance; select the model with the smaller loss as the final SMI model.

Tuning penalty parameters

To obtain an estimated model with the best possible forecasting accuracy, it is important to choose appropriate values for the penalty parameters λ_0 and λ_2. We use a greedy search procedure over a grid of possible values for λ_0 and λ_2, selecting the combination of penalty parameters that yields the lowest Mean Squared Error (MSE) on a validation data set that does not overlap with the data set used for model training. Here, a greedy search is used instead of a full grid search to reduce the computational cost.

The range of possible values for the two penalty parameters is determined by the scale of the first term of the SMI model objective function in Equation 2.3. We fit a GAM and calculate the first term of the SMI model objective function using it as a benchmark model, which is taken as the maximum value for λ_0 (λ_0`.max`). A pseudo minimum for λ_0 is then calculated as a user-specified ratio (λ_0`.min.ratio`) of λ_0`.max`. For all the experiments in this chapter, we have used λ_0`.min.ratio` $= 0.0001$. Once the maximum and pseudo minimum values have been calculated, a sequence of possible values for λ_0 (λ_0`.seq`) is generated to make them decrease on a logarithmic scale, where the length of the sequence is user-specified; the default being 100. Once the sequence is generated, the value zero is included in the sequence as the minimum value for λ_0 (λ_0`.min`). The maximum for λ_2 (λ_2`.max`)

is taken to be the power of ten that matches the scale of λ_0`.max`, and the pseudo minimum for λ_2 is taken to be 0.01. Then a sequence of powers of ten is constructed as the possible values for λ_2 (λ_2`.seq`). Next, similar to the λ_0`.seq`, zero is included in λ_2`.seq` to be the minimum for λ_2 (λ_2`.min`). The grid of values on which the greedy search is performed is therefore constructed as the Cartesian product λ_0`.seq` $\times$ λ_2`.seq`.

To perform a greedy search on the grid, it is necessary to determine a starting point. We choose the starting point by selecting the point that gives the lowest validation set MSE of the following points—the four corners of the grid, the midpoints of the four edges of the grid, the midpoint of the grid, and the points that are one grid distance apart around each of the above nine points. Once the starting point is chosen, we perform the first greedy search by iteratively searching the points at a grid distance around the point chosen in the previous iteration. This first round of greedy search continues until the lowest validation set MSE of the previous iteration is less than the lowest validation set MSE of the current iteration.

If the best point selected from the first round of the greedy search happens to lie on the top or right edge of the grid, the corresponding penalty parameter values are modified by plus or minus ten per cent, and a second round of greedy search is performed, going beyond the original grid if necessary. If the selected value for a penalty parameter is zero, no adjustment is made.

Finally, the validation set MSE of the SMI model, estimated with the chosen best penalty parameter combination, is compared with the validation set MSE of the no-index model (a model estimated with additional non-linear (non-index) and linear predictors, if such predictors are available) and the null model (intercept-only model). If the validation set MSE of the SMI model, estimated with the chosen best penalty parameter combination, is lower than the no-index and null models, we choose it as the final model. If the no-index or null model has the lowest validation set MSE, then we choose the respective model as the final model.

The following algorithm summarises the greedy search procedure for tuning penalty parameters.

Algorithm 2: Greedy search algorithm for tuning penalty parameters

1. Fit a GAM as a benchmark model to the training data.
2. Determine the greedy search range.
3. Select the starting point as the point that gives the lowest validation set MSE from the following points: the four corners of the grid, the midpoints of the four edges of the grid, the midpoint of the grid, and the points that are one grid distance apart around each of the above nine points.

4. Perform the first round of greedy search by iteratively searching the points that are one grid distance apart around the point selected in the previous iteration, until the lowest validation set MSE of the previous iteration is less than the lowest validation set MSE of the current iteration.
5. If the best point is at the top or right edge of the grid, modify the parameters by plus or minus ten per cent and perform a second round of greedy search. If the selected value for a penalty parameter is zero, no adjustment is made.
6. Compare the validation set MSE of the estimated SMI model with the validation set MSE of the no-index model (if applicable) and the null model, and select the model with the lowest validation set MSE as the final model.

Other parameter choices

Moreover, it is also crucial to choose a suitable value for the big-M parameter, as the strength of the MIP formulation depends on the choice of a good upper bound (Bertsimas, King & Mazumder 2016). According to Hazimeh, Mazumder & Radchenko (2023), several methods have been used to select M in practice. For more details on estimating M in a linear regression setting, refer to Bertsimas, King & Mazumder (2016).

Throughout the experiments in this chapter, we use $M = 10$, a convergence tolerance of 0.001, a maximum of 50 iterations in step 4 of Algorithm 1, and a convergence tolerance of 0.001 for coefficients in step 6 of Algorithm 1.

Forecasting with SMI models

As previously mentioned, the sets of nonparametric functions $\boldsymbol{g}$ and $\boldsymbol{f}$ in the SMI model are estimated using penalised cubic regression splines within a GAM. One limitation of using splines is that they are typically ill-suited for extrapolation. Predictions made outside the range of the training data are likely to be unreliable and should be treated with caution, as splines are designed to capture patterns within the observed data. Outside this range, spline estimates often exhibit unpredictable behaviour (Wood 2017).

However, it is worth noting that natural cubic splines—which are cubic splines with an additional constraint of linearity at the boundaries (i.e. the second derivative is zero at the boundary knots) (Wood 2017; Ruppert, Wand & Carroll 2003)—are somewhat more robust to extrapolation. However, this approach results in linear extrapolations, which may still inaccurately represent the behaviour of the nonparametric function of interest, particularly when the focus is on variables that exhibit non-linear behaviour.

Therefore, when generating forecasts with SMI models for a test set that was not used during model training, we truncate the values of all predictor variables treated non-linearly (i.e. those entering

the indices, as well as those entering the model non-linearly but not via the indices) to fall within the corresponding in-sample data ranges. Specifically, if a test set value of a non-linearly treated predictor falls below the in-sample minimum, it is set to that minimum; if it exceeds the in-sample maximum, it is set to that maximum, before producing forecasts.

2.3 Simulation experiment

This section presents the results of a simulation experiment designed to demonstrate the performance and characteristics of the proposed SMI Modelling algorithm. In particular, we investigate how the estimated SMI model varies depending on the initialisation option used.

2.3.1 Data generation

Generating predictor variables

First, we generate two time series, each of length 1205: $x_{t,0}$ from a uniform distribution on the interval $[0, 1]$, and $z_{t,0}$ from a normal distribution $N(5, 4)$. Next, we construct lagged series of both $x_{t,0}$ and $z_{t,0}$, where $x_{t,i}$ denotes the i^{th} lag of $x_{t,0}$, and $z_{t,i}$ denotes the i^{th} lag of $z_{t,0}$. Then $\boldsymbol{x}_t = \{x_{t,0}, x_{t,1}, \ldots, x_{t,5}\}$, and $\boldsymbol{z}_t = \{z_{t,0}, z_{t,1}, \ldots, z_{t,5}\}$ are taken as predictors in the simulation experiment.

Here, the choice of i.i.d. variables as predictors is a deliberate simplification in the simulation setup, aimed at testing the predictor selection and grouping performance of the proposed SMI modelling algorithm. This avoids additional complications arising from autocorrelation or cross-correlation among predictors. While this setup does not fully reflect real-world time series dependencies, it serves as a useful baseline for investigating the algorithm's behaviour. Extending the study to include time-dependent variables would be valuable in future research.

Generating response variables

We generate two response variables y_t and w_t, using two different index structures and a normally distributed white noise component with two different variances, as follows:

- Low noise level - $N(\mu = 0, \sigma^2 = 0.01)$:
  $y_t = (0.9 * x_{t,0} + 0.6 * x_{t,1} + 0.45 * x_{t,3})^3 + \epsilon, \quad \epsilon \sim N(0, 0.01)$
  $w_t = (0.9 * x_{t,0} + 0.6 * x_{t,1} + 0.45 * x_{t,3})^3 + (0.35 * x_{t,2} + 0.7 * x_{t,5})^2 + \epsilon, \quad \epsilon \sim N(0, 0.01)$
- High noise level - $N(\mu = 0, \sigma^2 = 0.25)$:
  $y_t = (0.9 * x_{t,0} + 0.6 * x_{t,1} + 0.45 * x_{t,3})^3 + \epsilon, \quad \epsilon \sim N(0, 0.25)$
  $w_t = (0.9 * x_{t,0} + 0.6 * x_{t,1} + 0.45 * x_{t,3})^3 + (0.35 * x_{t,2} + 0.7 * x_{t,5})^2 + \epsilon, \quad \epsilon \sim N(0, 0.25)$

The response y_t is constructed using a single index consisting of the predictor variables $x_{t,0}, x_{t,1}$, and $x_{t,3}$. The other response w_t is constructed using two indices, where the first index consists of the predictors $x_{t,0}, x_{t,1}$, and $x_{t,3}$, and the second index consists of $x_{t,2}$ and $x_{t,5}$. Neither the variable $x_{t,4}$ nor any of the z variables are used in generating y_t and w_t.

Once the data set is generated, the first five observations are discarded due to the missing values introduced by lagged variables, leaving a data set with 1200 observations. We use the first 1000 observations as the training set, while the remaining 200 observations are kept aside as the test set to evaluate the estimated models.

2.3.2 Experiment setup

We estimate SMI models through the proposed algorithm for each of the two response variables, using three different sets of predictors as inputs. Our aim is to assess the algorithm's capability to correctly select the relevant predictor variables (and drop the irrelevant predictors) and to estimate the index structure of the true model.

The three different sets of predictors considered are as follows:

1. All $\boldsymbol{x}$ variables (denoted as "all $\boldsymbol{x}$");
2. All $\boldsymbol{x}$ variables and all $\boldsymbol{z}$ variables (denoted as "all $\boldsymbol{x}$ + all $\boldsymbol{z}$");
3. The first three $\boldsymbol{x}$ variables (i.e. $x_{t,0}, x_{t,1}$ and $x_{t,2}$) and all $\boldsymbol{z}$ variables (denoted as "some $\boldsymbol{x}$ + all $\boldsymbol{z}$").

We apply the proposed SMI Modelling algorithm with each of the above predictor combinations for both variations of the responses concerning the noise level. Moreover, we consider all the initialisation options discussed in Section 2.2.2 for each of the two responses.

2.3.3 Results

We summarise the results of the simulation experiment in Table 2.1. In the columns, we indicate the index structure (i.e. the predictor grouping among indices) estimated by the proposed algorithm under each of the initialisation options, i.e. "PPR", "Additive", "Linear", and "Multiple". The numbers stated (in parentheses) below the index structures are the test MSE values of the corresponding estimated models. This is detailed for each combination explored, considering the response, input predictors, and noise levels.

We did not perform any tuning for the penalty parameters λ_0 and λ_2 in the simulation experiment. The default values $\lambda_0 = 1$ and $\lambda_2 = 1$ were used in estimating all the models presented in Table 2.1. (However, our experiments indicated that using $\lambda_0, \lambda_2 \ll 1$ might result in estimated models with incorrect index structures and irrelevant variables in this example.)

Table 2.1: *Simulation experiment results showing the index structures and test MSE values (in parentheses below the index structures) of the estimated SMI models with different initialisations for each combination of response, input predictors, and noise levels.*

Model	Predictors	PPR	Additive	Linear	Multiple
Low noise level					
y_t	all $\boldsymbol{x}$	$(x_{t,0}, x_{t,1}, x_{t,3})$ (0.009)	$(x_{t,0}, x_{t,1}, x_{t,3})$ (0.009)	$(x_{t,0}, x_{t,1}, x_{t,3})$ (0.009)	$(x_{t,0}, x_{t,1}, x_{t,3})$ (0.009)
y_t	all $\boldsymbol{x}$ + all $\boldsymbol{z}$	$(x_{t,0}, x_{t,1}, x_{t,3})$ (0.009)	$(x_{t,0}, x_{t,1}, x_{t,3})$ (0.009)	$(x_{t,0}, x_{t,1}, x_{t,3})$ (0.009)	$(x_{t,0}, x_{t,1}, x_{t,3})$ (0.009)
y_t	some $\boldsymbol{x}$ + all $\boldsymbol{z}$	$(x_{t,0}, x_{t,1}, z_{t,2}, z_{t,4})$ (0.233)	$(x_{t,0}, x_{t,1})$ $(z_{t,4})$ $(z_{t,1})$ (0.235)	$(x_{t,0}, x_{t,1}, z_{t,2}, z_{t,4})$ (0.233)	$(x_{t,0}, x_{t,1}, z_{t,2}, z_{t,4})$ (0.233)
w_t	all $\boldsymbol{x}$	$(x_{t,0}, x_{t,1}, x_{t,3})$ $(x_{t,2}, x_{t,5})$ (0.157)	$(x_{t,0}, x_{t,1}, x_{t,3})$ $(x_{t,2}, x_{t,5})$ (0.157)	$(x_{t,0}, x_{t,1}, x_{t,2}, x_{t,3}, x_{t,5})$ (0.158)	$(x_{t,0}, x_{t,1}, x_{t,3})$ $(x_{t,2}, x_{t,5})$ (0.157)
w_t	all $\boldsymbol{x}$ + all $\boldsymbol{z}$	$(x_{t,0}, x_{t,1}, x_{t,3})$ $(x_{t,2}, x_{t,5})$ (0.157)	$(x_{t,0}, x_{t,1}, x_{t,3})$ $(x_{t,2}, x_{t,5})$ (0.157)	$(x_{t,0}, x_{t,1}, x_{t,2}, x_{t,3}, x_{t,5})$ (0.158)	$(x_{t,0}, x_{t,1}, x_{t,3})$ $(x_{t,2}, x_{t,5})$ (0.157)
w_t	some $\boldsymbol{x}$ + all $\boldsymbol{z}$	$(x_{t,0}, x_{t,1})$ $(x_{t,2})$ $(z_{t,4})$ (0.340)	$(x_{t,0}, x_{t,1}, z_{t,4})$ $(x_{t,2})$ (0.340)	$(x_{t,0}, x_{t,1}, x_{t,2}, z_{t,2})$ (0.336)	$(x_{t,0}, x_{t,1})$ $(x_{t,2}, z_{t,2}, z_{t,3})$ (0.345)
High noise level					
y_t	all $\boldsymbol{x}$	$(x_{t,0}, x_{t,1}, x_{t,3})$ $(x_{t,2}, x_{t,4}, x_{t,5})$ (0.235)	$(x_{t,0}, x_{t,1})$ $(x_{t,3})$ (0.274)	$(x_{t,0}, x_{t,1}, x_{t,3})$ (0.233)	$(x_{t,0}, x_{t,1}, x_{t,3})$ (0.233)
y_t	all $\boldsymbol{x}$ + all $\boldsymbol{z}$	$(x_{t,0}, x_{t,1}, x_{t,3})$ (0.233)	$(x_{t,0}, x_{t,1}, x_{t,3})$ $(z_{t,0})$ (0.232)	$(x_{t,0}, x_{t,1}, x_{t,3})$ (0.233)	$(x_{t,0}, x_{t,1}, x_{t,3})$ $(z_{t,0})$ (0.232)
y_t	some $\boldsymbol{x}$ + all $\boldsymbol{z}$	$(x_{t,0}, x_{t,1})$ $(z_{t,1})$ $(z_{t,4})$ (0.476)	$(x_{t,0}, x_{t,1})$ $(z_{t,1})$ $(z_{t,4})$ (0.476)	$(x_{t,0}, x_{t,1}, z_{t,2}, z_{t,4})$ (0.474)	$(x_{t,0}, x_{t,1})$ $(z_{t,0}, z_{t,4})$ $(z_{t,1})$ (0.475)
w_t	all $\boldsymbol{x}$	$(x_{t,0}, x_{t,1}, x_{t,3})$ $(x_{t,2}, x_{t,5})$ $(x_{t,4})$ (0.359)	$(x_{t,0}, x_{t,1}, x_{t,3})$ $(x_{t,2}, x_{t,5})$ $(x_{t,4})$ (0.359)	$(x_{t,0}, x_{t,1}, x_{t,2}, x_{t,3}, x_{t,5})$ $(x_{t,4})$ (0.379)	$(x_{t,0}, x_{t,1}, x_{t,3})$ $(x_{t,2}, x_{t,5})$ $(x_{t,4})$ (0.359)
w_t	all $\boldsymbol{x}$ + all $\boldsymbol{z}$	$(x_{t,0}, x_{t,1}, x_{t,3})$ $(x_{t,5}, z_{t,1})$ $(x_{t,2}, z_{t,0})$ (0.354)	$(x_{t,0}, x_{t,1}, x_{t,3})$ $(x_{t,2}, x_{t,5}, z_{t,1})$ (0.358)	$(x_{t,0}, x_{t,1}, x_{t,2}, x_{t,3}, x_{t,5}, z_{t,0})$ (0.380)	$(x_{t,0}, x_{t,1}, x_{t,3})$ $(x_{t,2}, x_{t,5})$ (0.361)
w_t	some $\boldsymbol{x}$ + all $\boldsymbol{z}$	$(x_{t,0}, x_{t,1}, z_{t,0}, z_{t,3}, z_{t,4})$ $(x_{t,2})$ (0.582)	$(x_{t,0}, x_{t,1}, z_{t,0}, z_{t,1}, z_{t,3}, z_{t,4})$ $(x_{t,2})$ (0.576)	$(x_{t,0}, x_{t,1}, x_{t,2}, z_{t,0}, z_{t,3}, z_{t,4})$ (0.575)	$(x_{t,0}, x_{t,1}, z_{t,0}, z_{t,1}, z_{t,3}, z_{t,4})$ $(x_{t,2})$ (0.575)

At the low noise level, in both cases of "all $\boldsymbol{x}$" and "all $\boldsymbol{x}$ + all $\boldsymbol{z}$", all initialisations enable the algorithm to estimate the correct index structure for both y_t and w_t, with an exception in the "Linear" option for w_t. The "Linear" option for w_t selects the correct variables but fails to identify the two-index structure. This suggests that initialising the algorithm with a higher number of indices might be more effective than with a lower number. In the case of "some $\boldsymbol{x}$ + all $\boldsymbol{z}$", for both y_t and w_t, the models estimated under all initialisations include some noise variables. This indicates that when the available predictors are insufficient to capture the data signal, the algorithm might select irrelevant variables to make up for the missing signal.

When the fitted models are evaluated for y_t, in both cases of "all $\boldsymbol{x}$" and "all $\boldsymbol{x}$ + all $\boldsymbol{z}$", all initialisations result in a test set MSE of ≈ 0.01, which corresponds to the variance of the true model error. This confirms the accuracy with which the SMI Modelling algorithm estimates the index structure for y_t. For w_t, all the estimated models result in a test set MSE of ≈ 0.16. This is an interesting result as the test set MSE of an estimated model with an incorrect index structure, but with correct predictors ("Linear"), is similar to the models with correct index structure ("PPR", "Additive", and "Multiple"). This suggests that the selection of the predictor variables is more important than determining the index structure of the model in this case. For both y_t and w_t, in the case of "some $\boldsymbol{x}$ + all $\boldsymbol{z}$", MSE on the test set increases in comparison to the above cases, probably due to the inclusion of the noise variables.

Moreover, in contrast to y_t, the test set MSE values for w_t are higher than the variance of the true model error. Intuitively, the complexity of the model for w_t is higher than for y_t, where the total estimation error of two non-linear functions (corresponding to the two indices) for w_t might be higher than the error of estimating a single non-linear function for y_t.

As expected, the SMI Modelling algorithm's accuracy in estimating the index structure is generally lower with the high noise level than with the low noise level. At the high noise level, most of the estimated models have selected irrelevant variables for both y_t and w_t. In both the cases of "all $\boldsymbol{x}$" and "all $\boldsymbol{x}$ + all $\boldsymbol{z}$", all the models estimated for y_t (except for the "Additive" option in the "all $\boldsymbol{x}$" case) result in a test set MSE of ≈ 0.23, which is slightly lower than the variance of the true model error. This occurs irrespective of the inclusion of irrelevant variables in the "PPR" option for the "all $\boldsymbol{x}$" case and in the "Additive" and "Multiple" options for the "all $\boldsymbol{x}$ + all $\boldsymbol{z}$" case. This suggests over-fitting when there is a low signal-to-noise ratio in the data. The same result is observed for w_t, where, irrespective of the different index structures and predictor choices, the estimated models in the above two predictor combinations produce similar MSE values on the test set.

Similar to the previous case with the low noise level, when only part of $\boldsymbol{x}$ variables is provided, the test set MSE values increase for both y_t and w_t, with the estimated models for w_t producing higher test MSE values compared to those for y_t.

It is worth mentioning here that, in real-world forecasting problems, the true DGP is unknown, and we do not expect an estimated model to precisely capture the true DGP. Therefore, as long as the estimated model demonstrates good forecasting accuracy, the exact index structure is less important.

The simulation study further indicates that the choice of initialisation depends on the specific data and application. Thus, users are encouraged to follow a trial-and-error procedure to determine the most suitable initial model for a given application.

All SMI models estimated in this simulation experiment required less than 15 seconds of run time on an Apple Silicon MacBook Pro, equipped with an Apple M1 Pro chip and 16 GB of memory.

2.4 Empirical applications

2.4.1 Forecasting daily mortality

Studying the effects of various environmental exposures, such as weather-related variables, pollutants, and man-made environmental conditions, on human health is of significant importance in environmental epidemiology. Here, we apply the SMI Modelling algorithm to forecast daily mortality based on heat exposure.

Description of the data

For this analysis, we consider daily mortality and heat exposure data for the metropolitan area of Montreal, Québec, Canada, from 1990 to 2014 for June, July, and August (i.e. the summer months) (Masselot et al. 2023). The daily all-cause mortality data were obtained from the National Institute of Public Health, Québec, while *DayMet*, a 1km × 1km gridded data set (Thornton et al. 2021), was used to extract daily temperature and humidity data.

Figure 2.1 shows the time plots of daily deaths during the summer for the years 1990 to 1993. The series for each of the four years is presented separately in a faceted grid for visual clarity.

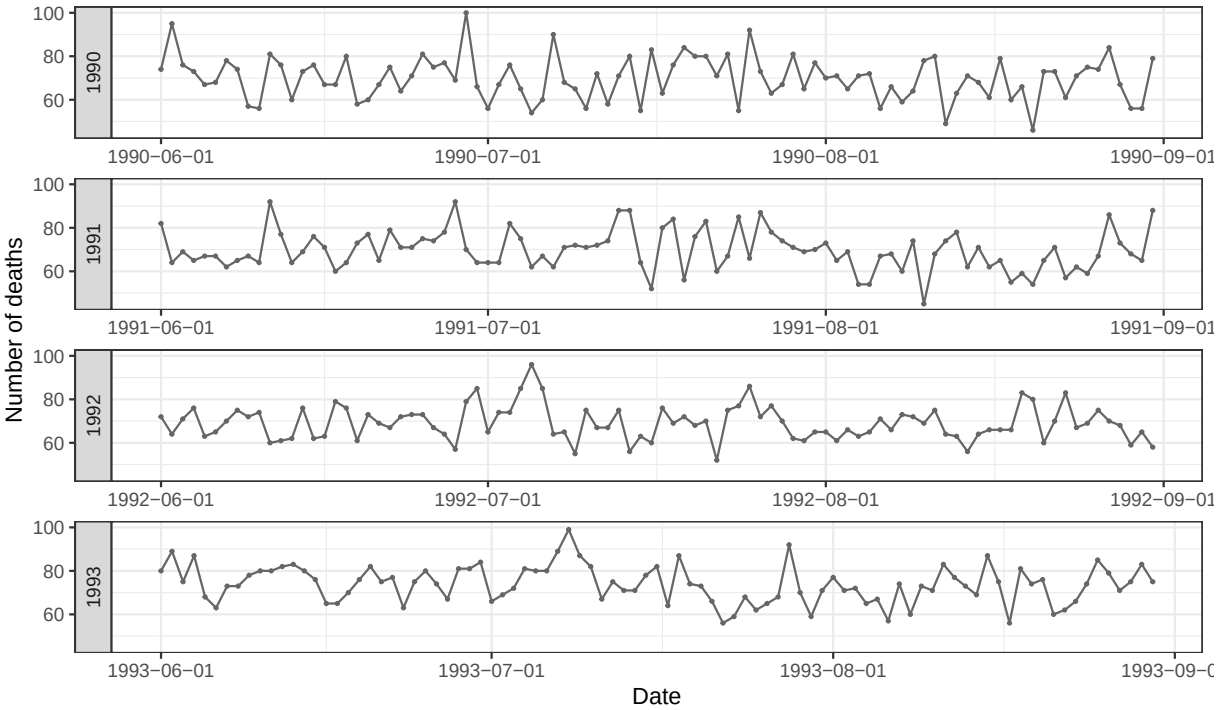


Figure 2.1: *Daily mortality in summer in Montreal, Canada, from 1990 to 1993.*

Maximum temperature, minimum temperature, and vapour pressure (representing the level of humidity) are considered predictors in this empirical study. The number of daily deaths is plotted against each of these predictors in Figure 2.2, where we observe that the relationships between these predictors and the response are slightly non-linear.

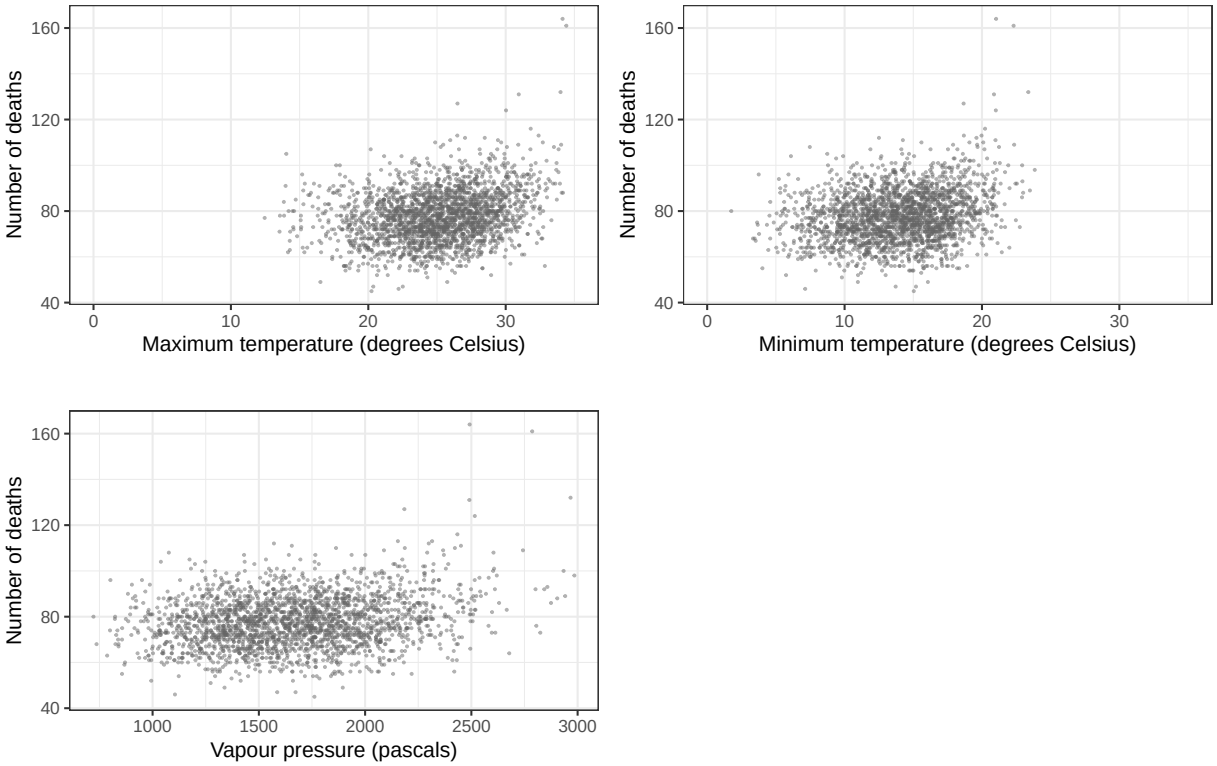


Figure 2.2: *Daily mortality in summer (from 1990 to 2014) plotted against maximum temperature, minimum temperature, and vapour pressure.*

Predictors considered

1) Daily death lags:

To incorporate the serial correlation present in the data into the modelling process, 14 lags of the daily deaths (denoted *Deaths_01* to *Deaths_14*) are used as predictors in the model.

2) Current maximum/minimum temperatures and lags:

In addition to the current maximum and minimum temperatures (denoted *Tmax* and *Tmin*, respectively), the temperature measurements up to 14 days prior (denoted *Tmax_01* to *Tmax_14* and *Tmin_01* to *Tmin_14*) are considered predictors in the forecasting model. This takes into account the cumulative effect of both current and recent past temperatures on a person's heat exposure.

3) Current vapour pressure and lags:

Similarly, the current value and 14 lags of vapour pressure (denoted *Vp* and *Vp_01* to *Vp_14*, respectively) are considered as predictors, as a proxy for the level of humidity.

4) Calendar effects:

Finally, a couple of calendar variables (*day of the season (DOS)* and *Year*) are included in the model to capture the annual trend and seasonality, which is a common practice in environmental epidemiology.

Modelling framework

Death lags, maximum temperature lags, minimum temperature lags, and vapour pressure lags are predictors that may enter the indices. The two calendar variables, *DOS* and *Year*, are included in the model as separate nonparametric components that do not enter any indices. Hence, the relevant SMI model can be written as

$$\mathbf{Deaths} = \beta_0 + \sum_{j=1}^{q} g_j(\boldsymbol{X}\boldsymbol{\alpha}_j) + f_1(\mathbf{DOS}) + f_2(\mathbf{Year}) + \boldsymbol{\varepsilon},$$

where

- **Deaths** is the vector containing daily death observations;
- β_0 is the model intercept;
- q is the unknown number of indices to be estimated via the proposed algorithm;
- $\boldsymbol{X}$ is a matrix containing the $p = 59$ predictor variables that may enter indices (i.e. death lags, maximum temperature lags, minimum temperature lags, and vapour pressure lags);
- $\boldsymbol{\alpha}_j$, $j = 1, \ldots, q$, are the index coefficient vectors, each of length p;
- $g_1, \ldots, g_q$, f_1, and f_2 are unknown nonparametric functions; and
- $\boldsymbol{\varepsilon}$ is the error term.

To evaluate the forecasting accuracy of the estimated SMI model, a time series cross-validation procedure is implemented using rolling window forecasting. Each rolling training window consists of 1656 observations, on which a SMI model is estimated. For each rolling training window, a subsequent validation window of 92 observations is used to tune the penalty parameters, λ_0 and λ_2. Once the tuning of the two penalty parameters is completed, the SMI model is re-fitted using the combined training and validation sets at each rolling window. The initial training window for model estimation spans from June 1990 to August 2007, and the window is rolled forward in steps of five observations to strike a balance between the number of windows and computational cost, resulting in a total of 93 windows. At each window, 92-step-ahead forecasts are generated.

The forecasting accuracy of the model is evaluated using MSE and Mean Absolute Error (MAE). We assume that the future values of the maximum and minimum temperatures and vapour pressure are known for use in the forecasting model; thus, it is a post hoc analysis. It is important to note that the two calendar variables (i.e. *DOS* and *Year*) are not truncated when generating forecasts for the test sets, as explained in Section 2.2.2, since such truncation would be inappropriate in the context of these variables.

Benchmark methods

In addition to our proposed SMI model, we also evaluate three benchmark models for comparison purposes.

The first benchmark is a nonparametric additive model formulated through *backward elimination (Backward)*, as proposed by Fan & Hyndman (2012). The process starts with all variables included in an additive model, and variables are progressively omitted until the optimal model is obtained based on the validation set MSE. Once the optimal model is obtained, the final model is re-fitted for the data set combining the training and validation sets at each rolling window.

The second benchmark is a *Groupwise Additive Index Model (GAIM)*, which can be written in the form

$$y_i = \sum_{j=1}^{q} g_j(\boldsymbol{\alpha}_j^T \boldsymbol{x}_{ij}) + \varepsilon_i,$$

where y_i is the response, $\boldsymbol{x}_{ij} \in \mathbb{R}^{l_j}$, $j = 1, \ldots, q$ are pre-specified non-overlapping subsets of $\boldsymbol{x}_i$, and $\boldsymbol{\alpha}_j$, $j = 1, \ldots, q$ are the corresponding vectors of index coefficients, g_j, $j = 1, \ldots, q$ are unknown, possibly non-linear functions, and ε_i is the random error (Wang, Xu & Zhu 2015; Masselot et al. 2023). For this application, death lags, maximum temperature lags, minimum temperature lags, and vapour pressure lags are intuitively categorised into four separate groups, with an index estimated for each group.

The third benchmark is a *Projection Pursuit Regression (PPR)* model (Friedman & Stuetzle 1981) given by

$$y_i = \sum_{j=1}^{q} g_j(\boldsymbol{\alpha}_j^T \boldsymbol{x}_i) + \varepsilon_i,$$

where y_i is the response, $\boldsymbol{x}_i$ is the p-dimensional predictor vector, $\boldsymbol{\alpha}_j = (\alpha_{j1}, \ldots, \alpha_{jp})^T$, $j = 1, \ldots, q$ are p-dimensional projection vectors (or vectors of "index coefficients"), g_j, $j = 1, \ldots, q$ are unknown non-linear functions, and ε_i is the random error. The number of indices in the *PPR* model was taken as five, which corresponds to the number of indices in the PPR model used in the initialisation of the SMI model estimation with the "PPR" initialisation option.

It is important to note a key difference between the *GAIM* and *PPR* models—in the *PPR* model, all predictors in the p-dimensional predictor vector, $\boldsymbol{x}_i$, enter into every estimated index. In contrast, the *GAIM* assigns each predictor to a single, predetermined index with no overlap.

Additionally, no parameter tuning or variable/model selection is performed for the two benchmark models, *GAIM* and *PPR*. Therefore, these models are estimated directly on the combined training and

validation sets at each rolling window, ensuring a fair comparison of forecasting performance across models.

When generating forecasts for a test set, all predictor variables treated non-linearly in these benchmark models are also truncated to be within the in-sample range, as explained in Section 2.2.2.

Results

For each rolling window, we estimated the SMI models using three different initialisation options—"PPR", "Additive", and "Linear" (denoted as *SMI-P*, *SMI-A*, and *SMI-L*, respectively)—for comparison purposes. The penalty parameters λ_0 and λ_2 were tuned within each rolling window using the greedy search procedure described in Section 2.2.2, based on the validation set MSE. The length of the sequence of possible values for λ_0 was set to $50+1$ (including zero).

Table 2.2 presents the cross-validated MSE and MAE values for the estimated SMI models, alongside all benchmark methods considered, across forecast horizons $h = 1, \ldots, 92$. The table also reports the MSE and MAE values averaged over specific periods—seven days (i.e. one week), 14 days (i.e. two weeks), 30 days (i.e. one month), 61 days (i.e. two months), and 92 days (i.e. three months of the summer season). These are denoted as $h = 1{:}\,7$, $h = 1{:}\,14$, $h = 1{:}\,30$, $h = 1{:}\,61$, and $h = 1{:}\,92$, respectively.

Table 2.2: *Cross-validated out-of-sample point forecast results for daily mortality in summer. The column* Horizon *indicates the forecast horizon (h). Columns 2–7 present the MSE values, and columns 8–13 present the MAE values for the estimated models:* SMI-P*,* SMI-A*,* SMI-L*,* Backward*,* GAIM*, and* PPR*, respectively. The minimum MSE and MAE values for each forecast horizon are shown in bold.*

	MSE						MAE					
Horizon	SMI-P	SMI-A	SMI-L	Backward	GAIM	PPR	SMI-P	SMI-A	SMI-L	Backward	GAIM	PPR
h=1	97.6	249.7	**94.7**	101.1	103.9	125.3	**7.8**	8.8	**7.8**	8.1	8.0	9.3
h=2	93.5	184.8	**92.9**	99.4	141.3	95.8	7.8	8.9	**7.6**	7.9	8.5	**7.6**
h=3	98.6	141.8	116.8	**92.4**	97.0	141.5	8.2	8.8	8.6	8.0	**7.7**	9.0
h=4	136.5	257.9	162.1	132.1	**123.4**	178.8	9.5	10.6	10.0	9.2	**8.8**	10.3
h=5	**107.3**	274.5	176.0	125.8	121.7	172.1	**8.3**	10.2	9.4	8.5	8.6	9.6
h=6	**95.1**	222.3	116.6	102.4	100.1	186.2	**7.6**	9.3	8.1	8.1	7.7	9.6
h=7	**98.0**	649.4	234.7	99.9	137.9	144.3	**7.6**	10.4	9.0	7.7	8.5	9.1
h=8	**87.6**	614.1	224.1	90.0	98.6	148.8	7.8	10.3	9.1	**7.7**	**7.7**	9.1
h=9	131.4	565.5	260.1	130.1	**120.3**	193.9	9.2	11.7	10.8	9.2	**8.8**	10.7
h=10	170.9	612.5	197.5	123.8	**113.7**	174.4	9.1	10.7	9.5	**8.5**	8.6	10.0
h=11	113.7	531.3	144.9	**100.1**	101.8	149.3	7.9	10.1	8.4	7.9	**7.8**	9.4
h=12	106.0	564.2	301.3	**101.3**	139.7	160.4	7.9	10.5	9.8	**7.8**	8.7	9.7
h=13	98.4	529.9	102.1	**87.8**	98.5	126.5	7.9	10.3	8.2	**7.7**	**7.7**	8.6

(continued)

	MSE						MAE					
Horizon	SMI-P	SMI-A	SMI-L	Backward	GAIM	PPR	SMI-P	SMI-A	SMI-L	Backward	GAIM	PPR
h=14	129.1	550.6	159.1	126.3	**119.7**	161.5	9.1	11.8	9.7	8.9	**8.8**	9.7
h=15	128.4	590.0	149.2	121.3	**109.8**	178.3	8.4	11.0	9.4	**8.2**	8.4	9.6
h=16	**96.0**	658.6	119.6	100.2	101.0	189.7	**7.8**	11.0	8.3	8.0	**7.8**	9.9
h=17	100.6	571.6	164.8	**96.3**	143.0	167.5	7.9	10.0	8.5	**7.6**	8.6	8.8
h=18	105.5	571.3	110.5	**90.8**	98.1	217.9	8.2	10.2	8.4	7.8	**7.6**	10.6
h=19	127.9	606.8	135.1	127.6	**116.1**	188.9	9.2	12.1	9.3	9.2	**8.6**	10.4
h=20	129.2	541.5	132.2	121.7	**115.5**	166.4	8.7	11.2	8.8	**8.4**	8.6	9.0
h=21	99.0	610.9	100.4	101.4	**98.9**	159.3	7.9	10.3	7.8	8.0	**7.6**	9.2
h=22	**99.6**	630.7	110.3	102.0	139.2	172.7	**7.6**	10.7	8.0	7.9	8.5	9.4
h=23	**84.4**	527.6	97.8	90.8	103.9	229.5	**7.5**	10.1	7.9	7.7	7.8	10.8
h=24	141.4	520.5	133.9	124.1	**117.0**	211.6	9.6	11.1	9.4	8.8	**8.6**	11.1
h=25	107.9	611.5	116.8	124.6	**104.2**	238.0	**8.2**	11.4	8.7	8.5	8.4	10.2
h=26	101.6	541.7	**94.6**	98.1	100.4	208.4	7.9	9.9	**7.6**	7.8	7.7	10.2
h=27	102.1	637.3	**101.9**	102.2	138.8	196.9	7.9	11.0	**7.7**	7.9	8.8	9.7
h=28	**94.6**	735.4	102.2	99.5	129.5	209.2	**7.8**	11.3	8.0	7.9	8.3	10.4
h=29	124.1	570.2	124.3	120.4	**114.4**	254.0	9.1	11.3	8.9	8.7	**8.5**	11.4
h=30	128.4	561.5	131.5	130.9	**108.7**	202.1	8.8	10.8	8.7	8.6	**8.5**	10.3
h=31	**89.9**	527.9	98.6	98.7	95.5	206.0	**7.3**	10.6	7.8	7.7	7.5	10.2
h=32	**95.8**	605.5	112.4	101.9	149.8	216.4	**7.9**	10.8	8.3	8.1	9.0	10.2
h=33	106.3	621.4	107.1	**99.3**	105.6	208.2	8.1	10.9	8.4	**7.9**	8.1	10.6
h=34	116.7	538.1	134.5	113.4	**110.6**	210.1	8.5	11.5	9.3	8.5	**8.4**	10.3
h=35	142.8	542.0	123.8	130.8	**112.3**	282.8	9.2	11.0	8.6	8.7	**8.5**	11.5
h=36	**93.9**	491.5	96.5	95.0	94.9	226.0	**7.5**	10.1	7.7	7.7	**7.5**	10.5
h=37	**102.0**	546.0	117.2	112.5	150.2	244.9	**7.8**	11.2	8.5	8.5	9.0	11.1
h=38	**95.5**	280.5	105.5	103.0	128.4	256.2	**7.7**	9.6	8.4	8.1	8.2	11.2
h=39	132.5	201.1	132.7	119.5	**119.2**	247.7	9.3	9.5	9.2	8.8	**8.7**	11.5
h=40	131.8	257.8	124.4	135.7	**116.0**	264.6	9.0	9.9	**8.6**	8.8	**8.6**	11.6
h=41	**98.4**	205.6	99.7	98.6	100.9	216.7	7.8	8.5	7.8	7.8	**7.7**	10.3
h=42	112.6	259.5	106.6	**104.0**	162.7	227.7	8.3	10.0	8.2	**8.1**	9.2	10.6
h=43	109.7	237.7	**97.4**	104.7	110.9	214.9	8.3	10.0	**7.7**	8.2	8.0	10.5
h=44	131.2	630.2	131.6	**116.8**	117.4	280.1	9.0	11.5	9.3	8.6	**8.5**	11.9
h=45	139.0	581.8	**124.1**	134.5	131.2	308.8	9.2	11.1	**8.6**	**8.6**	8.8	12.0
h=46	**96.5**	536.7	100.5	102.5	100.7	292.9	**7.7**	10.5	7.9	7.8	**7.7**	11.8
h=47	114.5	556.7	123.2	**107.8**	174.5	285.5	8.5	10.9	8.7	**8.3**	9.6	11.8
h=48	105.5	532.4	**99.3**	105.1	110.2	315.1	8.3	10.9	**8.0**	8.1	8.1	11.8
h=49	131.9	484.5	131.2	121.5	**120.3**	257.3	9.1	11.1	9.3	**8.8**	**8.8**	11.1
h=50	155.0	551.1	**129.9**	134.2	136.5	292.3	9.3	11.4	**8.8**	8.9	9.0	12.2
h=51	103.6	619.2	**100.0**	100.1	101.8	294.3	7.9	10.8	8.1	7.8	**7.7**	11.8
h=52	**100.5**	552.8	114.5	104.7	169.3	295.2	**7.9**	10.8	8.5	8.2	9.6	11.8

(continued)

	MSE						MAE					
Horizon	SMI-P	SMI-A	SMI-L	Backward	GAIM	PPR	SMI-P	SMI-A	SMI-L	Backward	GAIM	PPR
h=53	**104.4**	597.1	106.9	105.8	112.4	295.6	**8.1**	10.3	8.3	**8.1**	**8.1**	12.4
h=54	125.6	518.9	126.6	120.6	**116.8**	293.3	8.9	11.4	9.1	9.0	**8.6**	12.0
h=55	140.2	584.6	145.7	137.9	**121.6**	255.1	8.9	11.5	9.1	**8.8**	**8.8**	11.0
h=56	**97.5**	600.2	102.4	106.0	106.3	265.9	**7.9**	10.5	8.1	8.2	8.0	11.6
h=57	112.9	590.0	116.5	**110.3**	150.8	266.7	8.6	11.1	**8.5**	**8.5**	9.2	11.3
h=58	106.1	607.2	**98.6**	106.4	119.2	283.5	8.2	11.0	**8.0**	8.2	8.3	11.6
h=59	121.3	633.1	126.2	120.0	**118.6**	249.4	8.9	12.0	9.1	8.9	**8.7**	11.9
h=60	**132.3**	681.6	136.1	138.4	133.7	300.8	9.1	11.9	9.2	**9.0**	**9.0**	12.1
h=61	102.5	688.9	**92.6**	103.1	98.9	230.0	8.0	10.8	7.7	8.0	**7.6**	10.7
h=62	114.9	635.7	119.2	**107.0**	153.1	237.6	8.6	11.1	8.5	**8.2**	9.0	11.2
h=63	123.2	620.7	**93.7**	108.8	124.4	268.0	8.4	10.8	**7.8**	8.1	8.1	12.1
h=64	135.6	560.0	132.8	128.8	**122.2**	316.3	9.4	11.7	9.3	9.3	**8.9**	12.7
h=65	135.0	651.2	133.7	132.8	**128.6**	316.6	9.0	11.4	8.9	**8.7**	8.9	11.9
h=66	**100.6**	688.0	107.9	109.7	102.7	258.8	**7.9**	11.0	8.1	8.3	**7.9**	11.3
h=67	**103.0**	608.7	122.0	104.5	144.7	198.3	**8.0**	10.5	8.6	8.2	8.7	10.5
h=68	**107.4**	483.8	108.7	110.1	123.8	284.7	**8.3**	10.4	**8.3**	**8.3**	8.4	12.0
h=69	129.2	642.7	134.9	125.4	**124.2**	303.6	9.3	12.0	9.5	9.2	**9.0**	11.7
h=70	145.5	576.1	**120.2**	130.1	131.0	268.7	9.3	11.2	8.7	**8.6**	8.9	11.7
h=71	96.5	616.1	**90.0**	106.5	99.0	246.4	7.7	10.7	**7.5**	8.3	7.7	11.4
h=72	110.7	524.9	110.3	**108.4**	152.8	283.5	8.6	10.5	**8.2**	8.3	9.0	11.7
h=73	**99.5**	628.3	107.8	112.8	124.2	237.1	**8.0**	11.0	8.1	8.4	8.3	12.0
h=74	137.0	693.9	130.8	123.0	**121.4**	285.8	9.6	12.0	9.3	9.1	**8.9**	12.1
h=75	**130.9**	380.8	139.8	132.4	133.4	328.2	8.9	10.8	8.9	**8.7**	9.0	12.5
h=76	**93.3**	533.5	100.9	102.1	95.2	285.6	**7.6**	10.3	8.1	8.1	**7.6**	12.2
h=77	109.1	585.1	109.6	**109.0**	147.5	303.9	8.5	11.2	8.4	**8.3**	9.0	11.9
h=78	**103.8**	575.6	104.3	119.2	135.1	301.2	**8.1**	11.1	**8.1**	8.7	8.5	11.9
h=79	134.9	564.1	141.3	125.9	**125.7**	301.0	9.5	11.6	9.7	9.2	**9.1**	12.1
h=80	151.7	622.9	**111.9**	127.0	129.9	350.4	9.0	11.4	**8.6**	8.7	8.9	13.2
h=81	100.1	678.6	**90.1**	100.1	95.9	359.2	8.0	10.9	7.8	8.0	**7.7**	12.5
h=82	120.7	552.0	119.6	**106.6**	149.7	300.0	8.6	11.0	8.7	**8.1**	9.1	12.5
h=83	**109.9**	679.4	189.6	117.7	130.6	315.2	8.6	11.3	9.6	8.6	**8.4**	12.9
h=84	124.9	638.9	181.7	124.3	**120.0**	407.3	9.2	12.3	10.0	9.1	**8.8**	14.2
h=85	158.1	588.8	217.2	**124.3**	126.4	326.4	8.7	11.0	9.5	**8.4**	8.8	13.2
h=86	109.5	513.5	319.8	102.2	**93.6**	293.0	8.1	10.6	9.4	8.1	**7.6**	12.1
h=87	116.0	583.3	336.3	**107.3**	147.9	323.7	8.4	11.0	9.8	**8.2**	9.0	12.8
h=88	**105.0**	470.7	332.6	121.0	125.1	337.1	**8.4**	10.8	10.1	8.6	**8.4**	13.6
h=89	132.8	543.0	333.4	121.9	**116.6**	297.2	9.4	11.9	10.6	9.1	**8.6**	12.1
h=90	**121.2**	554.9	255.5	125.0	127.8	338.6	**8.6**	11.2	9.7	8.7	8.9	13.0
h=91	101.3	554.4	160.5	99.5	**92.1**	297.4	8.3	10.1	8.6	7.9	**7.5**	12.0

(continued)

	MSE						MAE					
Horizon	SMI-P	SMI-A	SMI-L	Backward	GAIM	PPR	SMI-P	SMI-A	SMI-L	Backward	GAIM	PPR
h=92	111.0	669.1	298.1	**107.5**	145.1	342.2	8.3	11.0	10.0	**8.2**	8.6	12.6
h=1:7	**103.8**	282.9	142.0	107.6	117.9	149.1	**8.1**	9.6	8.6	8.2	8.3	9.2
h=1:14	111.7	424.9	170.2	**108.0**	115.5	154.2	8.3	10.2	9.0	**8.2**	8.3	9.4
h=1:30	111.1	514.5	143.6	**108.8**	115.2	178.3	8.3	10.5	8.7	**8.2**	8.3	9.8
h=1:61	112.8	513.1	129.0	**110.8**	118.9	220.2	**8.3**	10.6	8.6	**8.3**	8.4	10.6
h=1:92	114.7	538.2	140.5	**112.4**	121.1	247.2	**8.4**	10.8	8.7	**8.4**	**8.4**	11.1

According to Table 2.2, *SMI-P* achieves the highest forecasting accuracy over the averaged short-term horizon $h = 1:7$, outperforming the other estimated SMI models and the benchmark methods. However, as the forecast horizon increases, the *Backward* model yields the best overall performance. Among the three SMI models, *SMI-P* performs best and continues to outperform the *GAIM* and *PPR* models. It is worth noting that *SMI-A* produces comparatively higher forecast errors than all other estimated models. This is likely because the Gurobi solver reported the optimisation problem as non-convex for this model. As a result, due to the time limits and MIP gap constraints applied during model fitting (see Section 2.4.3), only sub-optimal solutions were obtained.

Figures 2.3, 2.4, 2.5, and 2.6 depict the index variables selected within each of the 93 rolling windows, for the estimated *SMI-P*, *SMI-A*, *SMI-L*, and *Backward* models, respectively. Each figure includes a colour gradient column denoting the frequency with which each variable is selected across all windows. Correspondingly, Tables 2.3, 2.4, 2.5, and 2.6 present the associated selection frequencies for each model, thereby providing complementary information to the respective figures.

Table 2.3: *Selection frequency of each index variable in the estimated* SMI-P *models for forecasting daily mortality in summer across 93 rolling windows. The columns labelled* Index Variable *list the variable names, while the columns labelled* Selection Frequency *show the number of rolling windows in which each variable was selected. The columns are repeated for compact presentation.*

Index Variable	Selection Frequency	Index Variable	Selection Frequency
Deaths_01	58	Tmin_01	42
Deaths_02	41	Tmin_02	43
Deaths_03	43	Tmin_03	39
Deaths_04	40	Tmin_04	41
Deaths_05	40	Tmin_05	45
Deaths_06	40	Tmin_06	39
Deaths_07	41	Tmin_07	40

(continued)

Index Variable	Selection Frequency	Index Variable	Selection Frequency
Deaths_08	41	Tmin_08	43
Deaths_09	40	Tmin_09	40
Deaths_10	37	Tmin_10	40
Deaths_11	40	Tmin_11	40
Deaths_12	39	Tmin_12	40
Deaths_13	44	Tmin_13	38
Deaths_14	40	Tmin_14	51
Tmax	72	Vp	50
Tmax_01	49	Vp_01	48
Tmax_02	68	Vp_02	47
Tmax_03	39	Vp_03	46
Tmax_04	40	Vp_04	45
Tmax_05	39	Vp_05	40
Tmax_06	41	Vp_06	41
Tmax_07	44	Vp_07	41
Tmax_08	40	Vp_08	39
Tmax_09	39	Vp_09	37
Tmax_10	40	Vp_10	41
Tmax_11	39	Vp_11	40
Tmax_12	40	Vp_12	41
Tmax_13	40	Vp_13	41
Tmax_14	39	Vp_14	41
Tmin	44		

Table 2.4: *Selection frequency of each index variable in the estimated* SMI-A *models for forecasting daily mortality in summer across 93 rolling windows. The columns labelled* Index Variable *list the variable names, while the columns labelled* Selection Frequency *show the number of rolling windows in which each variable was selected. The columns are repeated for compact presentation.*

Index Variable	Selection Frequency	Index Variable	Selection Frequency
Deaths_01	44	Tmin_01	30
Deaths_02	29	Tmin_02	34
Deaths_03	31	Tmin_03	28
Deaths_04	27	Tmin_04	29
Deaths_05	27	Tmin_05	29
Deaths_06	29	Tmin_06	27
Deaths_07	31	Tmin_07	27
Deaths_08	28	Tmin_08	33
Deaths_09	27	Tmin_09	27

(continued)

Index Variable	Selection Frequency	Index Variable	Selection Frequency
Deaths_10	28	Tmin_10	28
Deaths_11	30	Tmin_11	27
Deaths_12	28	Tmin_12	28
Deaths_13	42	Tmin_13	28
Deaths_14	28	Tmin_14	42
Tmax	73	Vp	42
Tmax_01	52	Vp_01	49
Tmax_02	65	Vp_02	45
Tmax_03	33	Vp_03	39
Tmax_04	28	Vp_04	34
Tmax_05	29	Vp_05	37
Tmax_06	27	Vp_06	34
Tmax_07	38	Vp_07	30
Tmax_08	27	Vp_08	29
Tmax_09	28	Vp_09	28
Tmax_10	28	Vp_10	28
Tmax_11	28	Vp_11	29
Tmax_12	28	Vp_12	33
Tmax_13	31	Vp_13	28
Tmax_14	28	Vp_14	38
Tmin	32		

Table 2.5: *Selection frequency of each index variable in the estimated* SMI-L *models for forecasting daily mortality in summer across 93 rolling windows. The columns labelled* Index Variable *list the variable names, while the columns labelled* Selection Frequency *show the number of rolling windows in which each variable was selected. The columns are repeated for compact presentation.*

Index Variable	Selection Frequency	Index Variable	Selection Frequency
Deaths_01	53	Tmin_01	28
Deaths_02	25	Tmin_02	27
Deaths_03	24	Tmin_03	23
Deaths_04	23	Tmin_04	23
Deaths_05	23	Tmin_05	28
Deaths_06	25	Tmin_06	23
Deaths_07	24	Tmin_07	23
Deaths_08	26	Tmin_08	26
Deaths_09	25	Tmin_09	23
Deaths_10	23	Tmin_10	23
Deaths_11	23	Tmin_11	24

(continued)

Index Variable	Selection Frequency	Index Variable	Selection Frequency
Deaths_12	23	Tmin_12	23
Deaths_13	32	Tmin_13	23
Deaths_14	23	Tmin_14	44
Tmax	77	Vp	37
Tmax_01	40	Vp_01	33
Tmax_02	61	Vp_02	35
Tmax_03	26	Vp_03	32
Tmax_04	23	Vp_04	27
Tmax_05	24	Vp_05	30
Tmax_06	23	Vp_06	26
Tmax_07	23	Vp_07	23
Tmax_08	24	Vp_08	25
Tmax_09	23	Vp_09	24
Tmax_10	22	Vp_10	25
Tmax_11	23	Vp_11	24
Tmax_12	23	Vp_12	29
Tmax_13	23	Vp_13	24
Tmax_14	24	Vp_14	29
Tmin	28		

Table 2.6: *Selection frequency of each index variable in the estimated* Backward *models for forecasting daily mortality in summer across 93 rolling windows. The columns labelled* Index Variable *list the variable names, while the columns labelled* Selection Frequency *show the number of rolling windows in which each variable was selected. The columns are repeated for compact presentation.*

Index Variable	Selection Frequency	Index Variable	Selection Frequency
Deaths_01	43	Tmin_01	81
Deaths_02	56	Tmin_02	71
Deaths_03	69	Tmin_03	72
Deaths_04	68	Tmin_04	76
Deaths_05	80	Tmin_05	69
Deaths_06	72	Tmin_06	76
Deaths_07	53	Tmin_07	86
Deaths_08	45	Tmin_08	80
Deaths_09	79	Tmin_09	82
Deaths_10	71	Tmin_10	74
Deaths_11	52	Tmin_11	78
Deaths_12	46	Tmin_12	74
Deaths_13	59	Tmin_13	56

(continued)

Index Variable	Selection Frequency	Index Variable	Selection Frequency
Deaths_14	66	Tmin_14	52
Tmax	85	Vp	76
Tmax_01	75	Vp_01	40
Tmax_02	66	Vp_02	65
Tmax_03	44	Vp_03	78
Tmax_04	52	Vp_04	84
Tmax_05	54	Vp_05	79
Tmax_06	52	Vp_06	74
Tmax_07	60	Vp_07	74
Tmax_08	48	Vp_08	66
Tmax_09	42	Vp_09	87
Tmax_10	85	Vp_10	74
Tmax_11	82	Vp_11	87
Tmax_12	68	Vp_12	70
Tmax_13	58	Vp_13	69
Tmax_14	54	Vp_14	58
Tmin	78		

Figures 2.3, 2.4, and 2.5 indicate that, among the three estimated SMI models, the *SMI-L* model yields sparser results across a comparatively larger number of rolling windows. However, all three SMI models also produce some rolling windows in which no index variables are selected. In contrast, Figure 2.6 shows that the estimated *Backward* models are generally denser than the SMI models, particularly in comparison with *SMI-A*, despite both approaches beginning with the same full nonparametric additive model. The greater density of the *Backward* models likely arises from their use of a direct backward elimination procedure, which sequentially removes variables without applying coefficient penalisation. In comparison, the *SMI-A* model employs iterative optimisation with penalisation, simultaneously shrinking coefficients and grouping predictors, thereby promoting sparser solutions. Note that the *Backward* model has not produced any rolling windows in which no index variables are selected.

Among the 59 index variables, the current maximum temperature (*Tmax*), its first and second lags (*Tmax_01* and *Tmax_02*), and the first lag of the number of deaths (*Deaths_01*) consistently display higher selection frequencies across all three estimated SMI models. In the case of the *Backward* model, many additional variables show higher selection frequencies, reflecting its generally denser structure.

Examining the SMI models estimated in adjacent rolling windows reveals that periods of relatively denser models are sometimes immediately followed by sparser models, or vice versa. Ideally, we would

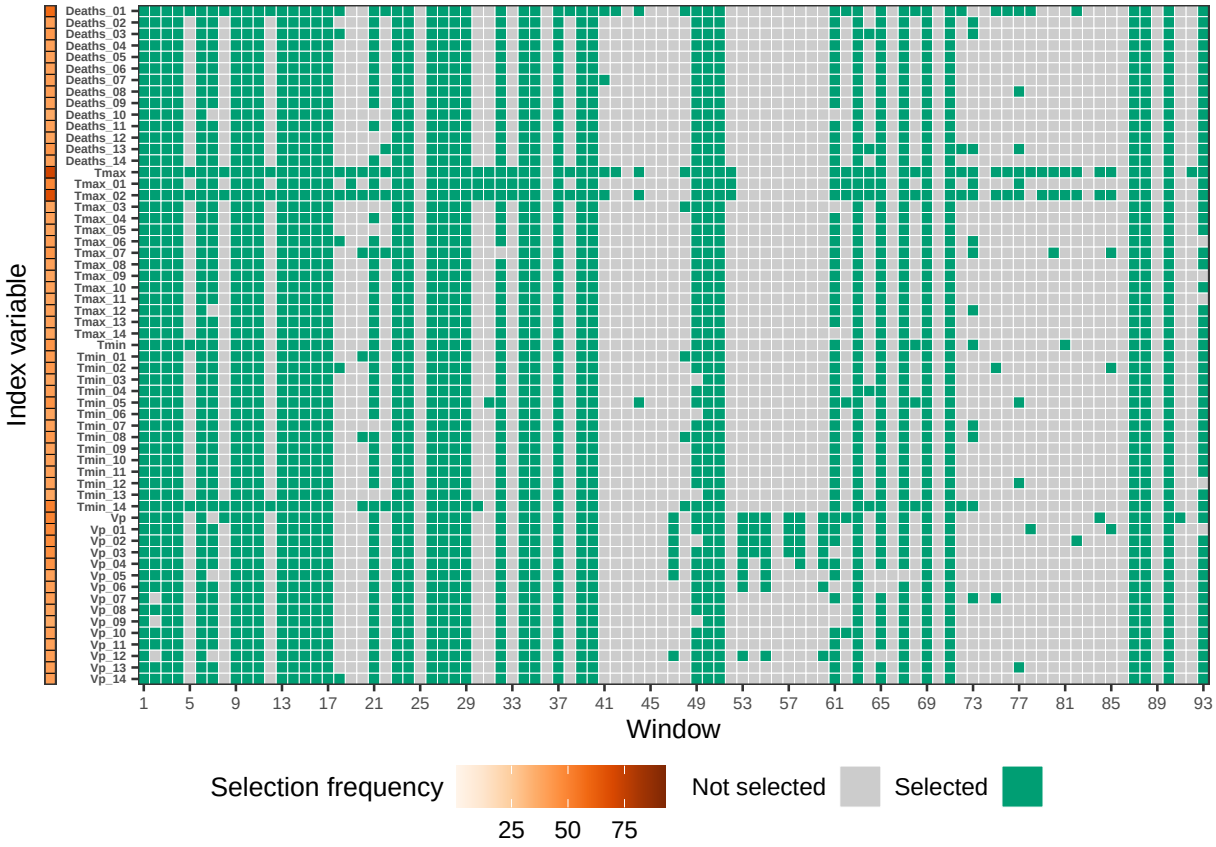


Figure 2.3: *Plot illustrating the index variables (i.e. predictors considered for inclusion in the indices) selected in the estimated* SMI-P *models for forecasting daily mortality in summer across 93 rolling windows. The colour gradient column on the left-hand side of the plot indicates the selection frequency of each index variable across all rolling windows, with darker shades representing higher selection frequencies and lighter shades representing lower selection frequencies. A complementary table of selection frequencies is provided in Table 2.3.*

expect models estimated in adjacent windows to have a similar structure. One possible explanation is that the SMI Modelling algorithm becomes trapped in local minima. Two different solutions selected in nearby windows may be close to each other in the solution space, but the algorithm converges to one local minimum in one window and to another in the next, resulting in markedly different models for adjacent windows. Additionally, in real-world scenarios, high correlations between predictors can often result in multiple distinct models with comparable performance, a situation that appears to be reflected in this empirical application. However, further research is required to better understand this observation.

The rolling window model estimation procedure from which the above results were obtained was carried out on the MonARCH (Monash Advanced Research Computing Hybrid) HPC Cluster using parallelisation. This was necessary as the estimation of the SMI models, particularly due to their automatic predictor selection and grouping mechanism implemented through an iterative optimisation algorithm, was computationally intensive and time-consuming.

Figure 2.4: *Plot illustrating the index variables (i.e. predictors considered for inclusion in the indices) selected in the estimated* SMI-A *models for forecasting daily mortality in summer across 93 rolling windows. The colour gradient column on the left-hand side of the plot indicates the selection frequency of each index variable across all rolling windows, with darker shades representing higher selection frequencies and lighter shades representing lower selection frequencies. A complementary table of selection frequencies is provided in Table 2.4.*

To indicate the comparative run times, the approximate times required to estimate each model (including the three benchmark models) for the first rolling window are presented in Table 2.7. These run times were recorded by estimating the models for only the first rolling window on an Apple Silicon MacBook Pro equipped with an Apple M1 Pro chip and 16 GB of memory. Note that these run times are reported as approximate indications of the computation time per rolling window; the actual run times for each model on each rolling window may vary depending on the characteristics of the data in that particular window.

Table 2.7: *Approximate run times for the estimated models for forecasting daily mortality in summer, in the first rolling window.*

Estimated Model	Run Time
SMI-P	4.57h
SMI-A	>24h
SMI-L	53.31m

(continued)

Estimated Model	Run Time
Backward	>24h
GAIM	21.65s
PPR	1.30s

According to Table 2.7, among the three estimated SMI models, *SMI-A* is the most time-consuming. Our experiments indicate that the run time of a SMI model depends on both the number of index variables and the number of indices estimated within the model. The "Additive" initialisation employs a nonparametric additive structure that assigns a separate index to each variable, thereby increasing the number of indices to be estimated relative to the "PPR" or "Linear" initialisations, and consequently raising the overall computational cost. However, as mentioned earlier, in this particular case, the Gurobi solver identified the optimisation problem for *SMI-A* as non-convex, which resulted in an extended computation time, reaching the time limit imposed for a single optimisation iteration (i.e. 600 seconds) in most iterations. Among the three SMI models, *SMI-L* exhibits the shortest run time, likely because it is initialised using a single-index structure.

We can observe that the *Backward* model also requires considerably more computational time, owing to the large number of nonparametric components that must be estimated iteratively at each step of the backward elimination procedure. In contrast, the *GAIM* and *PPR* models are computationally efficient.

It is worth noting that *SMI-P* achieved forecasting performance only marginally lower than that of the *Backward* model. Therefore, when considering the trade-off between computational time and forecasting accuracy, the substantially longer run time required by the *Backward* model did not yield a correspondingly greater improvement in forecasting performance compared with *SMI-P*.

2.4.2 Forecasting daily solar intensity

Next, we utilise the SMI Modelling algorithm to forecast daily solar intensity, using other weather conditions.

Description of the data

We use solar intensity and other weather variables measured at the Davis weather station in Amherst, Massachusetts, obtained from the *UMass Trace Repository* (*The UMass Trace Repository* 2023). The data were recorded every five minutes from 21 February 2006 to 27 February 2013, using sensors for measuring weather variables. We converted the five-minute data to daily data by averaging each

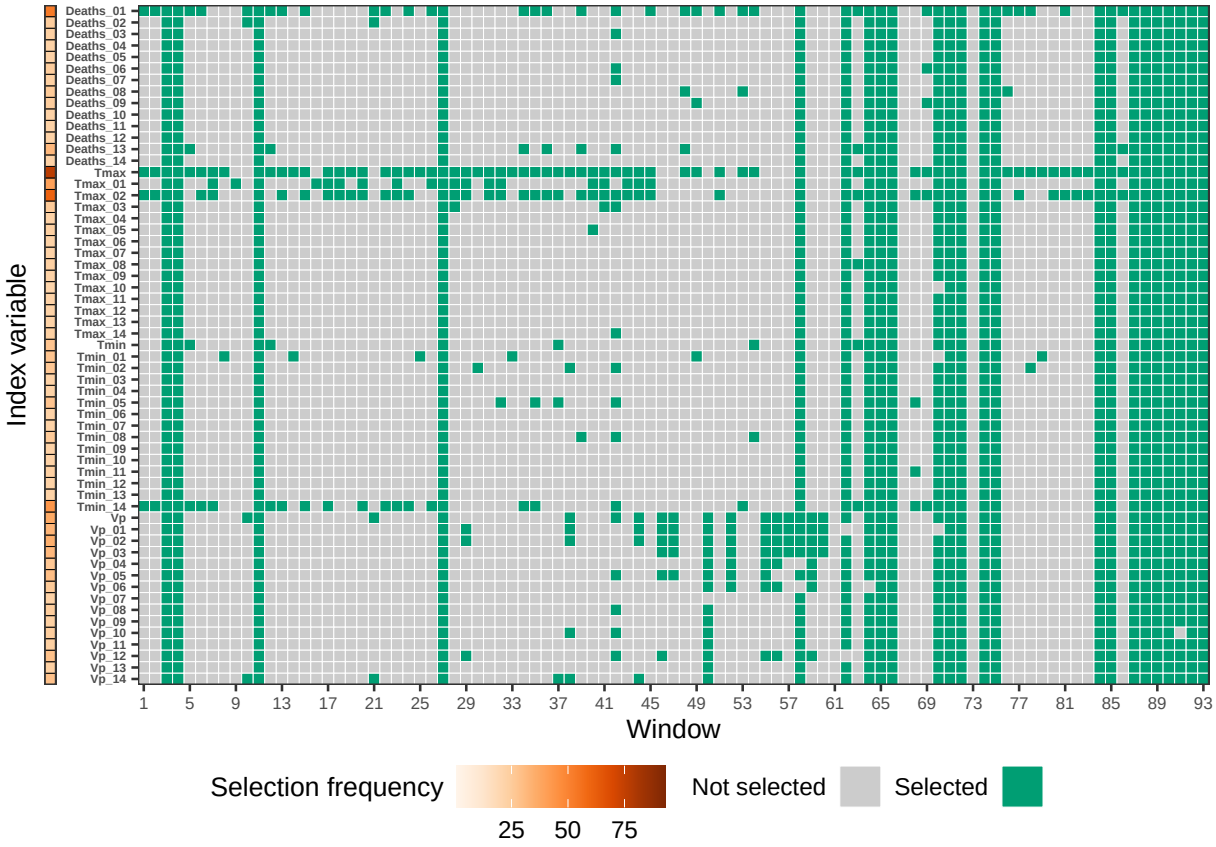


Figure 2.5: *Plot illustrating the index variables (i.e. predictors considered for inclusion in the indices) selected in the estimated* SMI-L *models for forecasting daily mortality in summer across 93 rolling windows. The colour gradient column on the left-hand side of the plot indicates the selection frequency of each index variable across all rolling windows, with darker shades representing higher selection frequencies and lighter shades representing lower selection frequencies. A complementary table of selection frequencies is provided in Table 2.5.*

variable over each day. The main reason for this conversion is to reduce the inflated number of zeros in the solar intensity series resulting from the absence of solar radiation during night-time.

Figure 2.7 shows the time plot of daily solar intensity over the entire period, clearly illustrating the annual seasonality present in the data. The gaps represent days with missing observations.

Additionally, there are three distinct periods of unusual observations identifiable in the data: from 3 January 2011 to 24 February 2011, from 11 November 2011 to 6 December 2011, and from 16 September 2012 to 19 September 2012. These periods were treated as missing data and subsequently removed from the analysis. The data set, after excluding these segments, is shown in Figure 2.8.

In Figure 2.9, the daily solar intensity is plotted against each of the predictors: temperature, dew point, wind, rain, and humidity. We can observe that the relationships between these predictors and the response are non-linear.

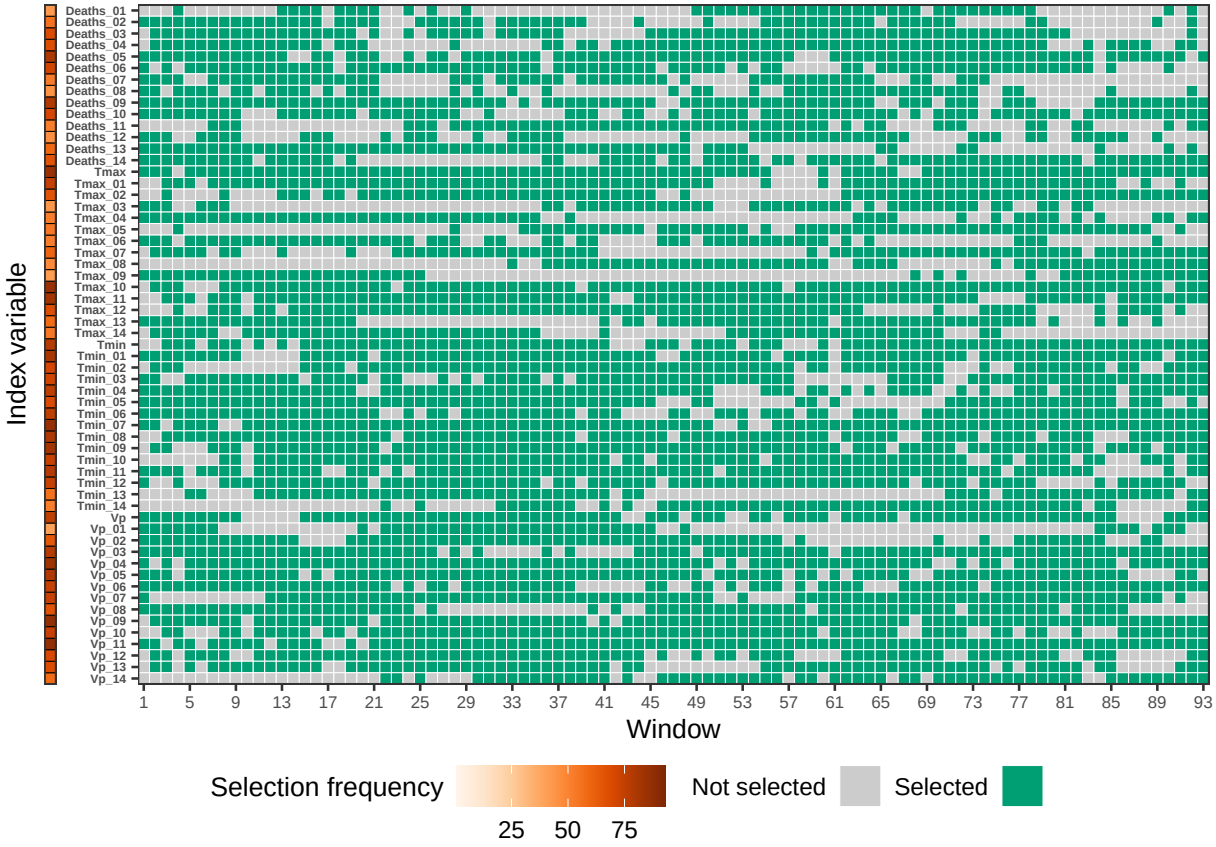


Figure 2.6: *Plot illustrating the index variables (i.e. predictors considered for inclusion in the indices) selected in the estimated* Backward *models for forecasting daily mortality in summer across 93 rolling windows. The colour gradient column on the left-hand side of the plot indicates the selection frequency of each index variable across all rolling windows, with darker shades representing higher selection frequencies and lighter shades representing lower selection frequencies. A complementary table of selection frequencies is provided in Table 2.6.*

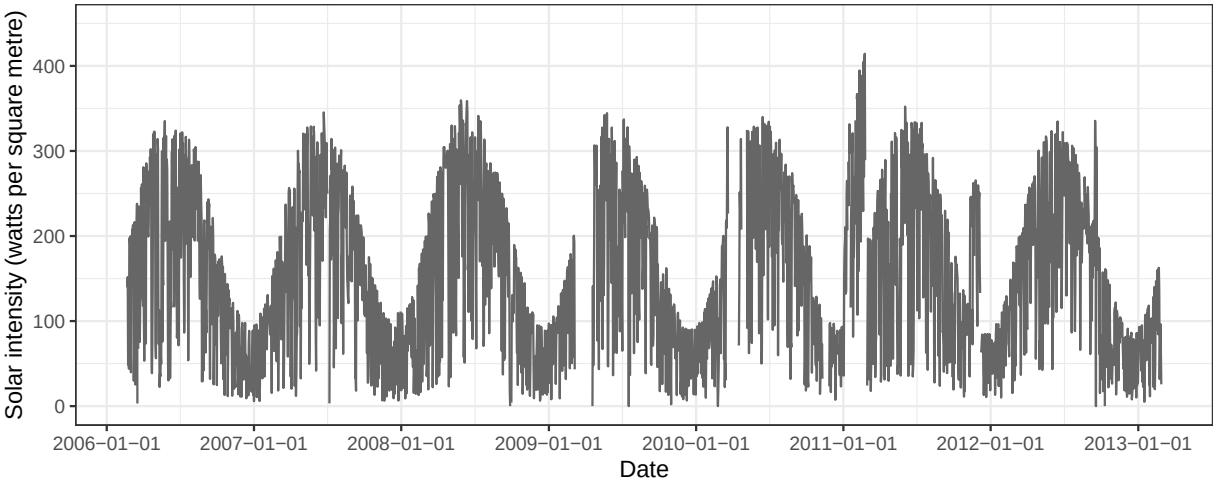


Figure 2.7: *Daily solar intensity in Amherst, Massachusetts, from February 2006 to February 2013—before removing unusual observations.*

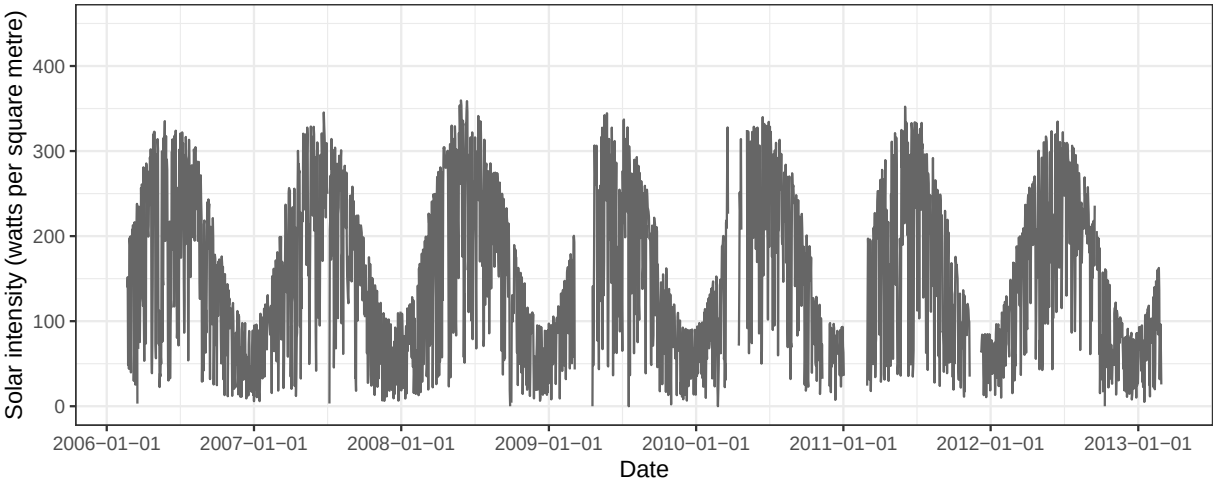


Figure 2.8: *Daily solar intensity in Amherst, Massachusetts, from February 2006 to February 2013—after removing unusual observations.*

Predictors considered

1) Solar intensity lags:

Three lags of daily solar intensity (denoted *Solar_01* to *Solar_03*) are used as predictors to incorporate serial correlations present in the data into the modelling process.

2) Current weather variables and lags:

In addition to current temperature, dew point, wind speed, rain, and humidity (denoted *Temp*, *Dewpt*, *Wind*, *Rain*, and *Humid*, respectively), the measurements of the three previous days (denoted *Temp_01* to *Temp_03*, *Dewpt_01* to *Dewpt_03*, *Wind_01* to *Wind_03*,, *Rain_01* to *Rain_03*, and *Humid_01* to *Humid_03*) for each of these weather variables are also included as predictors in the forecasting model.

3) Calendar effects:

Finally, a calendar variable, *day of the year (DOY)*, is incorporated into the model to account for annual seasonality. This variable is included as a smooth function using Fourier terms. A Fourier series approximates a periodic function using an infinite sum of sine and cosine terms (Tolstov 1976). Since seasonal patterns are periodic, Fourier terms can be used as an alternative to seasonal dummy variables in forecasting models, particularly for long seasonal periods. If m represents the seasonal period, the Fourier terms that serve as predictor variables in a forecasting model can be expressed as $x_{1,t} = \sin(2\pi t/m)$, $x_{2,t} = \cos(2\pi t/m)$, $x_{3,t} = \sin(4\pi t/m)$, $x_{4,t} = \cos(4\pi t/m)$, etc. (Hyndman & Athanasopoulos 2021).

We consider the number of Fourier term pairs (sine and cosine), denoted by K, ranging from $K = 1$ to $K = 10$, with $m = 365.25$ to account for leap years. The SMI model with "Linear" initialisation is used to determine the number of Fourier term pairs that best capture the *DOY* effect, based on

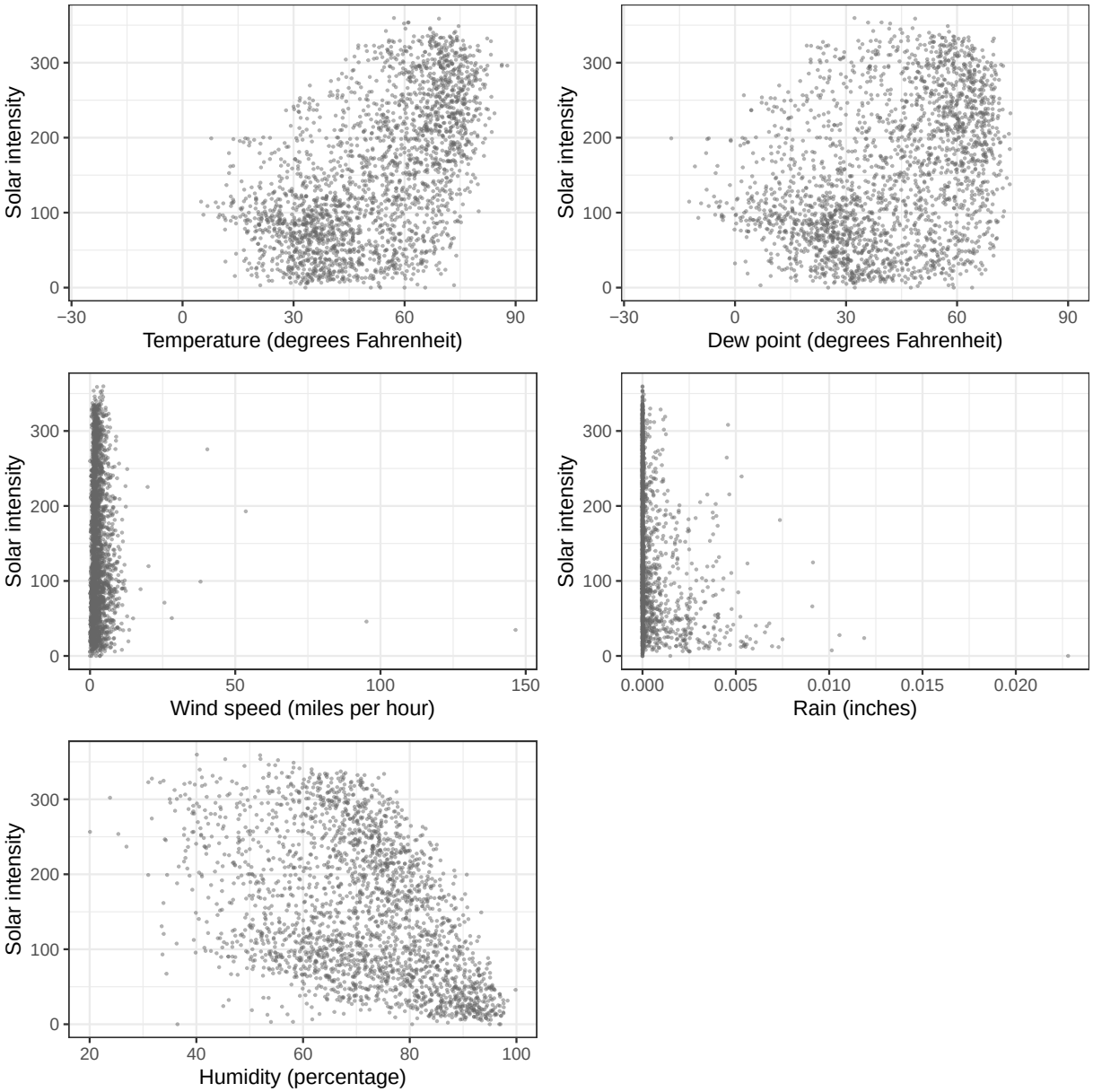


Figure 2.9: *Daily solar intensity (from February 2006 to February 2013) plotted against temperature, dew point, wind speed, rain, and humidity.*

the validation set MSE. That is, SMI models are fitted using different numbers of Fourier term pairs, and the number corresponding to the model that results in the lowest validation set MSE is selected. This selection process is carried out within each rolling window in the time series cross-validation procedure used to assess the forecasting performance of the estimated models. The details of the cross-validation procedure are provided in the following section.

Modelling framework

The lags of solar intensity and the lags of weather variables are considered predictors that may enter the indices. The Fourier terms capturing the day of the year effect are included in the model as linear variables (so that they are not part of the indices).

Hence, the relevant SMI model can be written as

$$\mathbf{Solar} = \beta_0 + \sum_{j=1}^{q} g_j(\boldsymbol{X}\boldsymbol{\alpha}_j) + \sum_{k=1}^{K}(\theta_k \mathbf{DOY\_S_k} + \delta_k \mathbf{DOY\_C_k}) + \boldsymbol{\varepsilon},$$

where

- **Solar** is the vector containing daily observations of solar intensity;
- β_0 is the model intercept;
- q is the unknown number of indices to be estimated via the proposed algorithm;
- $\boldsymbol{X}$ is a matrix containing the $p = 23$ predictor variables that may enter the indices (i.e. solar intensity, temperature, dew point, wind speed, rain, and humidity lags);
- $\boldsymbol{\alpha}_j$, $j = 1, \ldots, q$ are the index coefficient vectors, each of length p;
- $g_1, \ldots, g_q$ are unknown nonparametric functions;
- $\mathbf{DOY\_S_k}$ and $\mathbf{DOY\_C_k}$, $k = 1, \ldots, K$ are the sine and cosine terms of the Fourier term pairs corresponding to the variable **DOY** (The number of pairs K is selected for each rolling window separately, using the SMI model estimated with "Linear" initialisation.);
- θ_k and δ_k, $k = 1, \ldots, K$ are the coefficients of $\mathbf{DOY\_S_k}$ and $\mathbf{DOY\_C_k}$ respectively; and
- $\boldsymbol{\varepsilon}$ is the error term.

As in the previous application, a time series cross-validation procedure is implemented using a rolling window forecasting approach. Each rolling training window comprises 1569 observations, on which a SMI model is estimated. For each rolling training window, a subsequent validation window of 45 observations is used to tune the penalty parameters and to select the optimal number of Fourier term pairs to capture the effect of the *DOY* variable.

Once the tuning of the two penalty parameters and the selection of Fourier terms are completed, the models are re-estimated using the combined training and validation sets at each rolling window. The initial training window spans from 21 February 2006 to 8 June 2010, and the windows are rolled forward in increments of 10 observations, resulting in 93 windows. For each window, 30-step-ahead forecasts are generated.

To ensure the reliability of the validation process, any window in which the first two-thirds of the validation set consists of missing values is excluded, as it would not provide sufficient data to compute a meaningful validation MSE. Furthermore, windows, where the test set contains any missing values, are omitted to ensure that forecast errors can be calculated across the full forecast horizon in every rolling window included in the analysis. After excluding such windows, a total of 55 rolling windows remain.

The forecasting accuracy of the model is evaluated using MSE and MAE. We assume that the future values of the weather variables are known, thus making it a post hoc analysis.

Results

We estimated SMI models for the solar intensity data using three different initialisation options—"PPR", "Additive", and "Linear" (denoted as *SMI-P*, *SMI-A*, and *SMI-L*, respectively). We tuned the penalty parameters λ_0 and λ_2 within each rolling window using the greedy search procedure outlined in Section 2.2.2. The length of the sequence of possible values for λ_0 was set to $50+1$ (including zero).

Table 2.8 presents the cross-validated MSE and MAE values for the estimated SMI models, as well as all benchmark methods, across forecast horizons $h = 1, \ldots, 30$. The MSE and MAE values averaged over the following periods—seven days (i.e. one week), 14 days (i.e. two weeks), and 30 days (i.e. one month)—are also reported in Table 2.8. These are denoted as $h = 1{:}\,7$, $h = 1{:}\,14$, and $h = 1{:}\,30$, respectively.

Here, the *GAIM* was fitted by grouping the lags of each weather variable into a distinct group, resulting in six indices. The number of indices of the *PPR* model was set to five, matching the number of indices in the PPR model used for the initialisation of *SMI-P*. Note that the calendar variable *DOY* was excluded when estimating the *PPR* model, as including separate linear terms is not feasible in a PPR model.

According to the results, *SMI-L* demonstrates the best forecasting performance among the three estimated SMI models across the averaged horizons $h = 1{:}\,7$, $h = 1{:}\,14$, and $h = 1{:}\,30$. All three SMI models outperform both the *GAIM* and *PPR* models in terms of forecasting accuracy. However, none of the SMI models surpass the performance of the *Backward* model, which achieves the highest forecasting accuracy in this case. It is also worth noting that the *GAIM* yields the worst forecasting accuracy in this application. However, in the previous application, it achieved the second-best forecasting performance among the three benchmark models considered and outperformed both *SMI-A* and *SMI-L*.

It is worth reflecting on the differences between the SMI model and the *Backward* model, which demonstrates superior forecasting performance in both real-world forecasting problems presented above. When estimating the *Backward* model, the model structure is predetermined, with each predictor treated as an individual non-linear variable. In contrast, the SMI model adopts a more general and objective approach, where both the number of indices and the grouping of predictors within each index are automatically determined by the proposed algorithm. Consequently, the SMI model faces a more challenging estimation task, given the limited prior information available about

Table 2.8: *Cross-validated out-of-sample point forecast results for daily solar intensity. The column* Horizon *indicates the forecast horizon (h). Columns 2–7 present the MSE values, and columns 8–13 present the MAE values for the estimated models:* SMI-P*,* SMI-A*,* SMI-L*,* Backward*,* GAIM*, and* PPR*, respectively. The minimum MSE and MAE values for each forecast horizon are shown in bold.*

	MSE						MAE					
Horizon	SMI-P	SMI-A	SMI-L	Backward	GAIM	PPR	SMI-P	SMI-A	SMI-L	Backward	GAIM	PPR
h=1	2065.7	1899.5	1523.6	**1266.0**	2366.4	1483.2	34.5	33.0	32.3	**27.8**	38.2	28.1
h=2	1075.5	1433.5	**949.5**	1077.6	2285.8	1306.3	27.0	28.8	**24.4**	26.3	40.9	28.0
h=3	1622.9	1636.0	1295.7	**1153.1**	2177.8	1642.0	30.8	31.9	28.4	**27.0**	35.1	31.7
h=4	1308.7	1263.0	1281.2	**1089.0**	2158.1	1911.6	30.9	30.5	30.3	**27.2**	39.4	35.7
h=5	**908.0**	969.4	1193.8	1230.6	1967.5	1638.7	**24.1**	24.3	26.0	27.1	34.8	32.5
h=6	969.0	971.7	1018.5	**938.7**	2267.0	1721.6	24.5	25.7	25.9	**23.7**	35.1	31.6
h=7	1766.8	1267.9	1176.2	**1160.4**	1385.7	1841.6	30.6	26.6	25.7	**23.7**	28.1	32.5
h=8	1218.3	**1127.9**	1224.9	1141.6	2424.2	1977.4	26.7	26.9	27.5	**26.3**	39.6	36.1
h=9	1727.0	1361.4	**1140.2**	1288.8	2418.0	1887.9	32.0	28.9	**25.6**	28.2	38.3	33.8
h=10	1528.2	1585.9	1258.0	**1164.9**	1727.2	1821.8	30.6	30.8	26.8	**25.6**	35.2	30.0
h=11	2276.9	1524.2	1499.7	**1179.7**	2451.7	1864.8	35.2	30.3	31.1	**26.6**	38.2	33.7
h=12	1175.3	1174.6	**913.3**	1041.5	2407.7	1803.9	27.7	26.2	**23.4**	25.1	41.3	34.7
h=13	1550.6	**1301.8**	1500.0	1912.4	1993.8	2344.8	31.5	**30.2**	30.7	31.6	34.7	38.3
h=14	1294.5	1407.2	1106.9	**1103.8**	2266.0	2122.9	30.5	30.5	**26.3**	27.0	39.0	37.4
h=15	1278.9	1295.9	**1198.8**	1228.2	2075.0	1581.5	27.3	27.3	**25.2**	26.7	34.9	32.0
h=16	1684.7	1266.5	1269.6	**1088.9**	2140.8	1338.2	31.9	27.8	28.6	**25.2**	34.1	28.5
h=17	1575.6	1590.0	**1168.1**	1223.0	1562.6	2158.7	30.3	29.0	25.9	**24.7**	28.9	35.2
h=18	1299.8	1595.9	**1118.9**	1241.6	2636.4	2061.1	29.6	31.5	**26.3**	28.1	41.4	35.9
h=19	1967.1	1212.1	**1075.8**	1327.4	2369.7	2079.3	34.0	28.4	**24.6**	27.8	38.4	35.1
h=20	1369.9	1432.3	1434.6	**1180.9**	1953.1	2247.3	28.5	29.6	29.6	**26.6**	36.9	35.3
h=21	1329.3	1299.5	1721.7	**1196.7**	2488.8	2200.3	29.1	**27.1**	33.5	27.3	37.6	37.1
h=22	1412.2	**880.5**	1159.7	1195.1	2096.6	1680.0	28.4	**22.1**	26.6	27.0	37.2	31.3
h=23	2084.5	**1370.6**	1411.9	1999.7	2633.5	1886.0	33.5	29.8	**29.7**	33.0	40.8	33.2
h=24	1453.0	1539.5	**1108.6**	1117.7	2378.3	1892.6	29.8	32.0	**26.6**	27.7	40.2	35.2
h=25	1604.8	1478.0	**1011.1**	1173.0	2283.4	1915.1	30.6	29.6	**24.0**	26.1	36.0	36.2
h=26	1294.1	1189.4	1207.9	**919.9**	2407.2	1493.2	28.9	27.1	27.3	**23.7**	37.9	30.7
h=27	1730.8	1091.9	1250.6	**958.4**	1734.5	1950.7	28.6	24.4	25.8	**22.5**	31.6	33.5
h=28	2159.2	1445.3	**1314.6**	1404.2	2478.1	1910.1	33.9	29.3	**27.9**	31.0	39.5	35.3
h=29	1365.9	1316.7	**1244.0**	1251.7	2091.6	2153.0	29.5	27.9	27.9	**26.6**	35.9	35.1
h=30	1236.4	1409.7	1320.6	**1162.2**	1897.8	2405.6	28.8	28.4	27.4	**26.2**	34.6	35.3
h=1:7	1388.1	1348.7	1205.5	**1130.8**	2086.9	1649.3	28.9	28.7	27.6	**26.1**	35.9	31.4
h=1:14	1463.4	1351.7	1220.1	**1196.3**	2164.1	1812.0	29.8	28.9	27.5	**26.7**	37.0	33.1
h=1:30	1511.1	1344.6	1236.6	**1213.9**	2184.1	1877.4	30.0	28.5	27.4	**26.8**	36.8	33.6

its structure. Despite this demanding estimation problem, both *SMI-P* in the mortality forecasting application and *SMI-L* in the solar intensity forecasting application achieved forecasting performance close to that of the best-performing model.

Figures 2.10, 2.11, 2.12, and 2.13 present the index variables selected across the 55 rolling windows for the estimated *SMI-P*, *SMI-A*, *SMI-L*, and *Backward* models, respectively. In each figure, a colour gradient column indicates the frequency with which individual variables are selected throughout all windows. The corresponding Tables 2.9, 2.10, 2.11, and 2.12 report these selection frequencies for each model, providing complementary details to those shown in the figures.

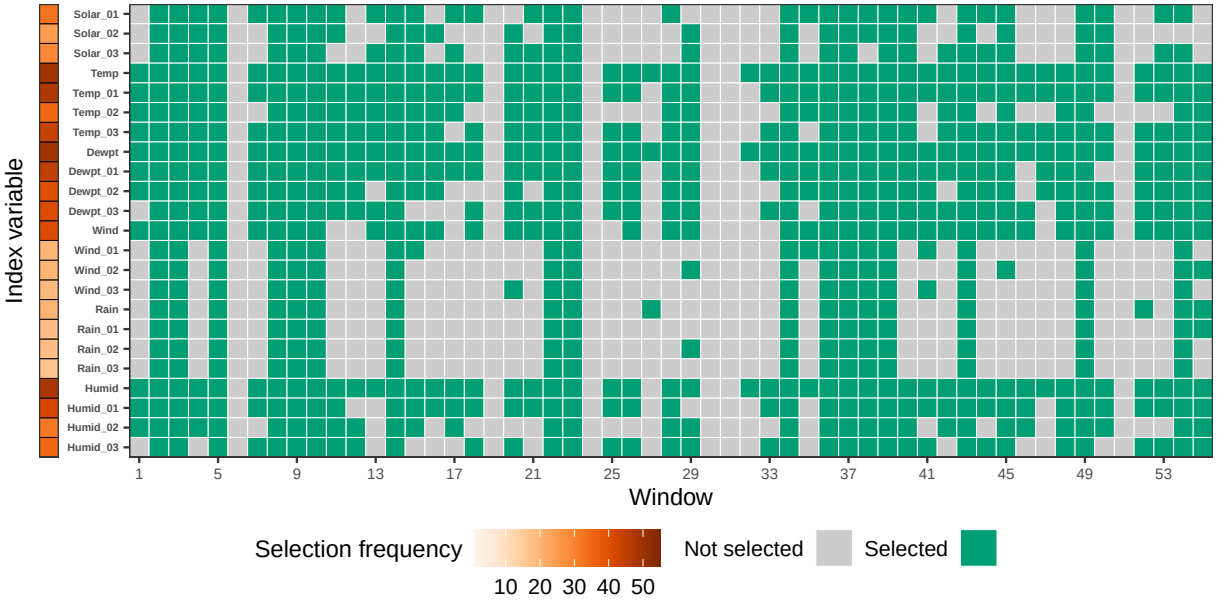


Figure 2.10: *Plot illustrating the index variables (i.e. predictors considered for inclusion in the indices) selected in the estimated* SMI-P *models for forecasting daily solar intensity across 55 rolling windows. The colour gradient column on the left-hand side of the plot indicates the selection frequency of each index variable across all rolling windows, with darker shades representing higher selection frequencies and lighter shades representing lower selection frequencies. A complementary table of selection frequencies is provided in Table 2.9.*

Table 2.9: *Selection frequency of each index variable in the estimated* SMI-P *models for forecasting daily solar intensity across 55 rolling windows. The columns labelled* Index Variable *list the variable names, while the columns labelled* Selection Frequency *show the number of rolling windows in which each variable was selected. The columns are repeated for compact presentation.*

Index Variable	Selection Frequency	Index Variable	Selection Frequency
Solar_01	33	Wind_01	20
Solar_02	25	Wind_02	20
Solar_03	29	Wind_03	19
Temp	49	Rain	20
Temp_01	47	Rain_01	18
Temp_02	35	Rain_02	18
Temp_03	44	Rain_03	17
Dewpt	49	Humid	48
Dewpt_01	45	Humid_01	42
Dewpt_02	40	Humid_02	32
Dewpt_03	41	Humid_03	35
Wind	41		

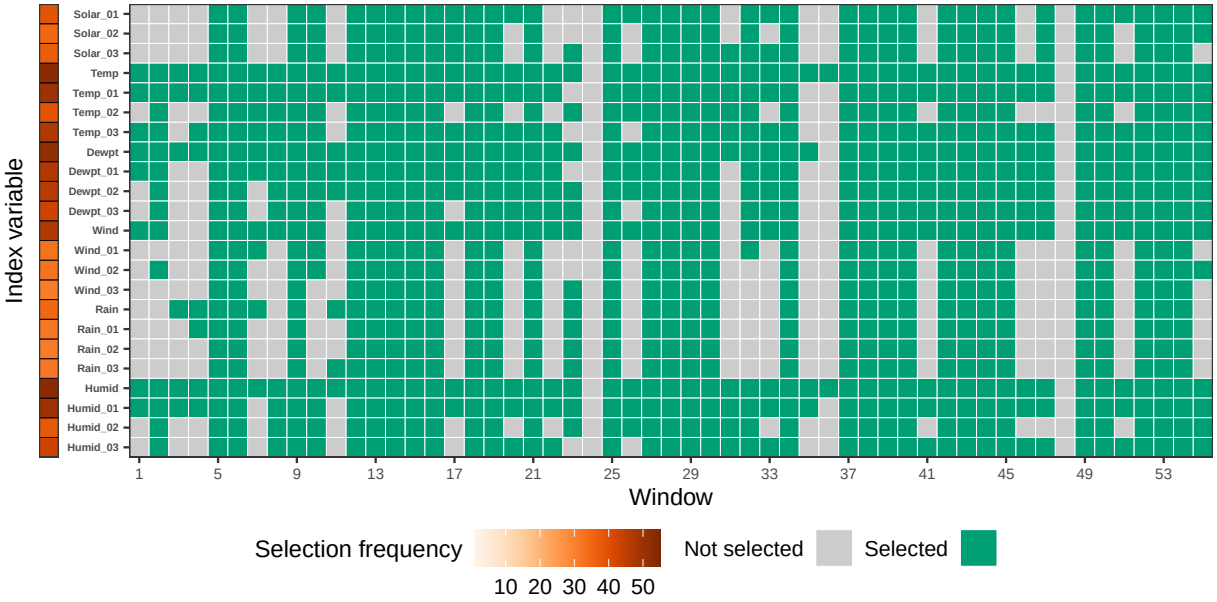


Figure 2.11: *Plot illustrating the index variables (i.e. predictors considered for inclusion in the indices) selected in the estimated* SMI-A *models for forecasting daily solar intensity across 55 rolling windows. The colour gradient column on the left-hand side of the plot indicates the selection frequency of each index variable across all rolling windows, with darker shades representing higher selection frequencies and lighter shades representing lower selection frequencies. A complementary table of selection frequencies is provided in Table 2.10.*

Table 2.10: *Selection frequency of each index variable in the estimated* SMI-A *models for forecasting daily solar intensity across 55 rolling windows. The columns labelled* Index Variable *list the variable names, while the columns labelled* Selection Frequency *show the number of rolling windows in which each variable was selected. The columns are repeated for compact presentation.*

Index Variable	Selection Frequency	Index Variable	Selection Frequency
Solar_01	39	Wind_01	33
Solar_02	35	Wind_02	33
Solar_03	37	Wind_03	31
Temp	53	Rain	35
Temp_01	50	Rain_01	32
Temp_02	39	Rain_02	31
Temp_03	47	Rain_03	32
Dewpt	52	Humid	53
Dewpt_01	47	Humid_01	50
Dewpt_02	46	Humid_02	38
Dewpt_03	43	Humid_03	43
Wind	47		

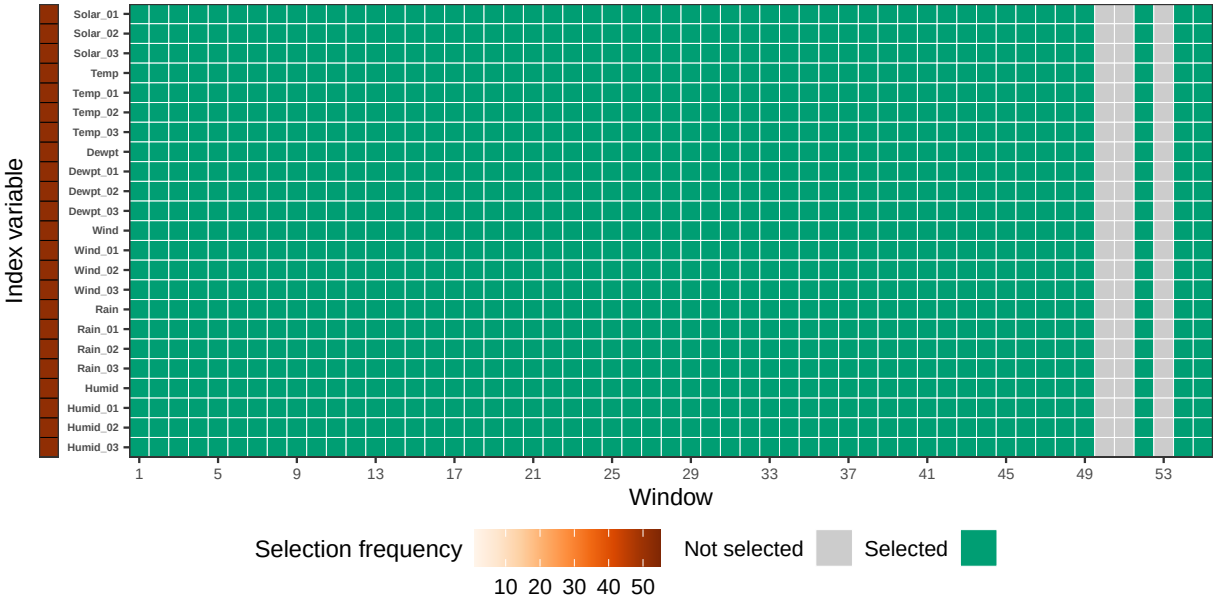


Figure 2.12: *Plot illustrating the index variables (i.e. predictors considered for inclusion in the indices) selected in the estimated* SMI-L *models for forecasting daily solar intensity across 55 rolling windows. The colour gradient column on the left-hand side of the plot indicates the selection frequency of each index variable across all rolling windows, with darker shades representing higher selection frequencies and lighter shades representing lower selection frequencies. A complementary table of selection frequencies is provided in Table 2.11.*

Table 2.11: *Selection frequency of each index variable in the estimated* SMI-L *models for forecasting daily solar intensity across 55 rolling windows. The columns labelled* Index Variable *list the variable names, while the columns labelled* Selection Frequency *show the number of rolling windows in which each variable was selected. The columns are repeated for compact presentation.*

Index Variable	Selection Frequency	Index Variable	Selection Frequency
Solar_01	52	Wind_01	52
Solar_02	52	Wind_02	52
Solar_03	52	Wind_03	52
Temp	52	Rain	52
Temp_01	52	Rain_01	52
Temp_02	52	Rain_02	52
Temp_03	52	Rain_03	52
Dewpt	52	Humid	52
Dewpt_01	52	Humid_01	52
Dewpt_02	52	Humid_02	52
Dewpt_03	52	Humid_03	52
Wind	52		

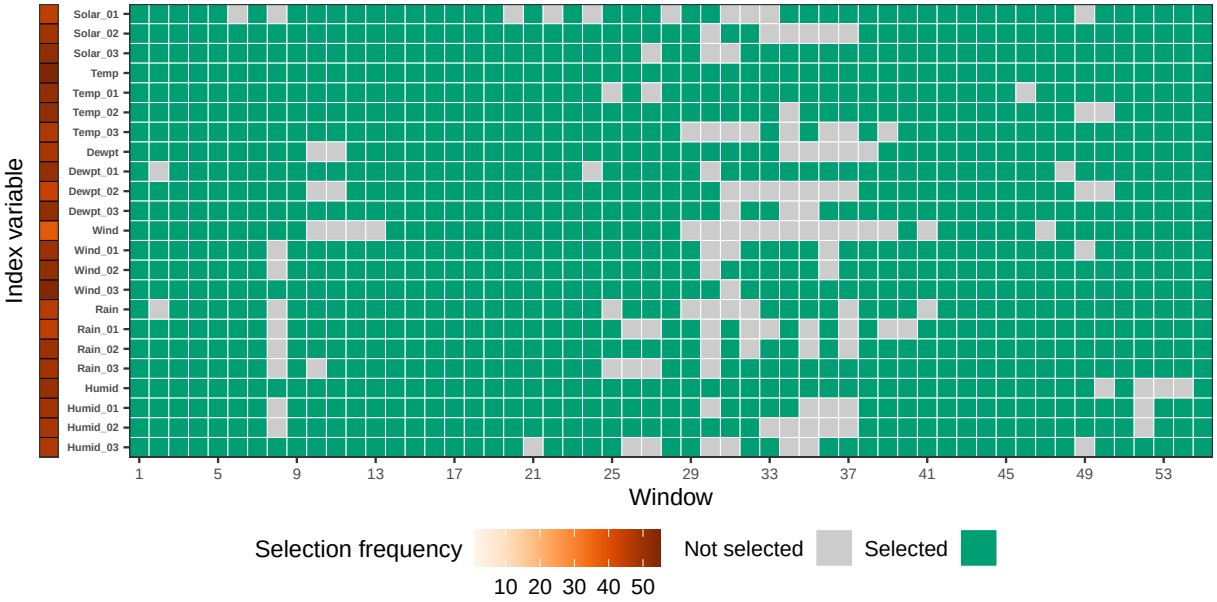


Figure 2.13: *Plot illustrating the index variables (i.e. predictors considered for inclusion in the indices) selected in the estimated* Backward *models for forecasting daily solar intensity across 55 rolling windows. The colour gradient column on the left-hand side of the plot indicates the selection frequency of each index variable across all rolling windows, with darker shades representing higher selection frequencies and lighter shades representing lower selection frequencies. A complementary table of selection frequencies is provided in Table 2.12.*

Table 2.12: *Selection frequency of each index variable in the estimated* Backward *models for forecasting daily solar intensity across 55 rolling windows. The columns labelled* Index Variable *list the variable names, while the columns labelled* Selection Frequency *show the number of rolling windows in which each variable was selected. The columns are repeated for compact presentation.*

Index Variable	Selection Frequency	Index Variable	Selection Frequency
Solar_01	45	Wind_01	50
Solar_02	49	Wind_02	52
Solar_03	52	Wind_03	54
Temp	55	Rain	46
Temp_01	52	Rain_01	45
Temp_02	52	Rain_02	50
Temp_03	47	Rain_03	49
Dewpt	48	Humid	51
Dewpt_01	51	Humid_01	49
Dewpt_02	44	Humid_02	48
Dewpt_03	52	Humid_03	47
Wind	38		

Figures 2.10, 2.11, and 2.12 show that, unlike in the previous application—where *SMI-L* yielded the sparsest results among the three estimated SMI models across many rolling windows—*SMI-L*

in this case retained all index variables except in three rolling windows. In those three windows, the model dropped all index variables and was estimated using only the Fourier terms representing the *DOY* effect, which enter the model as separate linear predictors. Further investigation is needed to determine the precise reason for this unexpected behaviour. Both *SMI-P* and *SMI-A*, however, produced comparatively sparser models across several rolling windows relative to *SMI-L*.

As observed in the previous application, the SMI models estimated in adjacent rolling windows occasionally display sequences of denser models that are immediately followed by sparser ones, or vice versa. This behaviour is likely due to the same factors discussed earlier. According to Figure 2.13, and consistent with the previous findings, the Backward model remains denser than both the *SMI-P* and *SMI-A* models.

In both the *SMI-P* and *SMI-A* models, temperature, dew point, and humidity lags tend to be selected with higher frequency compared with solar intensity, wind speed, and rainfall lags, the latter two generally exhibiting the lowest selection frequencies overall.

Similar to the previous application, all model estimations in the rolling window procedure were carried out on the MonARCH HPC Cluster using parallelisation. In Table 2.13, we report the approximate run times for each model for the first rolling window, obtained by running the estimations on an Apple Silicon MacBook Pro equipped with an Apple M1 Pro chip and 16 GB of memory. It should again be noted that the actual run times for each model and rolling window may vary depending on the characteristics of the data in that particular window.

Table 2.13: *Approximate run times for the estimated models for forecasting daily solar intensity, in the first rolling window.*

Estimated Model	Run Time
SMI-P	49.12m
SMI-A	1.40h
SMI-L	1.13m
Backward	50.59m
GAIM	15.40s
PPR	280ms

We observe a variation in run times similar to that of the previous application, where *SMI-A* requires the greatest amount of time, while *SMI-L* exhibits a considerably lower run time among the three estimated SMI models. However, it can be seen that the *SMI-P* and *Backward* models have recorded

nearly similar run times in this application, in contrast to the longer run time of the *Backward* model in the previous application. The *GAIM* and *PPR* models remain computationally efficient.

However, the most important point to note here is that the run times for this application are considerably lower than those in the previous application, primarily due to the smaller number of index variables used (23 index variables compared with 59 in the previous application).

2.4.3 Software

These two empirical applications were performed using the R statistical software (R Core Team 2024). We developed the `smimodel` package for R—introduced in Chapter 4 of the thesis and available at `github.com/nuwani-palihawadana/smimodel`—to implement the proposed SMI Modelling algorithm.

We used the commercial MIP solver *Gurobi* (*Gurobi Optimizer Reference Manual* 2023) to solve the MIQPs related to the proposed SMI Modelling algorithm, through the *Gurobi plug-in* (`ROI.plugin.gurobi`, Schwendinger 2023) available from the *R Optimization Infrastructure* (`ROI`, Theußl, Schwendinger & Hornik 2020; Hornik et al. 2023) package, with the parameter configurations, `MIPGap` $= 0.01$ and `TimeLimit` $= 600$ seconds, in both the empirical applications. The GAMs were fitted using the R package `mgcv` (Wood 2011, 2023).

2.5 Conclusion

In this chapter, we have presented a novel algorithm for estimating a nonparametric/semiparametric additive index model with optimal predictor selection and predictor grouping, which we refer to as a Sparse Multiple Index (SMI) Model. The SMI Modelling algorithm is an iterative procedure, developed based on mixed integer programming to solve an ℓ_0- and ℓ_2-regularised non-linear least squares optimisation problem with linear constraints.

The proposed SMI Modelling algorithm has a number of key features:

1) It performs automatic selection of both the number of indices and the predictor grouping when estimating the nonparametric additive index model. Users are allowed to input the set of predictors entering indices and a starting model (specifying index structure and a set of index coefficients) to initialise the algorithm.
2) It performs automatic variable selection, which is particularly beneficial in high-dimensional settings. These first two features contribute to an objective and principled estimation, reducing subjectivity across users.

3) It is capable of estimating a wide spectrum of models, from single index models (one index) to additive models (the number of indices equals the number of predictors entering indices). Hence, the SMI Modelling algorithm is a more general estimation tool for nonparametric additive models.
4) It provides the flexibility to include separate non-linear and linear predictors in the model that do not enter any indices, allowing the estimation of semiparametric additive models.

We demonstrated the performance of the proposed algorithm through a simulation study and two empirical applications. In the first empirical application, the SMI model estimated with "PPR" initialisation achieved the best forecasting performance among the three SMI models considered, whereas in the second empirical application, the SMI model estimated with "Linear" initialisation performed best. This suggests that the most suitable initialisation option depends on the data, and these two examples do not provide sufficient evidence to establish a universally applicable initialisation method for the SMI model across different scenarios. Therefore, as noted in Section 2.3, we encourage users to follow a trial-and-error procedure to identify the most effective initialisation option for their specific application.

Since the difficulty of specifying an initialisation that works in general is a limitation of the proposed algorithm, an interesting future research problem would be to explore the potential for determining a generalised initialisation for the SMI Modelling algorithm that works well across various applications.

This study should be viewed as a first attempt to develop a more objective methodology for variable selection and model estimation in the broader class of nonparametric additive models for forecasting. The SMI models were unable to outperform the backward elimination model in terms of forecasting accuracy, in each of the two empirical applications presented. However, one of the estimated SMI models always produced a model with forecasting performance close to that of the best model, while the other benchmark models did not perform consistently across the two applications. An important future research problem is, therefore, to evaluate the performance of the proposed SMI Modelling algorithm on different data sets with diverse properties, identifying scenarios where it outperforms the benchmark methods.

The MIQP in the algorithm is somewhat analogous to the best subset selection method frequently used in least squares problems. Thus, another limitation of the proposed algorithm is the increase in computational time as the number of predictors and indices increases. Therefore, it would be an interesting research exercise to gain further insights regarding the algorithm to see what improvements can be made to its design to reduce the computational cost in a high-dimensional context.

Chapter 3

Uncertainty Estimation for High-dimensional Nonparametric Forecasts

Point forecasts alone are considered inadequate unless they include an evaluation of their inherent uncertainty. This chapter focuses on producing interval forecasts to estimate the uncertainty of the point forecasts produced by the SMI model and other nonparametric additive models discussed in Chapter 2. A novel methodology for constructing prediction intervals named ***C***onformal ***B***ootstrap (***CB***) is introduced, a natural combination of block bootstrap and classical split conformal prediction. The CB prediction intervals are compared with those produced by block bootstrap and existing conformal prediction methodologies. This comparison is conducted within a generic online learning framework, as proposed by Wang & Hyndman (2024), through empirical applications to forecast heat exposure-related daily mortality and daily solar intensity.

3.1 Introduction

A point forecast is often considered insufficient without an accompanying assessment of the uncertainty it entails. *Prediction intervals* are commonly used to quantify this uncertainty. As Chatfield (1993) emphasised, it is important to provide interval forecasts (prediction intervals) alongside—or in place of—point forecasts. Prediction intervals allow us to evaluate future uncertainty, develop diverse plans for the range of possible future outcomes, and most importantly, perform careful and rigorous comparisons among different forecasting models applicable to a particular forecasting problem (Chatfield 1993). Hence, presenting prediction intervals is pivotal for planning and management in most of the fields, and warrants more attention.

Let y denote a time series $y_1, \dots, y_T$. Suppose we are interested in estimating the uncertainty associated with the h-step-ahead point forecast $\hat{y}_{t+h|t}$ for the future observation y_{t+h}, conditional on the observations up to time t. Assuming that the distribution of future observations, conditional on the observations up to time t, is normal, a theoretical $100(1-\alpha)\%$ prediction interval is given by

$$\hat{y}_{t+h|t} \pm z_{\alpha/2} \times \hat{\sigma}_h,$$

where $\hat{\sigma}_h$ is an estimate of the standard deviation of the h-step forecast distribution, and $z_{\alpha/2}$ denotes the $\alpha/2$ quantile of the standard normal distribution.

The main issue with theoretical prediction intervals is that they are particularly difficult to evaluate in real-world applications—especially in multivariate models based on multiple equations or incorporating non-linear relationships among variables (Chatfield 1993)—due to the difficulty or impossibility of analytically calculating h-step forecast variances for $h > 1$. Consequently, various empirical, simulation-based, and resampling methodologies have been developed over time to construct prediction intervals without relying on distributional assumptions. For comprehensive discussions of such procedures, refer to Chatfield (1993), Tian, Nordman & Meeker (2022) and Dewolf, Baets & Waegeman (2023).

In this thesis, we focus on nonparametric additive index models for time series forecasting, which often produce serially correlated forecast errors, making theoretical prediction intervals difficult to compute.

Therefore, in this chapter, we first consider *Block Bootstrapping (BB)*, the well-known resampling procedure for time series (Politis 2003), to construct prediction intervals. Here, we adapt the generic online learning framework proposed by Wang & Hyndman (2024) to introduce a time series cross-validation algorithm for block bootstrapping. This results in multiple prediction intervals that can be utilised to investigate the actual mean coverage of the estimated intervals, noting that the actual coverage may differ from the nominal level.

Within the class of nonparametric additive index models, our primary focus is the SMI model introduced in Chapter 2. Conventional approaches for constructing prediction intervals, such as block bootstrapping, are likely to underestimate the intervals due to the model's built-in variable selection procedure. To overcome this limitation, we also examine *Conformal Prediction (CP)* (Vovk, Gammerman & Shafer 2005), a distribution-free method that provides theoretical coverage guarantees under the assumption of exchangeable data. Although time series data inherently violate the exchangeability assumption—rendering traditional CP methods theoretically invalid in such settings—we aim to

evaluate their empirical performance in this context. In particular, we explore whether these methods produce well-calibrated intervals that compensate for the underestimation commonly observed in models involving variable selection. This investigation is conducted within the online learning framework mentioned earlier (Wang & Hyndman 2024), employing three existing CP techniques to construct multiple prediction intervals.

Since block bootstrapping involves resampling blocks of *in-sample* residuals to construct prediction intervals, it is capable of capturing any remaining serial correlation in the residuals. However, block bootstrap is unable to capture the additional uncertainty introduced by forecasting predictor variables in *ex-ante forecasting*, due to the use of in-sample residuals. On the other hand, split conformal prediction uses a separate calibration test (non-overlapping with the training set) to obtain *out-of-sample* errors, which are used to generate prediction intervals. Using out-of-sample forecast errors allows split conformal prediction to capture the uncertainty introduced in ex-ante forecasting, assuming future forecast errors for both the forecast variable and the predictors are similar to past errors. However, it is difficult to account for the autocorrelation inherent in time series data through classical split conformal prediction.

Therefore, we propose a novel methodology for constructing prediction intervals, named ***Conformal Bootstrap (CB)***, a natural integration of *block bootstrap* and *split conformal prediction*. In the proposed CB methodology, we first obtain out-of-sample forecast errors following the principles of split conformal prediction. Next, in the case of autoregressive forecasting—where lagged responses are used as predictors—we apply block bootstrap resampling to the one-step-ahead forecast errors. Conversely, for non-autoregressive forecasting—where no response lags are included as predictors—we perform bootstrapping by resampling the multi-step forecast errors at each time point as a whole. These bootstrapped forecast errors are then used to generate possible future sample paths, from which prediction intervals are constructed for the test set. This approach allows CB to account for both the inherent temporal correlation in multi-step forecasts and the uncertainty arising from ex-ante forecasting, effectively combining the strengths of block bootstrap and conformal prediction.

The organisation of the rest of the chapter is as follows. In Section 3.2, we discuss the BB and CP methods used in this chapter to construct prediction intervals for the nonparametric additive models of interest, while providing concise, but comprehensive algorithms for each method. The proposed method, *Conformal Bootstrap*, is introduced in Section 3.3 as a novel approach for constructing prediction intervals. Section 3.4 presents and evaluates the prediction intervals constructed using the different methods employed, for the real-world problems of forecasting heat exposure-related daily mortality and daily solar intensity. Final remarks are presented in Section 3.5.

3.2 Methodology

3.2.1 Preliminaries

Let y denote the time series $y_1, \ldots, y_T$, and let our forecasting model be of the form

$$y_t = f(\boldsymbol{a}_t, \boldsymbol{a}_{t-1}, \ldots, \boldsymbol{a}_{t-s}, y_{t-1}, \ldots, y_{t-k}) + \varepsilon_t, \tag{3.1}$$

where y_t is the observed value of the response variable at time t, f is an arbitrary function, $\boldsymbol{a}_t$ is a vector of exogenous variables at time t, and the error term ε_t is stationary with mean zero and constant variance σ^2.

Note that several lagged values of y_t along with the lagged exogenous variables are included in the model to partially account for the serial correlation present in the data.

We also define $\boldsymbol{z}_t = (y_t, \boldsymbol{x}_t)$, to be the observation at time t, where $\boldsymbol{x}_t = (\boldsymbol{a}_t, \boldsymbol{a}_{t-1}, \ldots, \boldsymbol{a}_{t-s}, y_{t-1}, \ldots, y_{t-k})$ is a vector that contains all the predictors of interest at time t.

We estimate the forecasting model (Equation 3.1) using a training set $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{tr}\}$. Let the estimated model be

$$\hat{y}_t = \hat{f}(\boldsymbol{x}_t), \tag{3.2}$$

where $\hat{y}_t$ is the forecast for y_t obtained through the estimated model, and $\hat{f}$ is the estimated forecasting function. Then the model residuals are given by $e_t = y_t - \hat{y}_t$.

To test the coverage of the derived prediction intervals from various methods, we calculate the average coverage of multiple prediction intervals constructed under a generic online learning framework as proposed by Wang & Hyndman (2024).

3.2.2 Prediction intervals through block bootstrap

Here, we use the standard *block bootstrap* method for resampling residuals.

First, we estimate the forecasting model on the training set, $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{tr}\}$. Next, we derive the series of in-sample residuals, $e_1, \ldots, e_{tr}$. We then randomly resample segments (or blocks) of residuals and join them together to construct new simulated residual series.

If the residual series is identified as non-seasonal, we use the autocorrelation function (ACF) to determine an appropriate block size. Specifically, we identify the last significant lag of the ACF, subject to a maximum number of lags considered, denoted `lag.max`. If the i^{th} lag is the last significant lag within this limit, the block size is set to i. This helps to preserve remaining autocorrelation in the residual series.

If the residual series is seasonal, we apply the single season block bootstrap method (Hyndman & Fan 2010; Politis 2001), where the length of each block is chosen to be a multiple of the seasonal period of the series.

The complete workflow to obtain prediction intervals for the test set via the block bootstrap is outlined below.

Block bootstrap with time series cross-validation forecasting

1. Sequentially split the data into an initial training window, $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{tr}\}$, and an initial test window, $\{\boldsymbol{z}_{tr+1}, \ldots, \boldsymbol{z}_{tr+H}\}$, where H is the forecast horizon of interest.
2. Train a forecasting model on the training window.
3. Obtain the series of in-sample residuals, $e_1, \ldots, e_{tr}$.
4. Perform block bootstrapping by randomly resampling blocks of a selected length from the residuals series, and generate a desired number of bootstrapped series of residuals.
5. Obtain H-step-ahead simulated future values, $\hat{y}_{tr+1|tr}, \ldots, \hat{y}_{tr+H|tr}$, for each bootstrapped series.

- ***If no lagged responses were used as predictors in the forecasting model:*** Add first H elements of each bootstrapped residuals series to the H-step-ahead point forecasts obtained to derive simulated future sample paths.
- ***If lagged responses were used as predictors in the forecasting model:*** Perform recursive (autoregressive) forecasting.

    i. Obtain the one-step-ahead point forecast.
    ii. Add the first element of each bootstrapped series to the one-step-ahead forecast to obtain one-step-ahead simulated future values.
    iii. Use constructed one-step-ahead simulated future values as appropriate lagged responses for the next observation, and obtain two-step-ahead point forecasts.
    iv. Add the second element of each bootstrapped series to two-step-ahead point forecasts to obtain two-step-ahead simulated future values.
    v. Repeat the process iteratively until the desired forecasting horizon H is reached.

6. Calculate the required quantiles of the sets of simulated future values at each forecast horizon to construct prediction intervals based on the desired nominal coverage (confidence) level.
7. Iteratively roll the training window forward by one observation, and then repeat steps 2–6 until prediction intervals have been constructed for the entire test set.

The sequential data-splitting process corresponding to the above time series cross-validation procedure for the BB method is illustrated in Figure 3.1.

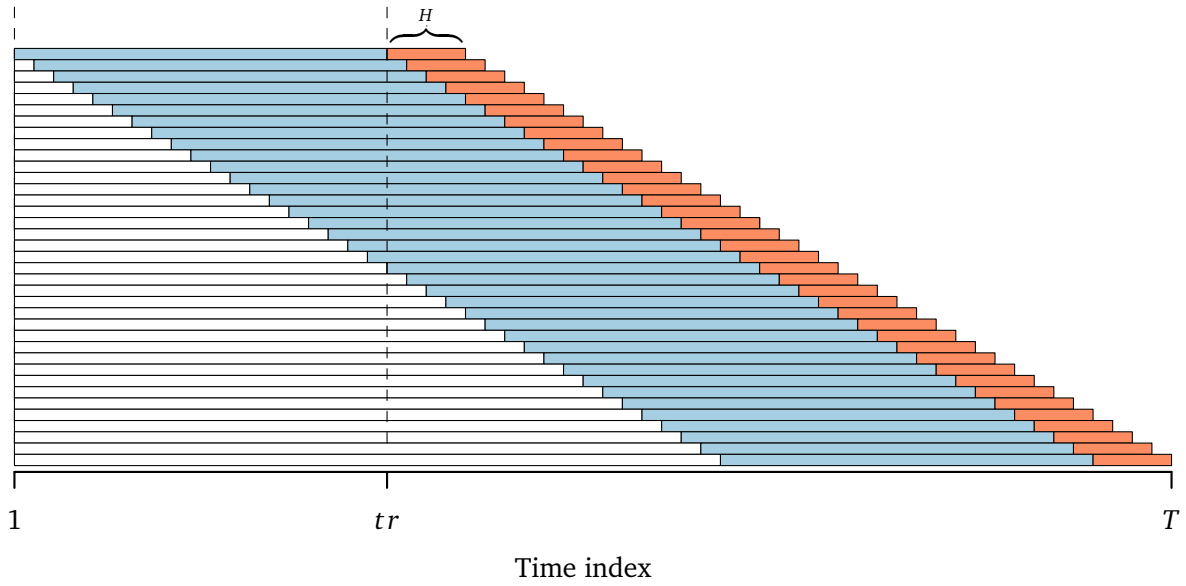


Figure 3.1: *Diagram of the time series cross-validation procedure for BB method. White: unused data; light blue: training data; orange: forecasts in test set.*

A single iteration of the BB method described above is presented as an algorithm in Algorithm 1.

3.2.3 Prediction intervals through conformal prediction

Here we use three existing conformal prediction methods: Split Conformal Prediction (SCP), Weighted Split Conformal Prediction (WSCP) and Adaptive Conformal Prediction (ACP), for computing prediction intervals under the online learning framework proposed by Wang & Hyndman (2024). The workflow is outlined as follows.

Conformal prediction with time series cross-validation forecasting

1. Sequentially split the data into an initial training window, $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{tr}\}$, an initial calibration window, $\{\boldsymbol{z}_{tr+1}, \ldots, \boldsymbol{z}_{tr+tc}\}$, and an initial test window, $\{\boldsymbol{z}_{tr+tc+1}, \ldots, \boldsymbol{z}_{tr+tc+H}\}$, where H is the forecast horizon of interest.

2. Estimate the forecasting model on the initial training window. Then obtain H-step-ahead forecasts, $\hat{y}_{tr+h|tr}$, and compute the non-conformity scores as the forecast error $s_{tr+h|tr} = y_{tr+h} - \hat{y}_{tr+h|tr}$, for $h = 1, \ldots, H$.

3. Perform cross-validation forecasting. Roll the training window forward one observation at a time, while repeating step 2 until the non-conformity scores are obtained for the entire initial calibration window.

Algorithm 1 : Block bootstrap interval forecast (BB_FORECAST)

procedure BB_FORECAST(response variable data $\{y_t\}_{t=1}^{tr}$, predictor variables data $\{\boldsymbol{x}_t\}_{t=1}^{tr+H}$, forecasting model `forecastModel`, block size s, number of future sample paths to be generated `num.futures`, significance level(s) α, whether to obtain recursive forecasts or not `recursive` (default: `FALSE`), if `recursive = TRUE`: column indices of the test set to be filled with preceding forecasts (i.e. lagged responses) `colRange` (default: `NULL`))
  modelFit $\leftarrow$ `forecastModel`$(\{y_t\}_{t=1}^{tr}, \{\boldsymbol{x}_t\}_{t=1}^{tr})$
  predictions $\leftarrow$ predict(modelFit, $\{\boldsymbol{x}_t\}_{t=tr+1}^{tr+H}$, `recursive`, `colRange`)
  bootstrap $\leftarrow$ **BLOCKBOOTSTRAP**(residuals(modelFit), s, `num.futures`) ▷ Block bootstrap method; return an $H \times$ `num.futures` matrix.
  if `recursive` = FALSE **then**
    for i in 1 to H **do**
      possibleFutures$[i,]$ $\leftarrow$ predictions$[i]$ + bootstrap$[i,]$
    end for
  else
    possibleFutures $\leftarrow$ **FUTURES**(modelFit, $\{\boldsymbol{x}_t\}_{t=tr+1}^{tr+H}$, bootstrap, `colRange`) ▷ **Algorithm-2**
  end if
  lowerQ $\leftarrow \alpha/2$
  upperQ $\leftarrow 1-\alpha/2$
  for j in 1 to H **do**
    QValues $\leftarrow$ quantile(possibleFutures$[j,]$, probs = (lowerQ, upperQ))
    for k in 1 to length(α) **do**
      lower$[k][j]$ $\leftarrow$ QValues$[1 : \text{length}(\alpha)][k]$
      upper$[k][j]$ $\leftarrow$ QValues$[(\text{length}(\alpha)+1) : \text{length}(\text{QValues})][k]$
    end for
  end for
  intervals $\leftarrow$ [lower, upper]
  return intervals
end procedure

Algorithm 2 : Generating future sample paths with recursive forecasting (FUTURES)

procedure FUTURES(fitted model `fitted_model`, test set for which forecasts are required `newdata`, an nrow(`newdata`) $\times$ `num.futures` matrix of bootstrapped residual series `bootstrap`, column indices of the test set to be filled with preceding forecasts (i.e. lagged responses) `colRange`)
  futures$[1,]$ $\leftarrow$ predict(`fitted_model`, `newdata`$[1,]$) + `bootstrap`$[1,]$
  for i in 1 to length(futures$[1,]$) **do**
    newdataList$[i]$ $\leftarrow$ `newdata`
    newdataList$[i]$ $\leftarrow$ **FILL_IN**(newdataList$[i]$, `colRange`, futures$[1, i]$) ▷ Fill-in the appropriate cells of `colRange` (lagged responses) in newdataList$[i]$, using futures$[1, i]$.
    for j in 2 to (nrow(newdataList$[i]$) - 1) **do**
      futures$[j, i]$ $\leftarrow$ predict(`fitted_model`, newdataList$[i][j,]$) + `bootstrap`$[j, i]$
      newdataList$[i]$ $\leftarrow$ **FILL_IN**(newdataList$[i]$, `colRange`, futures$[j, i]$) ▷ Fill-in the appropriate cells of `colRange` (lagged responses) in newdataList$[i]$, using futures$[j, i]$.
    end for
    futures[nrow(newdataList$[i]$), i] $\leftarrow$ predict(`fitted_model`, newdataList$[i]$[nrow(newdataList$[i]$),]) + `bootstrap`[nrow(newdataList$[i]$), i]
  end for
  return futures
end procedure

4. Apply the conformal prediction method of interest: SCP, WSCP, or ACP, to compute H-step prediction intervals of a desired nominal coverage by calculating the required lower and upper quantiles of the non-conformity scores obtained on the initial calibration window in step 3.

5. Iteratively roll the initial training window and calibration window forward one observation at a time and repeat steps 2–4, until the prediction intervals are obtained for the entire test set.

The relevant sequential data-splitting process is illustrated in Figure 3.2, while a single iteration of the above procedure for obtaining CP intervals is presented in the following Algorithm 3.

Algorithm 3 : Conformal prediction interval forecast (CP_FORECAST)

1: **procedure** CP_FORECAST(response variable data $\{y_t\}_{t=1}^{tr+tc}$, predictor variables data $\{\boldsymbol{x}_t\}_{t=1}^{tr+tc+H}$, forecasting model `forecastModel`, forecast horizon H, length of the rolling training set tr, length of the rolling calibration set tc, significance level(s) α, whether to obtain recursive forecasts or not `recursive` (default: `FALSE`), if `recursive = TRUE`: column indices of the test set to be filled with preceding forecasts (i.e. lagged responses) `colRange` (default: `NULL`), other arguments passed to CP methods ...)
2: indx $\leftarrow$ seq(from $= tr$, to $= tr + tc$, by $= 1$)
3: **for** i in indx **do**
4: modelFit$[i] \leftarrow$ `forecastModel`$(\{y_t\}_{t=i-tr+1}^{i}, \{\boldsymbol{x}_t\}_{t=i-tr+1}^{i})$
5: predictions$[i,] \leftarrow$ predict(modelFit$[i]$, $\{\boldsymbol{x}_t\}_{t=i+1}^{i+H}$, `recursive`, `colRange`)
6: errors$[i,] \leftarrow \{y_t\}_{t=i+1}^{i+H} -$ predictions$[i,]$
7: **end for**
8: intervals $\leftarrow$ **SCP** or **WSCP** or **ACP** (predictions, errors, α, ...) ▷ Apply the CP method of interest.
9: **return** intervals
10: **end procedure**

3.3 Conformal bootstrap

Block bootstrapping involves resampling blocks of in-sample residuals to simulate future sample paths. By resampling blocks rather than individual residuals, this approach more effectively captures autocorrelation present in the data. However, the use of in-sample residuals hinders the ability to account for the uncertainty introduced by forecasting predictors in ex-ante forecasting.

In contrast, SCP splits the data into non-overlapping training, calibration, and test sets, where the forecasting model is trained on the training set, and non-conformity scores (forecast errors) are calculated on the calibration set, which are in turn used to derive prediction intervals for the test set. This use of out-of-sample non-conformity scores allows us to capture the uncertainty introduced in ex-ante forecasting, assuming future forecast errors for both the forecast variable and the predictors are similar to past errors. However, accounting for the autocorrelation in the data will be problematic in conformal prediction.

Hence, we propose the ***Conformal Bootstrap (CB)*** method, which integrates the key ideas of the two preceding methods. The proposed CB method first generates out-of-sample forecast errors (i.e. non-conformity scores on a calibration set) through sequential data splitting, as in SCP. These out-of-sample forecast errors are then resampled using a block bootstrap procedure. Specifically, if autoregressive forecasting is performed (i.e. when lagged responses are used as predictors), block bootstrapping is applied to the one-step forecast errors from the calibration set. Conversely, if non-autoregressive forecasting is used (i.e. when no response lags are included as predictors), bootstrapping is applied to the multi-step forecast errors at each time point, treating them as a whole. Next, these bootstrapped forecast errors are used to simulate potential future sample paths that form the basis for constructing prediction intervals for the test set. This approach helps preserve the temporal dependencies inherent in multi-step forecasts during the construction of prediction intervals.

Thus, CB allows us to account for both the temporal correlation in multi-step forecasts, and additional uncertainty brought into the process by ex-ante forecasting, exploiting the strengths of both block bootstrap and SCP. The workflow of the proposed method is as follows.

Conformal bootstrap with time series cross-validation forecasting

1. Sequentially split the data into an initial training window, $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{tr}\}$, an initial calibration window, $\{\boldsymbol{z}_{tr+1}, \ldots, \boldsymbol{z}_{tr+tc}\}$, and an initial test window, $\{\boldsymbol{z}_{tr+tc+1}, \ldots, \boldsymbol{z}_{tr+tc+H}\}$, where H is the forecast horizon of interest.

2. Estimate the forecasting model on the initial training window. Then obtain H-step-ahead forecasts, $\hat{y}_{tr+h|tr}$, and non-conformity scores, $s_{tr+h|tr} = y_{tr+h} - \hat{y}_{tr+h|tr}$, for $h = 1, \ldots, H$.

3. Perform cross-validation forecasting. Roll the training window forward one observation at a time, while repeating step 2 until the non-conformity scores are obtained for the entire initial calibration window.

4. Once multi-step-ahead forecast errors for the entire initial calibration window have been obtained, use the model estimated on the most recent training window to generate H-step-ahead simulated future values.

- ***If no response lags were used as predictors in the forecasting model:***
    i. Treat the forecasting errors $s_{t+h|t}, h = 1, \ldots, H$, at each time point $t = tr, \ldots, tr + tc - 1$ as a whole, and perform bootstrapping across time t to obtain a desired number of bootstrapped samples.

ii. Add the multi-step forecast errors corresponding to each bootstrapped sample to the point forecasts at the respective horizons in the test window, to derive simulated future sample paths.

- ***If response lags were used as predictors in the forecasting model:***
  i. Apply block bootstrapping to the one-step forecast errors from the calibration window, using a block size of H, to obtain the desired number of bootstrapped samples.
  ii. Generate forecasts for the test window step by step, recursively adding the block-bootstrapped one-step forecast errors, to construct simulated future sample paths.

5. Calculate the required quantiles of the sets of simulated future values at each forecast horizon to construct prediction intervals based on the desired nominal coverage.

6. Iteratively roll the initial training window and calibration window forward one observation at a time, and repeat steps 2–5, until the prediction intervals are obtained for the entire test set.

Figure 3.2 illustrates the sequential data-splitting process relevant to the CB method, and a single iteration of the above procedure for constructing CB intervals is detailed in Algorithm 4.

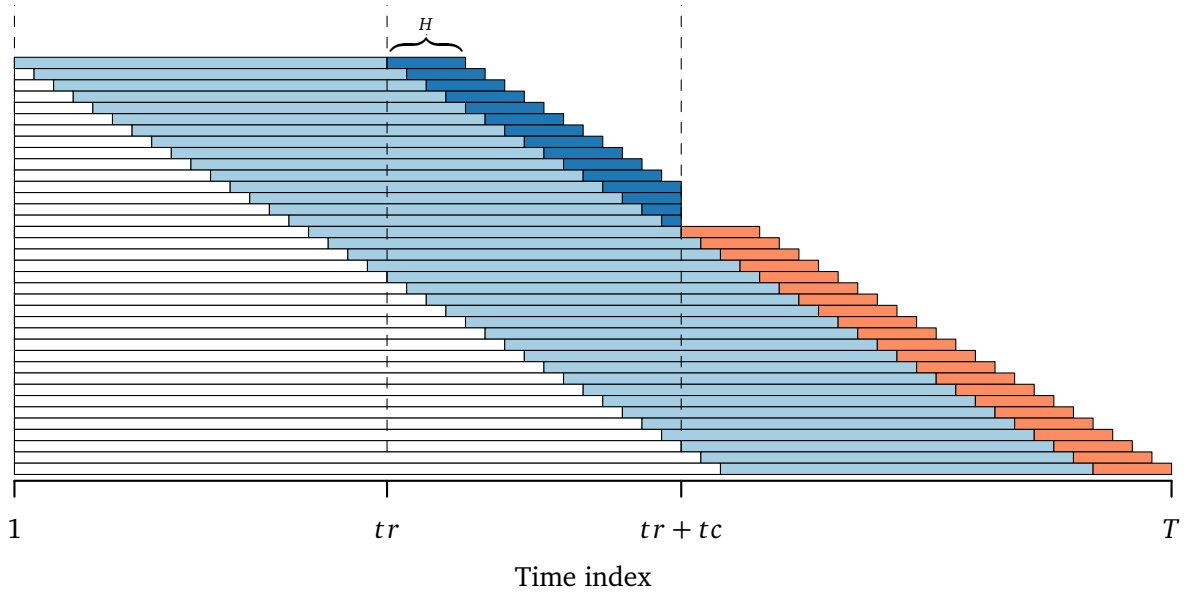


Figure 3.2: *Diagram of the time series cross-validation procedure for CP and CB methods. White: unused data; light blue: training data; dark blue: forecasts in calibration set; orange: forecasts in test set.*

Algorithm 4 : Conformal bootstrap interval forecast (CB_FORECAST)

procedure CB_FORECAST(response variable data $\{y_t\}_{t=1}^{tr+tc+H-1}$, predictor variables data $\{\boldsymbol{x}_t\}_{t=1}^{tr+tc+H}$, forecasting model `forecastModel`, forecast horizon H, length of the rolling training set tr, length of the rolling calibration set tc, number of future sample paths to be generated `num.futures`, significance level(s) α, whether to obtain recursive forecasts or not `recursive` (default: `FALSE`), if `recursive = TRUE`: column indices of the test set to be filled with preceding forecasts (i.e. lagged responses) `colRange` (default: `NULL`))
  indx ← seq(from $= tr$, to $= tr + tc$, by $= 1$)
  for i in indx **do**
    modelFit$[i]$ ← `forecastModel`$(\{y_t\}_{t=i-tr+1}^{i}, \{\boldsymbol{x}_t\}_{t=i-tr+1}^{i})$
    predictions$[i,]$ ← predict(modelFit$[i]$, $\{\boldsymbol{x}_t\}_{t=i+1}^{i+H}$, `recursive`, `colRange`)
    errors$[i,]$ ← $\{y_t\}_{t=i+1}^{i+H}$ − predictions$[i,]$
  end for
  if `recursive` = FALSE **then**
    bootstrap ← **BOOTSTRAP**(errors, `num.futures`) ▷
Bootstrap method; return an $H \times$ `num.futures` matrix. ▷
Treat $s_{t+1|t}, \ldots, s_{t+H|t}$ at each time point $t = tr, \ldots, tr + tc - 1$ as a whole, and bootstrap across t.
    for j in 1 to H **do**
      possibleFutures$[j,]$ ← predictions$[tr + tc, j]$ + bootstrap$[j,]$
    end for
  else
    bootstrap ← **BLOCKBOOTSTRAP**(errors$[\ , 1]$, `num.futures`) ▷
Block bootstrap method; return an $H \times$ `num.futures` matrix. ▷
Block bootstrap one-step forecast errors, using a block size of H.
    possibleFutures ← **FUTURES**(modelFit$[tr + tc]$, $\{\boldsymbol{x}_t\}_{t=tr+tc+1}^{tr+tc+H}$, bootstrap, `colRange`) ▷
Algorithm-2
  end if
  lowerQ ← $\alpha/2$
  upperQ ← $1 - \alpha/2$
  for k in 1 to H **do**
    QValues ← quantile(possibleFutures$[k,]$, probs = (lowerQ, upperQ))
    for l in 1 to length(α) **do**
      lower$[l][k]$ ← QValues$[1 : \text{length}(\alpha)][l]$
      upper$[l][k]$ ← QValues$[(\text{length}(\alpha) + 1) : \text{length(QValues)}][l]$
    end for
  end for
  intervals ← [lower, upper]
  return intervals
end procedure

3.4 Empirical applications

3.4.1 Forecasting daily mortality

Here, we return to the heat exposure-related daily mortality forecasting problem discussed in Chapter 2. In this chapter, we present interval forecasts to estimate the uncertainty associated with the point forecasts generated by selected nonparametric additive models.

Recap of data and forecasting models

Data: For this analysis, we consider daily mortality and heat exposure data for the metropolitan area of Montreal, Québec, Canada, from 1990 to 2014, covering the months of June, July, and August (i.e. summer). The daily all-cause mortality data were obtained from the National Institute of Public Health, Québec, while *DayMet*, a 1km × 1km gridded data set (Thornton et al. 2021), was used to extract daily temperature and humidity data (Masselot et al. 2023).

Predictors: The four main predictors considered in this empirical study are lags of daily deaths, maximum temperature, minimum temperature, and vapour pressure (to represent the level of humidity). Specifically, 14 lags of daily deaths are used as predictors in an attempt to incorporate serial correlation present in the data into the modelling process. In addition to current maximum and minimum temperatures, temperature measurements up to 14 days prior are also considered predictors in the forecasting model. We believe that 14 lags of temperature measurements are sufficient to account for the cumulative impact of current and recent past temperatures on a person's heat exposure. Similarly, the current value and 14 lags of vapour pressure are also included as predictors. Finally, a couple of calendar variables (*day of the season (DOS)* and *Year*) are incorporated into the model to capture the annual trend and seasonality.

Forecasting models: We use the SMI model (as proposed in Chapter 2) estimated with the "PPR" initialisation (*SMI-P*), a nonparametric additive model with backward elimination for variable selection (*Backward*), the Group-wise Additive Index Model (*GAIM*), and the Projection Pursuit Regression (*PPR*) model as the nonparametric models for generating point forecasts. The "PPR" initialisation was selected for the SMI model estimation as it demonstrated the best forecasting performance compared to other initialisation options in Chapter 2. For more details on the models, refer to Chapter 2.

Data split: When applying the BB, CP, and CB methods under the online learning process, the length of the rolling training window is set to 1748 observations (i.e. $tr = 1748$ in Figures 3.1 and 3.2). Hence, the initial training window contains data from 1990 to 2008.

During the online updating process, the full SMI model estimation procedure, including penalty parameter tuning, is performed only on the initial training window. First, this initial training window is split into a training set of 1656 observations (i.e. data from June 1990 to August 2007) and a validation set of 92 observations (i.e. data from June 2008 to August 2008). The SMI model is then estimated on the training set, while the validation set is used for tuning the penalty parameters. Once tuning is completed, the SMI model is re-fitted to the entire initial training window (i.e. the combined training and validation sets).

To reduce computational cost, the estimated index structure (i.e. the number of indices and the grouping of predictors within indices) obtained from the initial training window is held fixed for all subsequent rolling windows by setting the ℓ_0 penalty parameter, λ_0, to zero. The model is re-estimated at each rolling training window (of size 1748) using this fixed structure. Additionally, the ℓ_2 penalty parameter, λ_2, is also fixed at the value selected via greedy search on the initial training window.

Similarly, for the *Backward* model, estimation with variable selection is performed only on the initial training window. The training portion of this window (1656 observations) is used to fit the model, while the validation portion (92 observations) is used for variable selection. Once variable selection is completed, the *Backward* model is re-fitted to the entire initial training window (i.e. the combined training and validation sets). Subsequently, using the predictors selected from the initial training window, Generalised Additive Models (GAMs) are estimated for all subsequent rolling windows.

Since no variable or model selection is involved in estimating the *GAIM* or *PPR* models—and hence they are computationally efficient—these two models are re-estimated for each rolling training window during the online updating process.

The data from 2009 to 2014 (552 observations) are used as calibration/test sets for generating interval estimates. In the CP and CB methods, a rolling calibration window of length 300 (i.e. $tc = 300$ in Figure 3.2) is used. Since the BB method constructs prediction intervals using in-sample residuals, it does not require a separate calibration set. Therefore, the entire set of 552 observations is used as the test set. As a result, the BB method uses a longer test set compared to the CP and CB methods (see Figure 3.1) and consequently constructs a greater number of prediction intervals for each forecast horizon.

It is also important to note that CP methods construct prediction intervals separately for each forecast horizon, whereas the CB method generates multi-step prediction intervals simultaneously. Therefore, the CB method constructs a slightly higher number of intervals at each forecast horizon $h > 1$. However, in our analysis of the prediction intervals estimated using the different methods, we consider an equal number of intervals for each method, selecting those corresponding to the time indices common to all three methods.

In each method for generating prediction intervals, we consider 92-step-ahead forecasting, corresponding to 92 days in the summer season (i.e. $H = 92$ in in Figures 3.1 and 3.2). Since lags of the response variable (number of daily deaths) are used as predictors in the forecasting models, we perform recursive/autoregressive forecasting to obtain 92-step-ahead forecasts. However, as we use data only for the summer season, the time series contains gaps between years, which makes it impossible to perform autoregressive forecasting when moving from one year to the next. Therefore,

we assume that the lags of the number of daily deaths required to obtain the forecast for the first day of each summer season are known, and continue to perform autoregressive forecasting for the rest of the season. We also assume that the future values of the maximum and minimum temperatures and vapour pressure are known for use in forecasting; thus, this is a post hoc analysis.

Furthermore, when forecasting for the test set, as well as when generating future sample paths in the BB and CB methods, the predictor variables treated non-linearly in each of the nonparametric models are truncated to lie within the in-sample range to avoid extrapolation from splines, as discussed in Section 2.2.2. However, note that, as in Chapter 2, the two calendar variables are not truncated, as doing so would be inappropriate given the nature of these variables.

Interval forecasts

Block bootstrap:

The standard BB method is applied (using Algorithm 1 given in Section 3.2.2) to construct prediction intervals on the test set, with a nominal coverage of 95%. The block size is set to 50, the maximum of the last significant lags of the autocorrelation functions (ACF) of the in-sample residuals of the fitted models, subject to `lag.max` $= 50$. Thus, the block size is large enough to capture the dependence structure in the time series, while ensuring the comparability of the estimated prediction intervals across models.

The lower and upper bounds of the estimated prediction intervals by BB for the four nonparametric additive models considered are shown in Figure 3.3. Here, we have chosen the forecast horizons $1, 46$ and 92 for illustrative purposes.

According to Figure 3.3, the prediction intervals estimated using BB for the *SMI-P*, *Backward*, and *GAIM* models show no notable differences. However, in comparison, the intervals for the *PPR* model appear to be narrower. It is important to note that, compared to the other three nonparametric models considered here, the *PPR* model has a dense structure in which all predictor variables enter into every estimated index. This leads to overfitting in rolling training sets, resulting in lower in-sample residual variance relative to other models, and consequently producing narrower prediction intervals under the BB method.

Conformal prediction:

Three CP methods—SCP, WSCP with exponential weights, and ACP—are applied here using Algorithm 3 provided in Section 3.2.3, to construct 95% prediction intervals for the point forecasts obtained from all four estimated nonparametric additive models. In ACP, following Gibbs & Candès (2021), the step size parameter γ for updating the significance level α is set to 0.005.

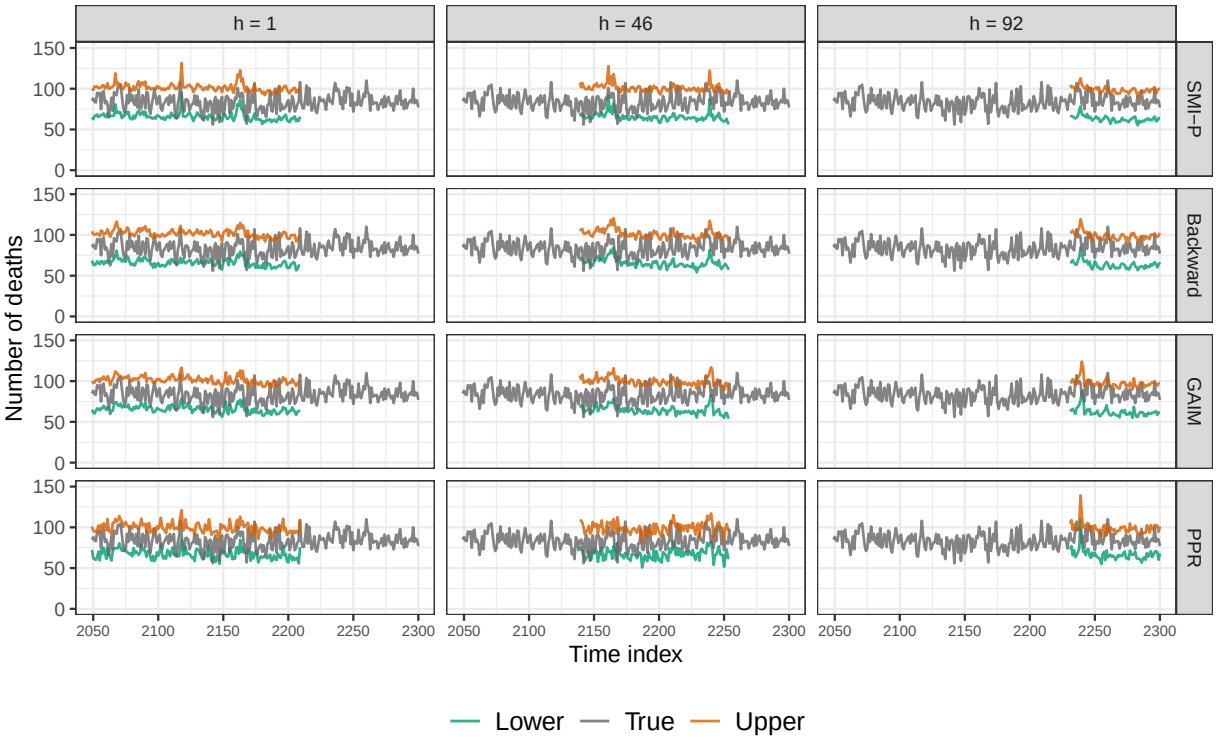


Figure 3.3: *Interval forecasts for daily mortality in summer, obtained using the BB method at forecast horizons 1, 46, and 92, for the* SMI-P*,* Backward*,* GAIM*, and* PPR *models. The lower and upper bounds of the prediction intervals are shown as green and orange lines, respectively, while the observed time series is plotted in grey.*

The lower and upper bounds of the prediction intervals estimated by SCP, WSCP, and ACP for the four nonparametric additive models considered are shown in Figures 3.4, 3.5, and 3.6, respectively.

According to Figures 3.4, 3.5, and 3.6, the prediction intervals estimated for the *SMI-P*, *Backward*, and *GAIM* models using the three CP methods are broadly similar, whereas those for the *PPR* model are noticeably wider, especially at longer forecast horizons. This behaviour can be partly explained by the overfitting nature of the PPR model, as discussed earlier. Due to in-sample overfitting, the estimated *PPR* model performs comparatively poorly on out-of-sample data, resulting in a higher out-of-sample error variance than the other models. Since CP methods rely on out-of-sample forecast errors, this is reflected in the form of wider intervals for the *PPR* model. Moreover, as we are performing autoregressive forecasting, the multi-step forecast errors accumulate over time, which further increases the width of the intervals at longer forecast horizons for the *PPR* model.

Conformal bootstrap:

We applied the proposed CB method using Algorithm 4 provided in Section 3.3 to construct 95% prediction intervals for the point forecasts obtained from all four nonparametric additive models.

The lower and upper bounds of the estimated prediction intervals by CB for the four nonparametric additive models considered are shown in Figure 3.7.

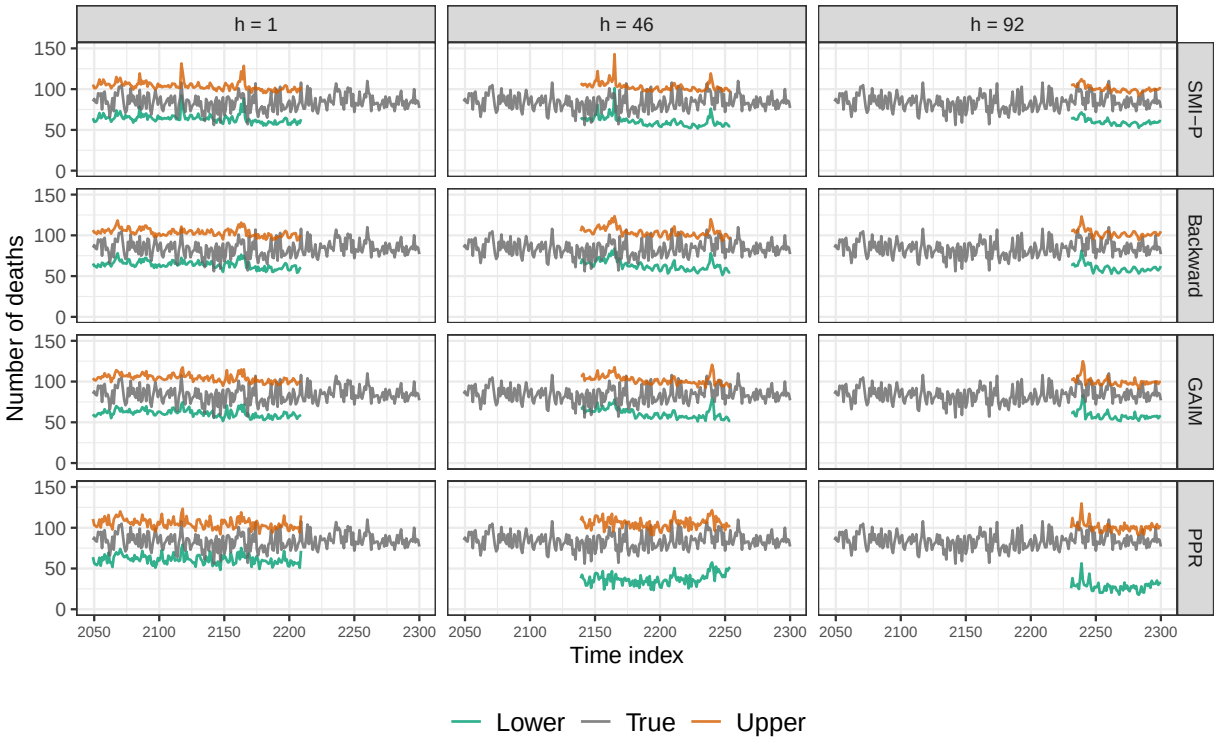


Figure 3.4: *Interval forecasts for daily mortality in summer, obtained using the SCP method at forecast horizons 1, 46, and 92, for the* SMI-P, Backward, GAIM, *and* PPR *models. The lower and upper bounds of the prediction intervals are shown as green and orange lines, respectively, while the observed time series is plotted in grey.*

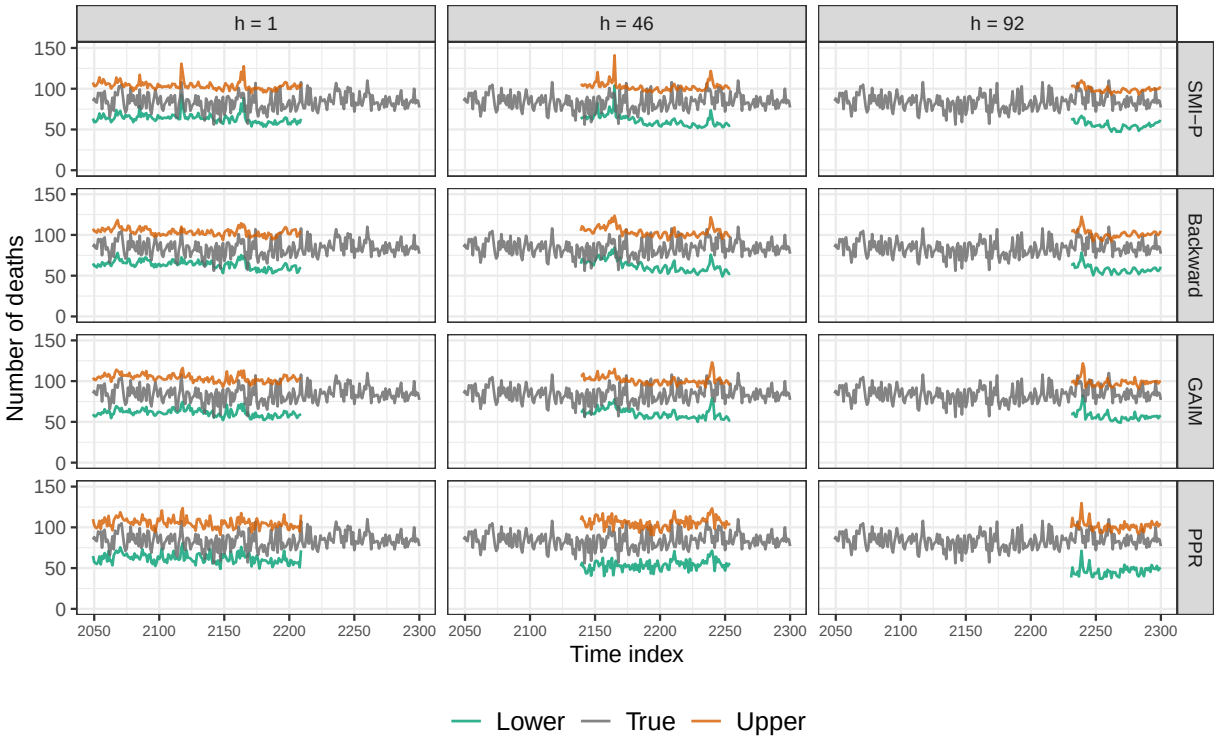


Figure 3.5: *Interval forecasts for daily mortality in summer, obtained using the WSCP method at forecast horizons 1, 46, and 92, for the* SMI-P, Backward, GAIM, *and* PPR *models. The lower and upper bounds of the prediction intervals are shown as green and orange lines, respectively, while the observed time series is plotted in grey.*

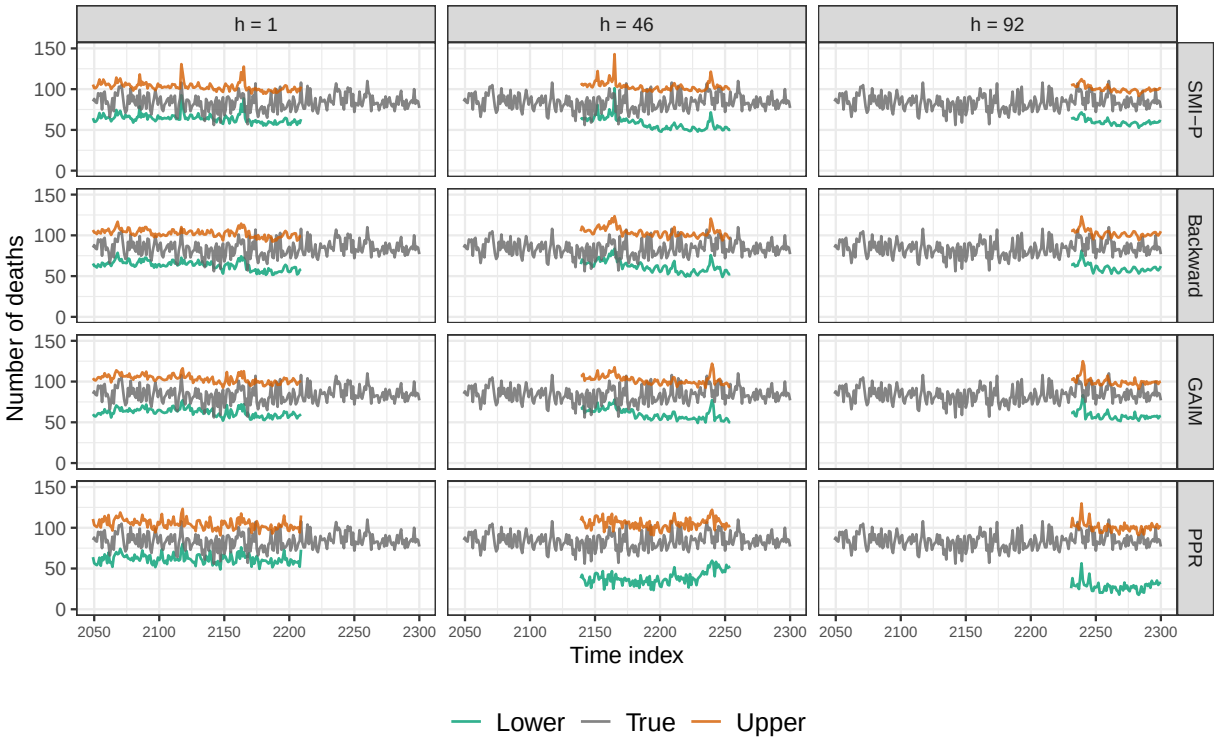


Figure 3.6: *Interval forecasts for daily mortality in summer, obtained using the ACP method at forecast horizons 1, 46, and 92, for the* SMI-P*,* Backward*,* GAIM*, and* PPR *models. The lower and upper bounds of the prediction intervals are shown as green and orange lines, respectively, while the observed time series is plotted in grey.*

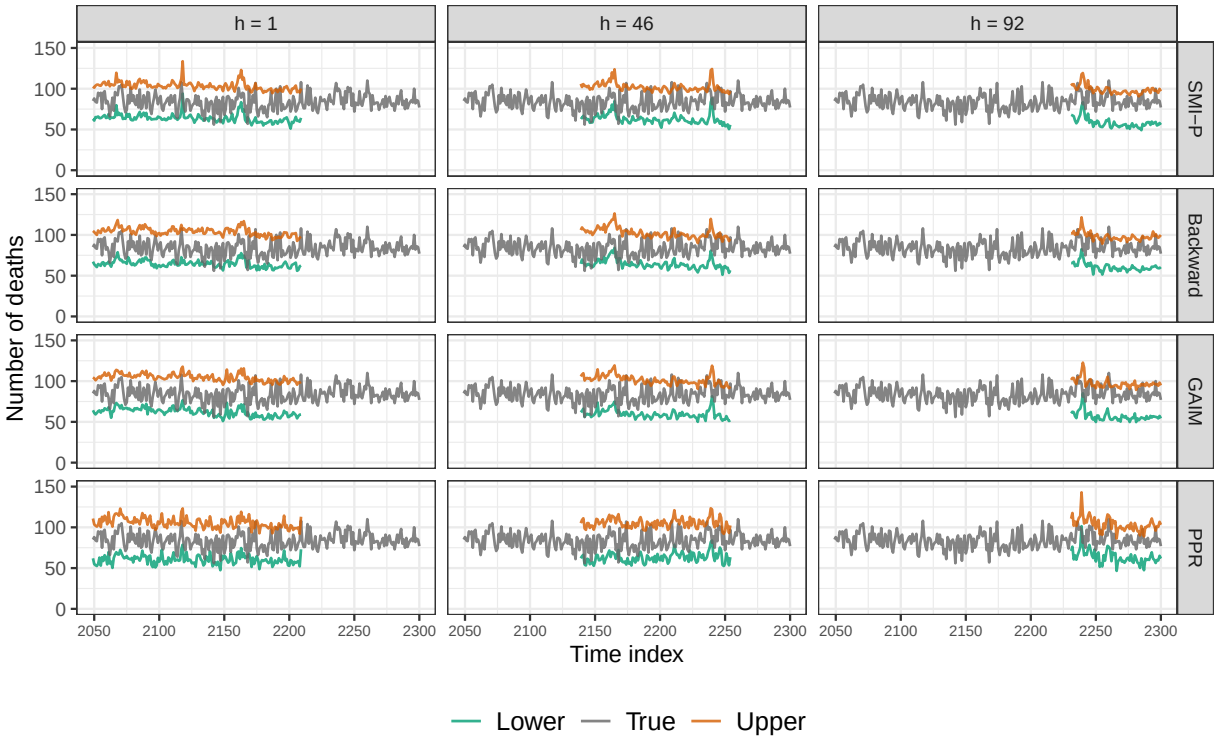


Figure 3.7: *Interval forecasts for daily mortality in summer, obtained using the CB method at forecast horizons 1, 46, and 92, for the* SMI-P*,* Backward*,* GAIM*, and* PPR *models. The lower and upper bounds of the prediction intervals are shown as green and orange lines, respectively, while the observed time series is plotted in grey.*

As shown in Figure 3.7, the prediction intervals estimated using CB for all four models do not exhibit any notable differences, except that the intervals for the *PPR* model at $h = 1$ appear slightly wider than those of the other models. This slightly wider interval for the *PPR* model at $h = 1$ can again be attributed to the same reason discussed earlier. In particular, since the CB method involves generating autoregressive sample paths (as response lags are included as predictors in forecasting models), it uses only one-step-ahead out-of-sample forecast errors, in contrast to the multi-step errors used in the CP methods discussed previously. Therefore, the accumulation of large forecast errors for the *PPR* model is not as prominent as in the CP methods, resulting in narrower CB intervals at longer forecast horizons compared to the CP intervals.

The mean (actual) coverage of the estimated prediction intervals by different methods, for all four models, across forecast horizons is plotted in Figure 3.8, where the nominal coverage of 0.95 (95%) is denoted by a dashed grey horizontal line in each plot. The mean widths of the estimated intervals are also presented in Figure 3.9 for comparison.

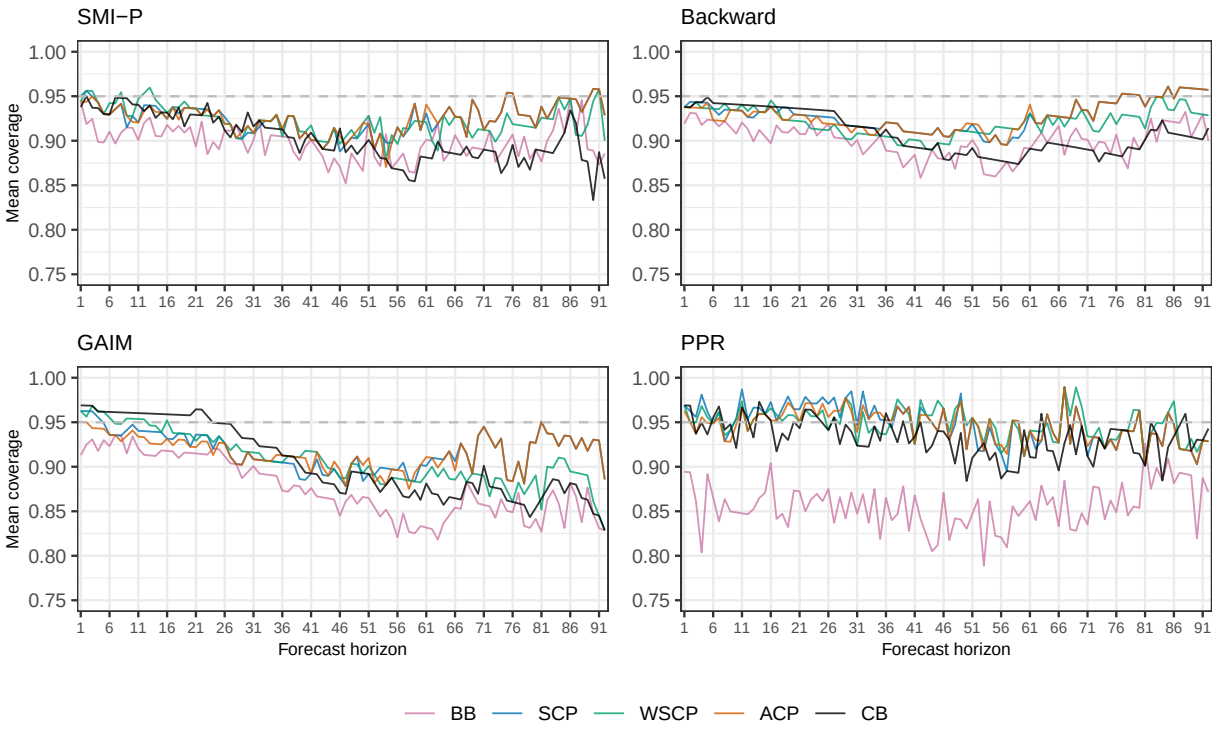


Figure 3.8: *Mean coverage of the prediction intervals for daily mortality in summer over forecast horizons for the* SMI-P*,* Backward*,* GAIM*, and* PPR *models. The nominal coverage* (0.95) *is indicated by a dashed grey horizontal line.*

Figure 3.8 shows that the prediction intervals constructed using the BB method yield mean coverage below the nominal level across all four models and forecast horizons. Moreover, the mean coverage of the BB prediction intervals is generally lower than that of the intervals produced by the three CP methods for all models and horizons. As shown in Figure 3.9, the BB intervals are also narrower than

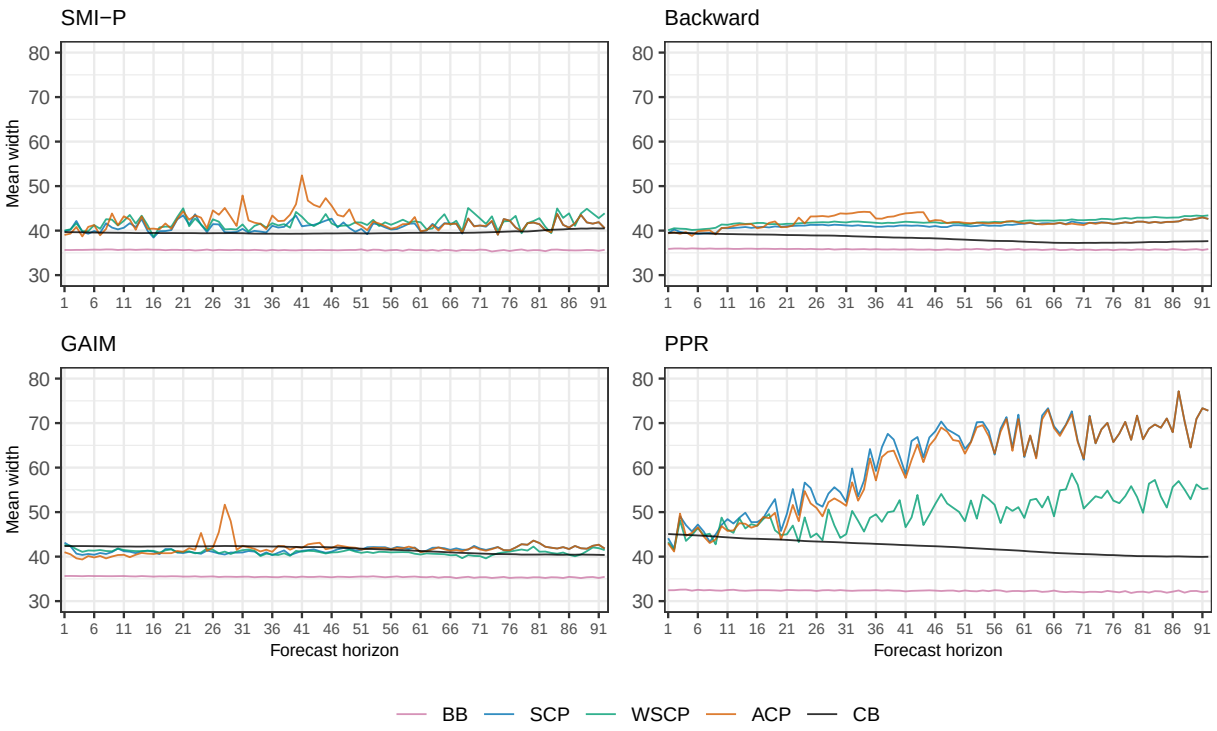


Figure 3.9: *Mean width of the prediction intervals for daily mortality in summer over forecast horizons for the* SMI-P*,* Backward*,* GAIM*, and* PPR *models.*

the CP intervals. It is important to note here that the BB method relies on in-sample residuals. In contrast, the CP methods rely on out-of-sample forecast errors, and it is typical for out-of-sample errors to have higher variance than in-sample residuals. This explains the narrower BB intervals for each forecasting model compared to the CP intervals, leading to under-coverage.

The mean coverage of the CP intervals estimated for the *PPR* model shows less variation around the nominal coverage of 0.95 compared to the other three models. Although there is a slight drop in mean coverage after $h = 50$, it is not substantial, and the coverage generally fluctuates around the nominal level. When the mean widths of the estimated CP prediction intervals in Figure 3.9 are considered, the mean widths for the *SMI-P*, *Backward*, and *GAIM* models are similar to each other. However, the mean widths of the CP intervals for the *PPR* model are larger than those of the other three models and increase with the forecast horizon. This observation is again related to the overfitting nature of the *PPR* model, which leads to out-of-sample errors with higher variance compared to the other models. These wider intervals for the *PPR* model have consequently resulted in better actual mean coverage than for the other models.

When the CB method is considered, its mean coverage is comparable to—or in some cases better than—that of all other methods for the *SMI-P*, *Backward*, and *GAIM* models, up to approximately $h = 50$. However, as the forecast horizon increases, the coverage declines and falls below that of the CP methods. While CB prediction intervals outperform BB intervals in terms of mean coverage at

shorter horizons for these three models, their coverage patterns become similar at longer horizons. For the *PPR* model, the mean coverage of the CB intervals generally resembles that of the CP methods.

The better performance of the CB method compared to the BB method—at least up to $h = 50$—can be attributed to the key methodological difference between the two: the BB method uses in-sample residuals, while the CB method uses out-of-sample (one-step-ahead) forecast errors. Although both are theoretically one-step-ahead errors, as discussed earlier, out-of-sample errors typically have a higher variance than in-sample residuals, resulting in comparatively wider intervals and consequently better coverage.

When comparing the CB and CP methods, one of the main differences is that CP methods use multi-step forecast errors, in contrast to the one-step-ahead errors used by the CB method. Although the autoregressive generation of sample paths in CB partly compensates for the error accumulation associated with multi-step forecast errors—as used in CP methods—this compensation appears insufficient at longer forecast horizons. As a result, the coverage of the CB method decreases relative to that of the CP methods.

Through our analysis, we observed that truncating predictor values to lie within the in-sample range during the generation of autoregressive sample paths in the CB method may contribute to this behaviour. It is important to clarify that in both CP and CB methods, predictor truncation occurs when generating point forecasts for the calibration set, which are then used to compute the out-of-sample forecast errors for constructing prediction intervals. However, while CP methods construct prediction intervals independently at each forecast horizon, the CB method generates possible future sample paths autoregressively across forecast horizons. In the latter case, predictor truncation is applied a second time when simulating future values at each forecast horizon. We believe this may adversely affect the width of the CB intervals by artificially narrowing them. Further research is needed to better understand this behaviour and to identify potential solutions to address this limitation.

3.4.2 Forecasting daily solar intensity

Next, we consider the daily solar intensity forecasting problem discussed in Chapter 2. Here, we present interval forecasts to estimate the uncertainty associated with the point forecasts generated by selected nonparametric additive models.

Recap of data and forecasting models

Data: For this application, we use data on solar intensity and other weather variables measured at the Davis weather station in Amherst, Massachusetts, obtained from the *UMass Trace Repository* (*The UMass Trace Repository* 2023). Five-minute interval measurements were available from 21 February

2006 to 27 February 2013, which we converted to daily data by averaging each variable over each day.

Predictors: Lags of daily solar intensity, temperature, dew point, wind speed, rain, and humidity are the six main predictor variables used in this application. Specifically, three lags of daily solar intensity are included as predictors to incorporate, at least partially, the serial correlation present in the data into the modelling process. In addition to the current values of temperature, dew point, wind speed, rain, and humidity, measurements from the previous three days for each of these weather variables are also included as predictors in the forecasting model. A calendar variable, *day of the year (DOY)*, is also incorporated into the model as a smooth function using Fourier terms, to capture annual seasonality.

Forecasting models: We use the SMI model estimated with the "Linear" initialisation (*SMI-L*), as it exhibited the best forecasting performance compared to other initialisation options in Chapter 2. As in the previous empirical application, we also use the *Backward*, *GAIM*, and *PPR* models as additional nonparametric models for generating point forecasts. For further details on these models, see Chapter 2.

Data split: When applying the BB, CP, and CB methods, the length of the rolling training window is set to 1614 observations (i.e. $tr = 1614$ in Figures 3.1 and 3.2). Hence, the initial training window contains data from 21 February 2006 to 23 July 2010.

As in the previous application, during the online updating process, the SMI model is estimated with penalty parameter tuning only on the initial training window, after which the estimated index structure is held fixed for the subsequent rolling training windows. To this end, the initial training window is split into a training set of 1569 observations (i.e. data from 21 February 2006 to 8 June 2010) and a validation set of 45 observations (i.e. data from 9 June 2010 to 23 July 2010), where the SMI model is estimated on the training set, and the penalty parameter tuning is based on the validation set MSE.

As in Chapter 2, we consider the number of Fourier term pairs (K) ranging from $K = 1$ to $K = 10$, and use the SMI model with the "Linear" initialisation to determine the number of pairs that best capture the day of the year effect, based on the validation set MSE. The SMI model (with "Linear" initialisation) estimated with $K = 9$ yields the lowest validation set MSE on the initial training window. Therefore, $K = 9$ is used as the number of Fourier term pairs when estimating all the forecasting models considered. Once penalty parameter tuning and the selection of Fourier terms are completed, the SMI model is re-fitted to the entire initial training window.

Similarly, for the *Backward* model, variable selection is performed only on the initial training set. The model is fitted on the training portion of the data, and the variable selection is based on the validation portion of the data. Once this is completed, the model is re-fitted to the entire initial training window. Then using the selected predictors, GAMs are estimated for each subsequent rolling training window. The *GAIM* and *PPR* models are re-estimated for each rolling training window in the usual way during the online updating process, as no variable or model selection is involved in their estimation.

The data from 24 July 2010 to 27 February 2013 (950 observations) are used as calibration/test sets for generating interval estimates. In the CP and CB methods, a rolling calibration window of length 500 (i.e. $tc = 500$ in Figure 3.2) is used.

In each method for generating prediction intervals, we consider 30-step-ahead forecasting (i.e. $H = 30$ in in Figures 3.1 and 3.2). Similar to the previous application, since lags of the response variable (daily solar intensity) are used as predictors in the forecasting models, we perform autoregressive forecasting to obtain these 30-step-ahead forecasts.

As noted in Chapter 2, the data set contains missing values. Therefore, during the time series cross-validation procedure, if any of the 30-step windows used to generate point forecasts on the calibration set (denoted in dark blue in Figure 3.2) contain missing values, those cross-validation iterations are omitted. This ensures that point forecasts can be generated across the full forecast horizon in every rolling window included in the analysis. Furthermore, if the first two-thirds of any rolling test window (denoted in orange in Figure 3.2) contains missing values, or if the relevant forecasting model is unavailable due to omissions in the previous cross-validation step, those test windows are excluded from the generation of future sample paths, as they are either unable to produce a sufficient number of non-missing forecasts or cannot produce forecasts at all due to the absence of a valid forecasting model.

We assume that the future values of temperature, dew point, wind speed, rain, and humidity are known for use in forecasting; thus it is a post hoc analysis. Moreover, when forecasting for the test set and generating future sample paths in the BB and CB methods, the predictor variables treated non-linearly are truncated to lie within the in-sample range. For the analysis of the prediction intervals generated by the different methods, we use the same number of intervals for each method by selecting those corresponding to the time indices shared across all three approaches.

Interval forecasts

Block bootstrap:

The standard BB method is applied to construct prediction intervals on the test set, with the nominal coverage of 95%. The block size is taken as 45, the maximum of the last significant lags of the

autocorrelation functions (ACF) of the in-sample residuals of the fitted models, subject to `lag.max` $= 50$.

The lower and upper bounds of the estimated prediction intervals by BB for the four nonparametric additive models considered are shown in Figure 3.10. Here, we have chosen the forecast horizons $1, 15$ and 30 for illustration.

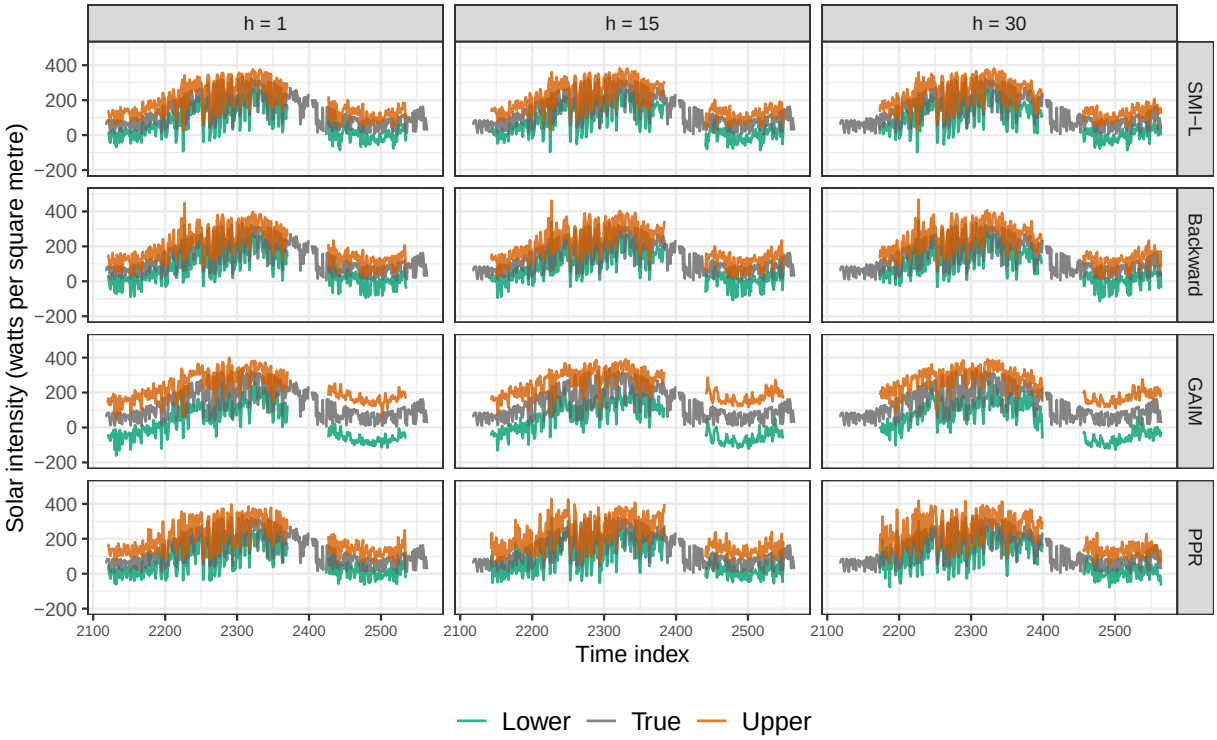


Figure 3.10: *Interval forecasts for daily solar intensity, obtained using the BB method at forecast horizons 1, 15, and 30, for the* SMI-L*,* Backward*,* GAIM*, and* PPR *models. The lower and upper bounds of the prediction intervals are shown as green and orange lines, respectively, while the observed time series is plotted in grey.*

According to Figure 3.10, the BB prediction intervals estimated for the *SMI-L*, *Backward*, and *PPR* models show no notable differences, whereas those for the *GAIM* appear noticeably wider. This can be attributed to poor in-sample fitting performance observed during the cross-validation process, where many rolling windows produced non-convergent *GAIM*s. As a result, the *GAIM* exhibited higher in-sample residual variance relative to the other three models, leading to wider BB prediction intervals.

Conformal prediction:

We applied conformal prediction methods—SCP, WSCP with exponential weights, and ACP—to construct 95% prediction intervals for all four nonparametric additive models. Similar to the previous application, the step size parameter γ for updating the significance level α in ACP is set to 0.005.

The lower and upper bounds of the estimated prediction intervals by SCP, WSCP, and ACP for the four nonparametric additive models considered are presented in Figures 3.11, 3.12, and 3.13, respectively.

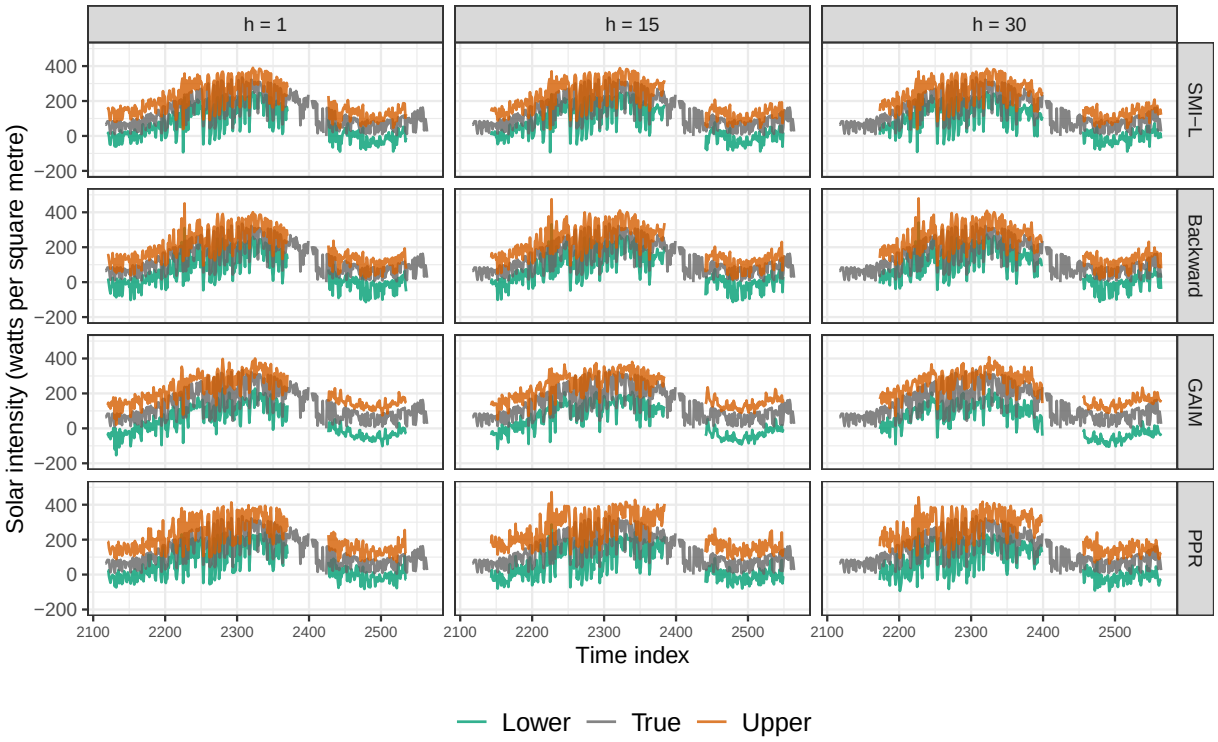


Figure 3.11: *Interval forecasts for daily solar intensity, obtained using the SCP method at forecast horizons 1, 15, and 30, for the* SMI-L*,* Backward*,* GAIM*, and* PPR *models. The lower and upper bounds of the prediction intervals are shown as green and orange lines, respectively, while the observed time series is plotted in grey.*

As shown in Figures 3.11, 3.12, and 3.13, the prediction intervals estimated using the three CP methods for the *SMI-L* and *Backward* models exhibit no notable differences. However, the intervals estimated for the *GAIM* are wider than those of the other models. The prediction intervals for the *PPR* model are also slightly wider than those for the *SMI-L* and *Backward* models, particularly towards the end of the test set, and show more fluctuation across the test period.

The wider intervals for *GAIM* are due to underfitting during estimation, which leads to poor forecasting performance and increased forecast error variance. However, the CP prediction intervals for *GAIM* remain narrower than those produced by the BB method, consistent with our finding that the in-sample residuals of *GAIM* often exhibit higher variance than its out-of-sample errors across many cross-validation steps—also a result of underfitting. The slightly wider intervals for the *PPR* model are attributed to its dense structure, which results in overfitting and, consequently, higher variance in out-of-sample forecast errors compared to the *SMI-L* and *Backward* models.

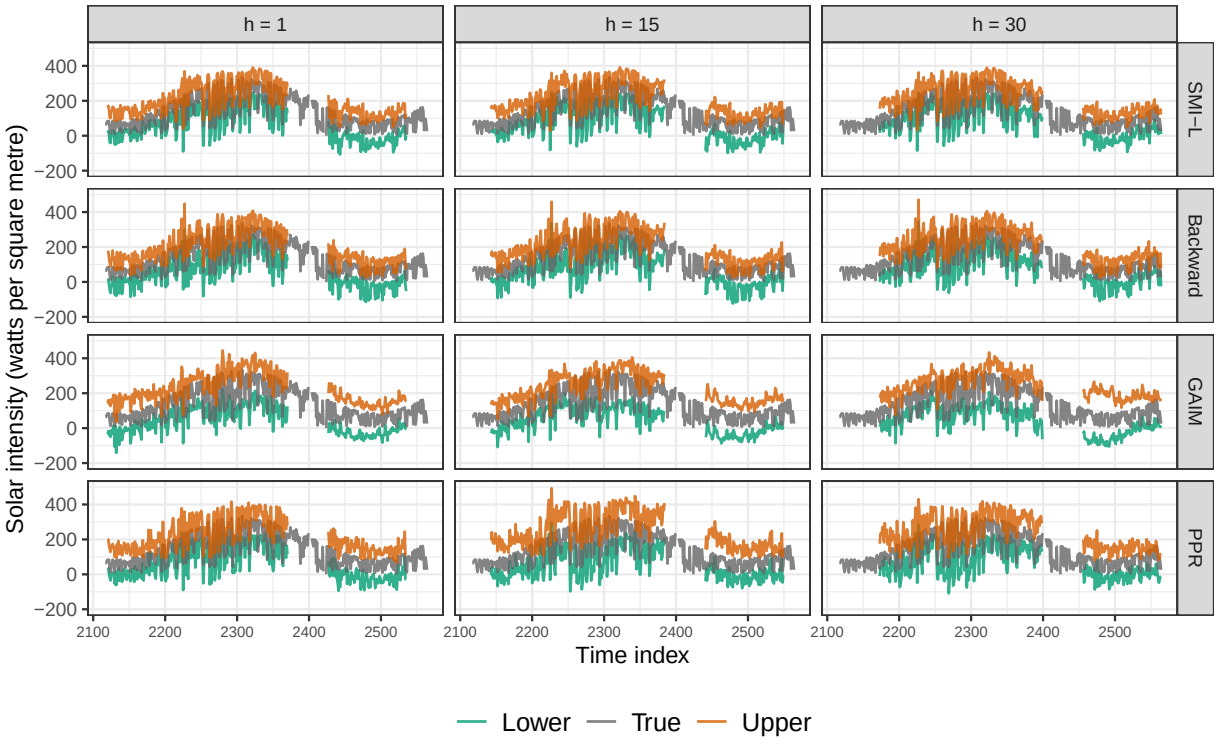


Figure 3.12: *Interval forecasts for daily solar intensity, obtained using the WSCP method at forecast horizons 1, 15, and 30, for the* SMI-L*,* Backward*,* GAIM*, and* PPR *models. The lower and upper bounds of the prediction intervals are shown as green and orange lines, respectively, while the observed time series is plotted in grey.*

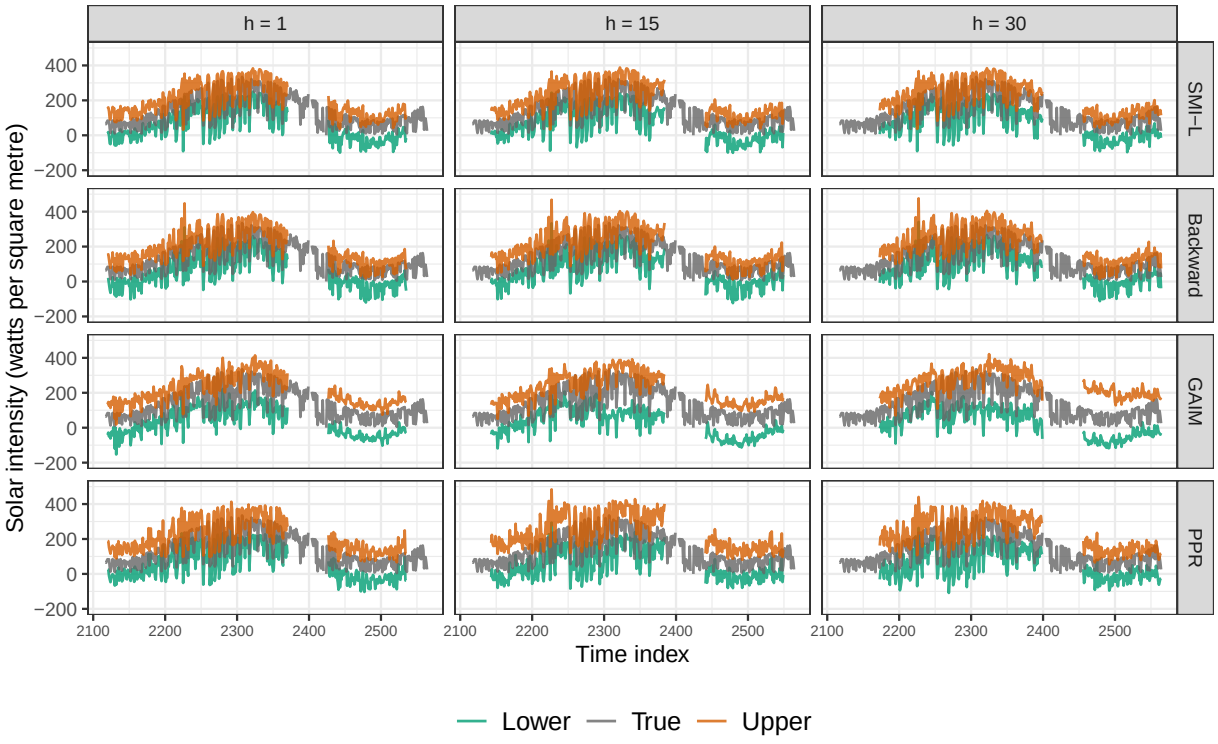


Figure 3.13: *Interval forecasts for daily solar intensity, obtained using the ACP method at forecast horizons 1, 15, and 30, for the* SMI-L*,* Backward*,* GAIM*, and* PPR *models. The lower and upper bounds of the prediction intervals are shown as green and orange lines, respectively, while the observed time series is plotted in grey.*

Conformal bootstrap:

The proposed CB method is applied here to construct 95% prediction intervals for the point forecasts obtained from all four nonparametric additive models.

The lower and upper bounds of the prediction intervals estimated by CB for the four nonparametric additive models considered are presented in Figure 3.14.

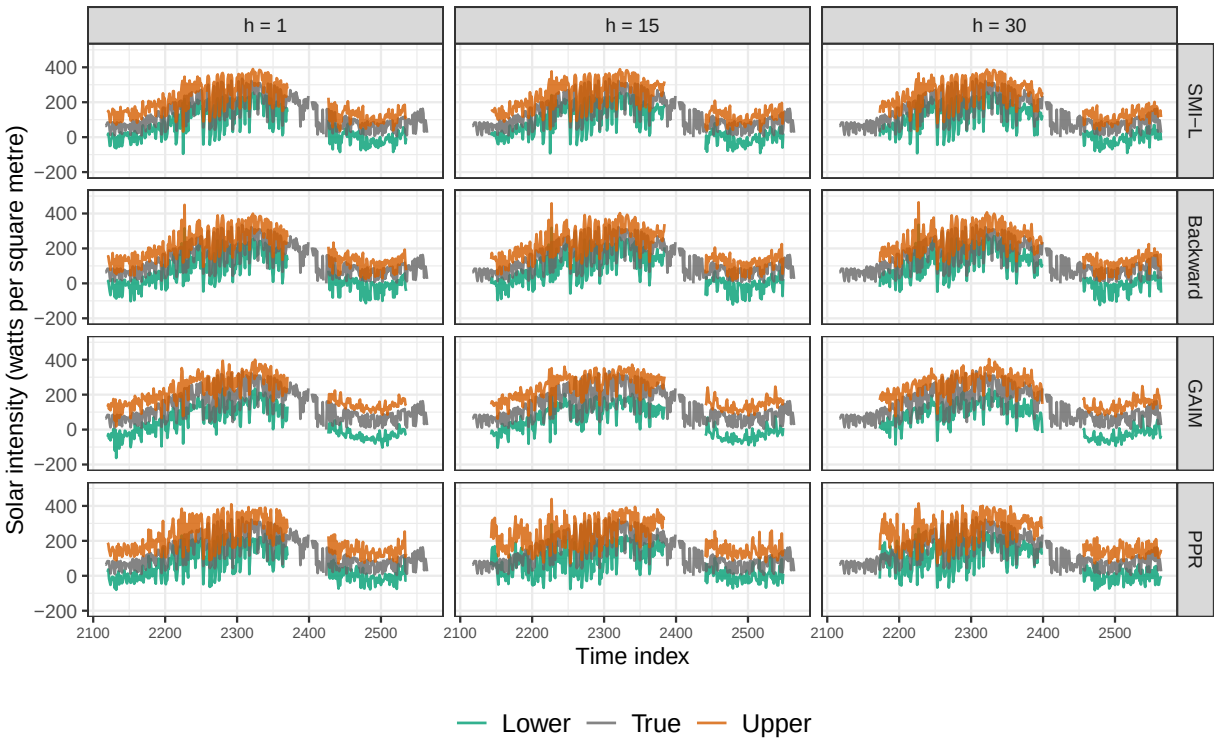


Figure 3.14: *Interval forecasts for daily solar intensity, obtained using the CB method at forecast horizons 1, 15, and 30, for the* SMI-L*,* Backward*,* GAIM*, and* PPR *models. The lower and upper bounds of the prediction intervals are shown as green and orange lines, respectively, while the observed time series is plotted in grey.*

According to Figure 3.14, similar to the previously examined methods, the intervals for *GAIM* appear slightly wider than those of the other models. However, the CB intervals for *GAIM* are narrower than those produced by the previously considered CP methods. This difference is likely due to the CB methods using only one-step-ahead forecast errors, whereas the CP methods rely on multi-step forecast errors.

The mean (actual) coverage of prediction intervals estimated by different methods, for all four estimated models, across forecast horizons, is plotted in Figure 3.15, where the nominal coverage of 0.95 (95%) is denoted using a dashed grey horizontal line in each plot. The mean widths of the estimated intervals are also presented in Figure 3.16.

According to Figure 3.15, the mean coverages of the prediction intervals estimated for *SMI-L* using the three CP methods and the CB method follow a similar pattern across forecast horizons, remaining

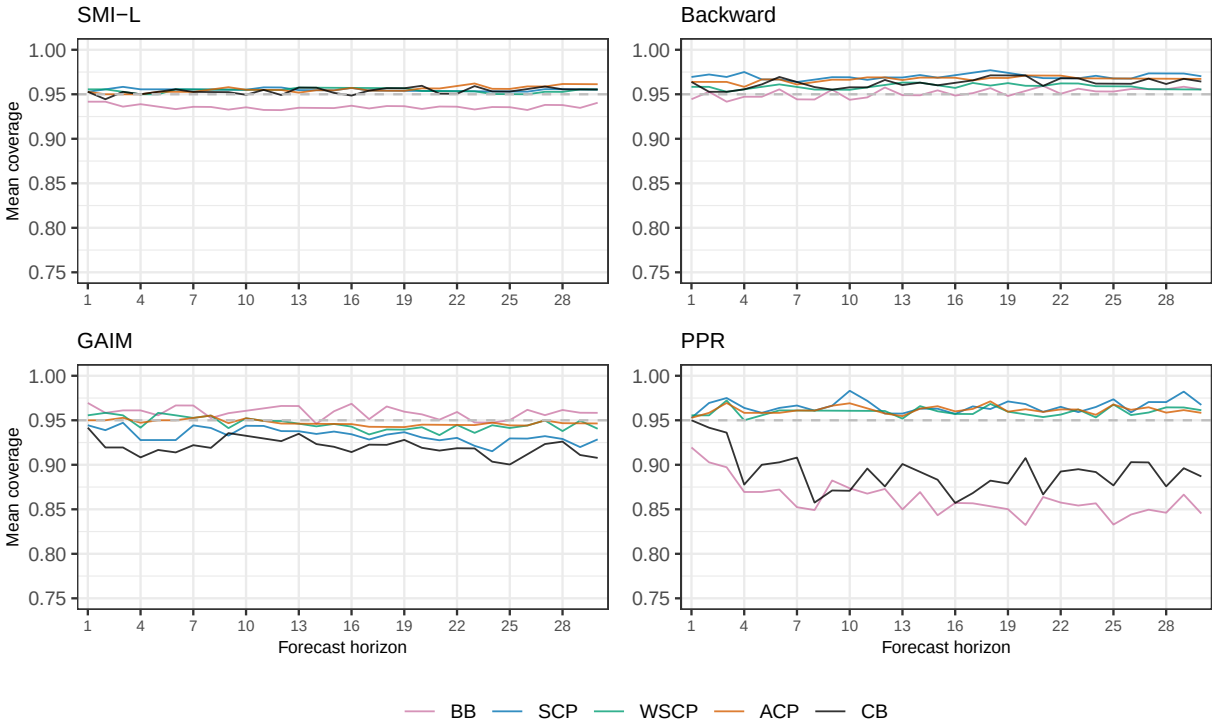


Figure 3.15: *Mean coverage of the prediction intervals for daily solar intensity over forecast horizons for the* SMI-L, Backward, GAIM, *and* PPR *models. The nominal coverage (*0.95*) is indicated by a dashed grey horizontal line.*

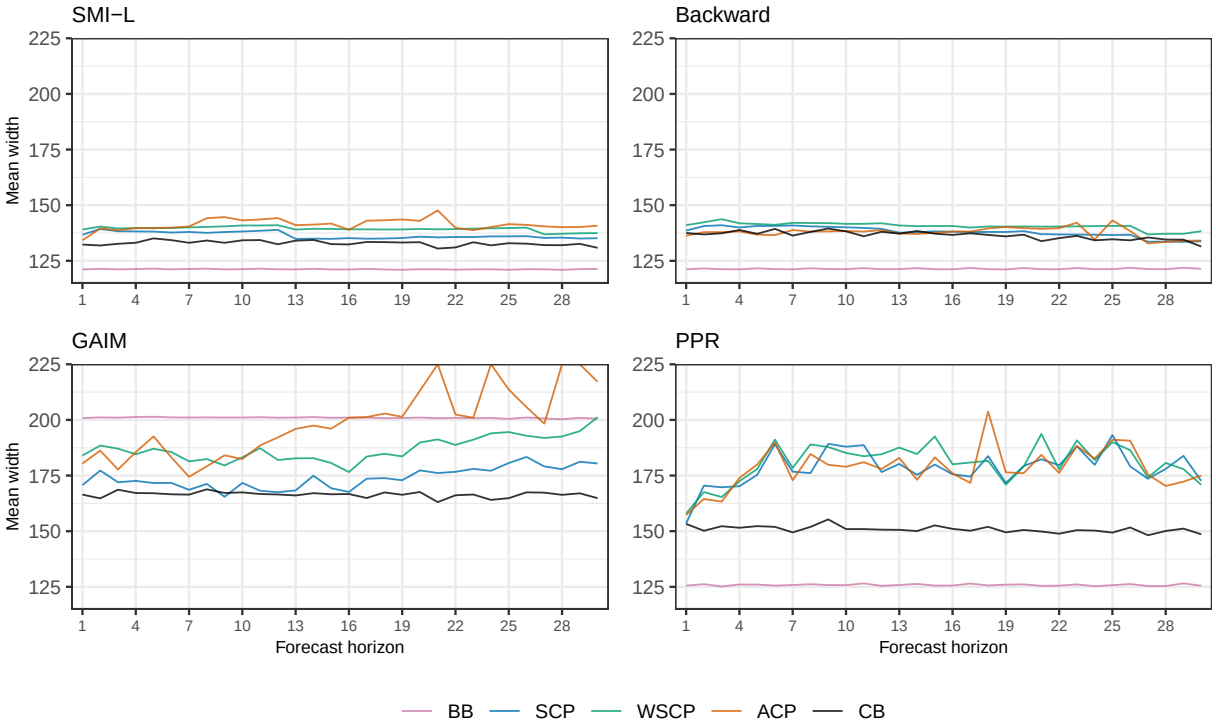


Figure 3.16: *Mean width of the prediction intervals for daily mortality in summer over forecast horizons for the* SMI-L, Backward, GAIM, *and* PPR *models.*

close to the nominal 95% level. The BB prediction intervals, however, show mean coverage below the nominal level, which is expected given that the variance of in-sample residuals is typically lower than that of out-of-sample forecast errors.

As with *SMI-L*, the mean coverages of the prediction intervals estimated for the *Backward* model using the CP and CB methods evolve similarly across forecast horizons, although in this case, they tend to slightly exceed the nominal level. The BB intervals for the *Backward* model achieve mean coverage that remains close to the nominal level across all horizons.

It is notable that the coverage performance of the CB method differs between the two empirical applications. While the CB method failed to achieve nominal coverage at longer forecast horizons for both the *SMI-L* and *Backward* models in the previous application, it performs comparably to the CP methods in the context of the current application. In this setting, using only one-step-ahead forecast errors in the CB method, as opposed to multi-step errors in the CP methods, does not reduce its coverage performance.

As discussed in the previous application, the truncation of predictors when generating future sample paths in the CB method appears to have some influence on the estimated intervals. However, this effect is not consistent and likely depends on the characteristics of the data. In some cases, it may lead to artificially narrow CB prediction intervals. In the present application, the impact of this effect seems less pronounced than in the previous one. A detailed investigation into the impact of predictor truncation on the coverage performance of the CB method is left for future research.

In the case of *GAIM*, the CP and CB methods exhibit mean coverages lower than that of the BB method. The mean coverage for BB fluctuates around the nominal level, with slight over-coverage observed at certain forecast horizons. Such a result deviates from the general pattern observed for the other models, where the CP and CB methods typically achieved better coverage than the BB method. This different behaviour can be attributed to the issue discussed earlier: due to the poor fit of the *GAIM*, its in-sample residuals exhibit greater variance than the out-of-sample forecast errors. As a result, the BB method produces much wider intervals for *GAIM* (as clearly shown in Figure 3.16), which in turn leads to better coverage. The lower coverage of the CB intervals compared to the CP intervals for *GAIM* may result from a combination of using only one-step-ahead forecast errors and the truncation of predictor variables, as previously discussed.

Notably, as shown in Figure 3.16, the ACP method produces infinite-width intervals for *GAIM* at several forecast horizons—an issue known to occur with this method. This is because ACP does not enforce any measure to prevent the significance level α_t at time t from reaching $\alpha_t \leq 0$ (or

occasionally $\alpha_t \geq 1$, even though this is rare since α is typically small) during the online updating process, which can lead to infinite intervals (Zaffran et al. 2022).

For the *PPR* model, all three CP methods show slight over-coverage, while the BB and CB intervals exhibit under-coverage—except that the CB intervals achieve nominal coverage at $h = 1$. The under-coverage of the BB intervals can again be attributed to the overfitting issue discussed earlier, which results in much lower in-sample residual variance compared to the out-of-sample forecast error variance for the *PPR* model.

However, the under-coverage of the CB intervals contrasts with the better coverage observed for the CB method in the other models considered in this application. In this instance, it appears that the one-step-ahead forecast errors used in the CB method have not sufficiently accounted for the accumulation of errors in the multi-step forecasts as in the CP methods. Additionally, the truncation of predictors when generating sample paths autoregressively might also contribute to this issue. As noted earlier, these two factors appear to influence the coverage performance of the CB intervals differently, depending on the characteristics of the data and the forecasting model used. Therefore, further investigation into this issue would be beneficial.

3.5 Conclusion

In this chapter, we focused on constructing prediction intervals to quantify the uncertainty associated with point forecasts produced by the SMI model and other nonparametric additive models studied in Chapter 2. First, we employed block bootstrapping (BB) and three existing conformal prediction (CP) methods to generate interval forecasts within the generic online learning framework proposed by Wang & Hyndman (2024). We then introduced a novel methodology for constructing prediction intervals, referred to as Conformal Bootstrap (CB). The CB method is a natural amalgamation of block bootstrapping and classical split conformal prediction, combining the strengths of both approaches.

We investigated the characteristics and behaviour of the prediction intervals generated by each method through two empirical applications: forecasting heat exposure-related daily mortality and daily solar intensity.

The results showed that the coverage performance of the methods was not consistent across the two applications, suggesting that performance is largely influenced by factors such as the characteristics of the data, the forecasting model used, and the forecast horizon.

Both the BB and the proposed CB methods generally exhibited better coverage performance in the solar intensity forecasting application compared to the mortality forecasting application. Specifically, the

BB method, which showed under-coverage across all models in forecasting daily mortality, achieved mean coverage close to the nominal level for the *SMI-L*, *Backward*, and *GAIM* models when forecasting daily solar intensity. It is worth noting, however, that the forecast horizon in the mortality forecasting application was longer than that used in the solar intensity forecasting application.

Notably, the pattern of mean coverages for the *PPR* model across forecast horizons—particularly with BB and CB—differed from what is observed in the other three models across both empirical applications. A key structural difference of the *PPR* model is that all predictor variables contribute to every estimated index (i.e. linear combination), whereas in the other models, each predictor is assigned to only one index. This dense structure in *PPR* likely introduces more complex interactions among predictors, increasing the risk of overfitting compared to the other relatively simpler models. These complexities influence the performance of the prediction interval generation methods when applied to the *PPR* model, depending on the data characteristics in a given application.

Interestingly, for *GAIM* in the solar intensity forecasting application, the CP and CB methods exhibited mean coverages lower than that of the BB method—unlike what is observed for the other models. The variance of the in-sample residuals was higher than that of the out-of-sample forecast errors for *GAIM* in many of the cross-validation steps, due to poorly fitted models. This resulted in wider BB intervals—as shown in Figure 3.16—which, in turn, led to higher coverage. Additionally, the use of one-step-ahead forecast errors (as opposed to the multi-step-ahead errors used in CP), and the possible influence of predictor truncation during the generation of sample paths, may have contributed to the comparatively lower coverage performance of the CB method. However, further analysis would be required to understand this behaviour better.

Although time series data typically violate the exchangeability assumption underpinning the CP methods, the three methods considered—split conformal prediction (SCP), weighted split conformal prediction (WSCP) with exponential weights, and adaptive conformal prediction (ACP)—generally performed well across models in both applications. While theoretical coverage guarantees are not expected under such conditions, the empirical results suggest that these methods can still be effective in practice.

The prediction intervals generated using the proposed CB method demonstrated calibration and sharpness that were generally comparable to those of the CP methods, sometimes performing slightly worse, but outperforming the BB method in most cases. This aligns with our expectations, given that CB combines elements of both BB and SCP. However, further research is needed to identify the specific scenarios in which CB outperforms existing methods in terms of actual coverage performance.

Chapter 4

smimodel: An R Package to Estimate Sparse Multiple Index Models for Nonparametric Forecasting

Nonparametric additive models have been applied in various fields, yet their implementation poses several unresolved challenges. A key issue is the selection of predictor variables from the available predictors to include in the model in high-dimensional settings. A critical challenge also lies in deciding how to group predictor variables to model interactions. We have proposed the *Sparse Multiple Index (SMI) Modelling* algorithm in Chapter 2, which performs automatic predictor selection and predictor grouping, providing an objective and principled estimation method for nonparametric additive index models to address the aforementioned challenges. Here we describe the R package `smimodel`, an open-source, automated implementation of the SMI Modelling algorithm. The package also provides implementations of the *block bootstrap* and *conformal bootstrap* methods used to generate prediction intervals, as described in Chapter 3.

4.1 Introduction

Nonparametric additive models have recently gained popularity as a versatile statistical tool in various forecasting and prediction applications (Hyndman & Fan 2010; Du, Cheng & Liang 2012; Goude, Nedellec & Kong 2014; Wood 2017; Ravindra et al. 2019; Ho, Chen & Hwang 2020; Boente & Martínez 2023; Ibrahim et al. 2023). However, when applied to high-dimensional data, model estimation is challenging, mainly due to the need to estimate many nonparametric functions. This naturally leads to the assumption of sparsity in real-world scenarios, where only a limited set of

predictors and their interactions are considered relevant. A review of related previous studies reveals a considerable degree of subjectivity in the process of selecting predictor variables and grouping them for modelling interactions. These choices are typically based on empirical exploration or expert knowledge.

Chapter 2 of this thesis proposed the *Sparse Multiple Index (SMI)* model, which uses algorithmic predictor selection and grouping to address the above-stated challenges, with a particular focus on time series forecasting. Chapter 3 of this thesis further extended the proposed forecasting workflow by discussing techniques for estimating the uncertainty of point forecasts generated via nonparametric additive models and proposed *Conformal Bootstrap (CB)*, a novel method for constructing prediction intervals. Disseminating these methods to a wider audience for the benefit of the forecasting community is the next step.

The Comprehensive R Archive Network (CRAN) hosts several packages for estimating nonparametric/semiparametric additive models using various techniques, with `mgcv` (Wood 2011, 2023) and `gam` (Hastie 2024) being two of the most widely used. In this thesis, we are interested in time series forecasting applications with numerous lagged variables as predictors that exhibit complicated interactions. A class of models which is well suited to this type of forecasting problem is *distributed lag models (DLMs)*. There are a number of packages on CRAN that are designed to estimate DLMs in various circumstances, including `dLagM` (Demirhan 2020), which fits DLMs for time series regression, `dlnm` (Gasparrini 2011), which provides functions to fit linear and non-linear DLMs, and `dlmtree` (Mork, Im & Wilson 2024), which fits DLMs based on a Bayesian additive regression tree framework. While the `mgcv` package is not specifically designed for DLMs, Wood (2017) provided an example of estimating a DLM with `mgcv` to forecast daily death rates based on multiple lags of air pollutant measurements and interactions between the corresponding lags of two pollutants through a sum of multiple nonparametric components. However, none of the above packages provides functionality to estimate linear indices by grouping predictors or to perform predictor selection.

The package `cgaim` (Masselot et al. 2023) provides functions to fit constrained (or unconstrained) groupwise additive index models using predetermined groups of predictors. Although the focus of the package is not on forecasting or prediction, but rather on inference and interpretation of the estimated indices, the model fitting functions provided can be used for such purposes. However, `cgaim` does not have the capability to perform automatic predictor grouping or variable selection.

To our knowledge, no existing open-source software provides an objective and principled predictor selection and predictor grouping method for estimating nonparametric additive index models. Therefore, we have developed the R package `smimodel`, which provides an implementation of the SMI

Modelling algorithm that automates the parsimonious estimation of nonparametric additive index models, thus filling a noticeable gap in the field of nonparametric forecasting.

The rest of this chapter has the following structure. Section 4.2 introduces `smimodel` with details of the package infrastructure, installation instructions, and details of additional software requirements. In Section 4.3, the functionality of the package is demonstrated using a real-world problem of forecasting daily electricity demand. Section 4.4 provides comparisons with some selected existing packages related to nonparametric additive models and DLMs. The conclusions of the chapter are presented in Section 4.5 with a discussion of the limitations of the package and the future improvement plans.

4.2 About smimodel

The main purpose of `smimodel` is to provide functions for estimating SMI models for nonparametric forecasting and prediction. As is apparent, the name `smimodel` stands for *sparse multiple index model*. While the functions of the package are designed to work with tidy temporal data in the `tsibble` format (Wang, Cook & Hyndman 2020), the underlying SMI model is inherently flexible and not constrained by any temporal features. This makes the model applicable beyond time series forecasting and allows it to be used with a wide variety of data, including non-temporal, cross-sectional data sets.

4.2.1 Installation

The current version of `smimodel` can be installed from GitHub as follows.

```
## Install `smimodel`
# install.packages("pak")
pak::pak("nuwani-palihawadana/smimodel")
```

The `smimodel` package is actively developed and maintained, and the latest updates are publicly available on GitHub at `github.com/nuwani-palihawadana/smimodel`.

4.2.2 Other software requirements

Gurobi solver

The estimation of the SMI model is based on an iterative procedure that is developed using mixed integer programming (MIP) to solve an ℓ_0- and ℓ_2-regularised non-linear least squares optimisation problem with linear constraints. We use the commercial MIP solver ***Gurobi*** (*Gurobi Optimizer Reference Manual* 2023) to solve the mixed integer programs, as it is the fastest and the most powerful MIP solver currently available. Hence, users of `smimodel` are required to have an active installation of

Gurobi on their local machines to use the SMI model estimation functions. If not, an error will be raised.

As Gurobi is a commercial software, users are required to obtain a licence for Gurobi before they can install or use it. Gurobi provides a free licence for academics affiliated with recognised educational institutions. Therefore, an academic can register for a free account with their verified institutional email and then request a free academic licence. A user who is not affiliated with a recognised educational institution may need to purchase a licence for long-term use. However, a free 30-day trial can be requested for an evaluation licence.

Fortunately, Gurobi provides extensive documentation (https://docs.gurobi.com/current/) on both installation and usage, as well as active support through the Gurobi Help Centre (https://support.gurobi.com/hc/en-us). For more helpful resources, users of `smimodel` can refer to its README file.

R package gurobi

Once the Gurobi solver is installed on the user's local machine, the next step is to install the Gurobi R interface through the `gurobi` R package, which enables the use of Gurobi within R. This package is not available on CRAN and is distributed exclusively with the Gurobi software suite.

4.2.3 Main functionality and design

Key functionalities

The key functions of `smimodel` and their roles within the overall process are briefly illustrated in Figure 4.1.

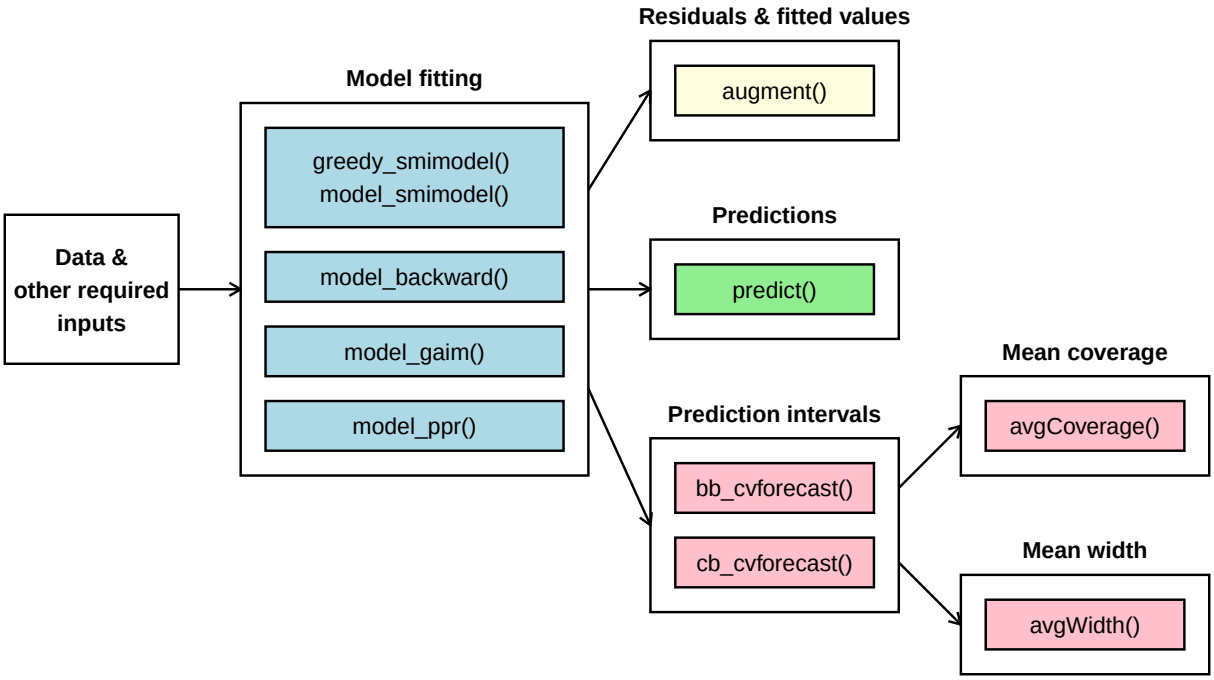


Figure 4.1: *Workflow diagram illustrating the key functionalities of* `smimodel`.

SMI model

As previously mentioned, the package `smimodel` is primarily developed to implement the SMI Modelling algorithm for estimating SMI models in the context of nonparametric forecasting and prediction. The main function for SMI model estimation is `greedy_smimodel()`. This function estimates a SMI model with a preferred initialisation option (refer Section 2.2.2), while tuning the two penalty parameters, λ_0 and λ_2, corresponding to the ℓ_0 penalty and ℓ_2 penalty, respectively in the SMI model optimisation problem (refer Section 2.2.1). The tuning is performed through a greedy search over a partially data-derived grid of possible values. (Refer Section 2.2.2 for details about the greedy search.)

Alternatively, the function `model_smimodel()` estimates a SMI model using user-specified values for the two penalty parameters, along with a chosen initialisation option. However, since the best penalty parameter values are typically identified through hyperparameter tuning, the use of the `greedy_smimodel()` function is recommended.

Other nonparametric models

The `smimodel` R package also provides functionality to fit several benchmark comparison methods, including the nonparametric additive model with backward elimination (Backward, Fan & Hyndman 2012), the group-wise additive index model (GAIM), and the projection pursuit regression model (PPR, Friedman & Stuetzle 1981). These methods can be implemented using the functions `model_backward()`, `model_gaim()`, and `model_ppr()`, respectively. These functions are designed to follow a syntax and output format consistent with `greedy_smimodel()` and `model_smimodel()` where possible, for ease of use and interchangeability.

A special feature of the model fitting functions in `smimodel` is their ability to fit multiple models based on a grouping variable inherent in the data set. When fitting multiple models, the grouping variable should be specified as the `key` of the data set in `tsibble` format, which is then passed to the model fitting function. If the data set does not have a `key`, the model fitting functions will automatically create a dummy key with a single level, fitting a single model to the entire data set. It is important to note that the functions `model_gaim()` and `model_ppr()` are wrappers around `cgaim::cgaim()` (only supporting unconstrained GAIMs) and `stats::ppr()`, respectively, to facilitate the fitting of multiple models based on a grouping variable.

The package `smimodel` uses the standard S3 generic system of R, and adheres to the tidyverse framework. Therefore, each of the model fitting functions assigns a separate class to the model object returned, where the classes `smimodel`, `backward`, `gaimFit` and `pprFit` correspond to the models

fitted using `greedy_smimodel()`/`model_smimodel()`, `model_backward()`, `model_gaim()` and `model_ppr()`, respectively. All these model object classes return a special type of `tibble` (Müller & Wickham 2023), and hence inherit the classes `c("tbl_df", "tbl", "data.frame")`. The package provides a separate `augment` method for each class to obtain fitted values and residuals of the fitted model, and a `predict` method to obtain forecasts/predictions on a given test set.

Prediction intervals

Another key feature of `smimodel` is its support for prediction interval methods, which quantify the uncertainty associated with the point forecasts produced by the nonparametric models fitted using the aforementioned model fitting functions. The `bb_cvforecast()` and `cb_cvforecast()` functions apply block bootstrap (BB) and conformal bootstrap (CB) techniques, respectively, to construct prediction intervals via time series cross-validation, as explained in Chapter 3. These functions support the `smimodel`, `backward`, `gaimFit`, and `pprFit` classes. Additionally, the functions `avgCoverage()` and `avgWidth()` compute the mean coverage and mean width of the estimated prediction intervals, respectively, serving as wrappers for `conformalForecast::coverage()` and `conformalForecast::width()` (Wang & Hyndman 2025).

Input requirements

Data format requirements

All the model fitting functions in the `smimodel` package accept only `tsibble` format data sets. For time series data, the `index` of the `tsibble` should be a date-time, date, or time-span variable indicating the time index of the time series variable we want to forecast. It is important to ensure that there are no additional date-time, date, or time-span variables in the data set other than the `index`, as such variables could interfere with the various data conversions and mathematical operations performed within the model fitting functions. As mentioned above, the SMI model formulation does not depend exclusively on any temporal features and can therefore be used with cross-sectional data. In such a case, a numeric variable can be used as the `index` when constructing the `tsibble`, instead of a date-time, date, or time-span variable, acting as a dummy index.

Variable specification

The model fitting functions in `smimodel` identify the response and predictor variables by different arguments. The argument `yvar` takes the name of the response variable as a character string. The arguments `index.vars`, `s.vars`, and `linear.vars` accept the names of the predictor variables based on their roles in the model: `index.vars` for variables involved in linear indices, `s.vars` for variables entering the model non-linearly, and `linear.vars` for variables entering the model linearly.

The specific assignment of predictor variables to these arguments depends on the type of model being estimated.

Validation settings

In `greedy_smimodel()`, the best penalty parameter values are selected based on the combination of values that gives the lowest MSE on a validation set that is non-overlapping with the training set. Moreover, in `model_backward()`, the optimal model is obtained based on a validation set. Hence, in these two instances, it is required to pass a separate validation data set in addition to the training data set through the argument `val.data` in the corresponding model-fitting functions.

Handling of lagged variables

If lags of a particular variable are used as predictors in time series forecasting problems, the input `tsibble` should contain those lagged variables. The `lag_matrix()` function provided in the package can be used to construct those lagged variables as required.

When lags of the response variable are used as predictors, recursive (autoregressive) forecasting is performed when generating forecasts on a test set, or on a validation set for computing forecast error during penalty parameter tuning or model selection. In such cases, the lagged response variables must be positioned sequentially in increasing lag order (i.e. lag 1, lag 2, ..., lag m, where lag m is the maximum lag) in the test or validation set, with no breaks in the sequence—even if some intermediate lags are not used as predictors in the model.

Additionally, the `recursive` argument should be set to `TRUE` to indicate that recursive forecasting is required, and the range of column numbers corresponding to the lagged response variables must be specified using the `recursive_colRange` argument. These arguments should be provided in the `greedy_smimodel()` and `model_backward()` functions (for validation set forecasting), and in the `predict()` function when forecasting on a test set.

Other inputs

As explained in Chapter 2, when generating out-of-sample forecasts/predictions using models that include splines to estimate non-linear link functions, the values of predictor variables treated non-linearly are truncated to fall within the training data range to avoid extrapolation with splines. However, in real-life forecasting/prediction scenarios, there may be predictors for which such truncation is not appropriate (see, for example, Section 2.4.1). If such predictors are present, their names should be specified using the `exclude.trunc` argument in both the `greedy_smimodel()` and `model_backward()` functions, which involve forecasting/predicting for a validation set used in parameter tuning/model selection. The same argument should also be specified when forecast-

ing/predicting for a test set using the `predict()` function. By default, this argument is set to `NULL`.

There are several other arguments related to the model structure and the estimation process, depending on the model fitted, as well as Gurobi-related arguments in the SMI model estimation functions. Default values are provided for these arguments, while the user can modify them if necessary, depending on the forecasting/prediction problem of interest. See the package documentation for more details on the arguments.

Package dependencies

The `smimodel` package utilises several existing R packages in its implementation, including the following. The *R Optimisation Infrastructure* (`ROI`) package (Theußl, Schwendinger & Hornik 2020; Hornik et al. 2023) is used to set up the optimisation problems and as the main communicator between the Gurobi R interface and the optimisation problems. Nonparametric additive models are estimated using the `mgcv` package (Wood 2011, 2023). Note that here, we use penalised cubic regression splines to estimate the non-linear functions. The `cgaim` package (Masselot et al. 2023) is used in instances where GAIMs are estimated. The `stats` package is used for estimating PPR models.

4.3 Usage of smimodel

In this section, we illustrate the functionality of the `smimodel` package using an example of daily electricity demand forecasting. As the focus of this thesis is on nonparametric forecasting, we use a time series forecasting problem to demonstrate the functionality of the package. However, as mentioned above, the package could also be used for non-temporal prediction problems.

4.3.1 Data

We use the data set `vic_elec` from the `tsibbledata` R package (O'Hara-Wild et al. 2022), which was originally provided by the Australian Energy Market Operator (AEMO). It contains half-hourly electricity demand data from 2012 to 2014 in Victoria, Australia. The data set also contains half-hourly temperature readings from the Melbourne Regional Office weather station, and a binary variable indicating whether a given day is a public holiday.

For this demonstration, we convert the half-hourly demand data into daily data by taking the sum of the electricity demand measurements over each day. We also consider the daily maximum temperature (hereafter referred to as "temperature"), derived from the available half-hourly data, as a predictor variable.

```
## Load required packages
library(smimodel)
library(dplyr)
library(ggplot2)
library(lubridate)
library(tidyr)
library(tsibble)
library(tsibbledata)
```

```
## Data preparation: Convert half-hourly data into daily data
vic_elec_daily <- vic_elec |>
  index_by(Date = date(Time)) |>
  summarise(
    Demand = sum(Demand) / 1e3,
    Temperature = max(Temperature),
    Holiday = any(Holiday)
  )
vic_elec_daily
```

```
# A tsibble: 1,096 x 4 [1D]
   Date       Demand Temperature Holiday
   <date>      <dbl>       <dbl> <lgl>
 1 2012-01-01   222.        32.7 TRUE
 2 2012-01-02   258.        39.6 TRUE
 3 2012-01-03   267.        31.8 FALSE
 4 2012-01-04   223.        25.1 FALSE
 5 2012-01-05   211.        21.2 FALSE
 6 2012-01-06   210.        23.6 FALSE
 7 2012-01-07   203.        29   FALSE
 8 2012-01-08   193.        27.8 FALSE
 9 2012-01-09   214.        24   FALSE
10 2012-01-10   215.        19.6 FALSE
# i 1,086 more rows
```

Here, the `index` of the `tsibble` is the variable `Date`.

Predictors considered

In this example, we use the following variables as predictors in the forecasting models.

1) Electricity demand lags:

The lags of the daily electricity demand are used as predictors to incorporate the serial correlation present in the data into the modelling process (Fan & Hyndman 2012). Here, we consider seven lags of electricity demand (i.e. one week of historical demand measurements).

2) Current temperature and temperature lags:

Due to thermal inertia in buildings, the electricity demand on a given day depends not only on the temperature on that day but also on historical temperatures. Hence, in addition to the current temperature, we consider a seven-day history, i.e. seven lags of the temperature, as predictors in the model.

3) Calendar effects:

Finally, a couple of calendar variables, *day of the week (DOW)* and *holiday effect (Holiday)* (with four levels: "holiday", "day before holiday", "day after holiday", and "not a holiday") are included in the model to capture daily and weekly seasonalities.

Thus, we prepare the data set as follows, with all the required predictor variables. Note that we use `lag_matrix()` from `smimodel` to construct the lagged demand and temperature variables.

```
## Data preparation: Add required predictor variables
vic_elec_daily <- vic_elec_daily |>
  # Rename Demand and Temperature variables
  rename(Demand_lag_000 = Demand,
         Temp_lag_000 = Temperature) |>
  mutate(
    # Add lagged Demand
    Demand_lag = lag_matrix(Demand_lag_000, 7),
    # Add lagged Temperature
    Temp_lag = lag_matrix(Temp_lag_000, 7)
  ) |>
  unpack(c(Demand_lag, Temp_lag), names_sep = "_") |>
  # Add calendar variables
  mutate(
```

```
    DOW = factor(wday(Date, label = TRUE), ordered = FALSE),
    Holiday = case_when(
      Holiday ~ "Holiday",
      lead(Holiday, 1) ~ "Day before holiday",
      lag(Holiday, 1) ~ "Day after holiday",
      TRUE ~ "Not a holiday"
    ),
    Holiday = factor(Holiday)
  ) |>
  # Reorder columns
  select(Date, DOW, Holiday, starts_with("Demand"), starts_with("Temp")) |>
  # Convert to a `tsibble`
  as_tsibble(index = Date)
vic_elec_daily
```

```
# A tsibble: 1,096 x 19 [1D]
   Date       DOW   Holiday         Demand_lag_000 Demand_lag_001 Demand_lag_002
   <date>     <fct> <fct>                    <dbl>          <dbl>          <dbl>
 1 2012-01-01 Sun   Holiday                   222.            NA             NA
 2 2012-01-02 Mon   Holiday                   258.           222.            NA
 3 2012-01-03 Tue   Day after holi~           267.           258.           222.
 4 2012-01-04 Wed   Not a holiday             223.           267.           258.
 5 2012-01-05 Thu   Not a holiday             211.           223.           267.
 6 2012-01-06 Fri   Not a holiday             210.           211.           223.
 7 2012-01-07 Sat   Not a holiday             203.           210.           211.
 8 2012-01-08 Sun   Not a holiday             193.           203.           210.
 9 2012-01-09 Mon   Not a holiday             214.           193.           203.
10 2012-01-10 Tue   Not a holiday             215.           214.           193.
# i 1,086 more rows
# i 13 more variables: Demand_lag_003 <dbl>, Demand_lag_004 <dbl>,
#   Demand_lag_005 <dbl>, Demand_lag_006 <dbl>, Demand_lag_007 <dbl>,
#   Temp_lag_000 <dbl>, Temp_lag_001 <dbl>, Temp_lag_002 <dbl>,
#   Temp_lag_003 <dbl>, Temp_lag_004 <dbl>, Temp_lag_005 <dbl>,
#   Temp_lag_006 <dbl>, Temp_lag_007 <dbl>
```

We use data from the first two years, 2012 and 2013, as the training set to estimate the forecasting models. Data from January 2014 to February 2014 are set aside as a validation set, which is used to tune the penalty parameters (λ_0 and λ_2) when estimating the SMI models and to perform model selection for the nonparametric additive model using backward elimination.

In both cases, once penalty parameter tuning or model selection is completed, the final model is re-fitted by default to the combined training and validation sets. This behaviour can be overridden by setting `refit = FALSE` in the respective model-fitting functions to fit the model only to the training set.

The data from March 2014 to December 2014 are used as the test set to evaluate the forecasting performance of the fitted models, amounting to a 306-step-ahead (or 10-month-ahead) forecast. (We use a long test set here because we later construct prediction intervals, which require a long test set to carry out the time series cross-validation procedure.)

```
## Data split
# Training
train <- vic_elec_daily |>
  filter(Date <= "2013-12-31")
# Validation
val <- vic_elec_daily |>
  filter(Date >= "2014-01-01" & Date <= "2014-02-28")
# Test
test <- vic_elec_daily |>
  filter(Date >= "2014-03-01")
```

4.3.2 Modelling framework

When estimating a SMI model, the lagged electricity demand variables and the lagged temperature variables are taken as predictors that can enter the indices, while the variables *DOW* and *Holiday* are included as linear predictors in the model.

Thus, the corresponding SMI model can be expressed as

$$\mathbf{Demand} = \beta_0 + \sum_{j=1}^{q} g_j(\boldsymbol{X}\boldsymbol{\alpha}_j) + \sum_{k=1}^{6} \theta_k \mathbf{DOW}_{\boldsymbol{k}} + \sum_{l=1}^{3} \delta_l \mathbf{Holiday}_{\boldsymbol{l}} + \boldsymbol{\varepsilon},$$

where

- **Demand** is the vector of daily electricity demand observations;

- β_0 is the model intercept;
- q is the unknown number of indices to be estimated by the SMI Modelling algorithm;
- $\boldsymbol{X}$ is a matrix containing the predictor variables that can enter the indices (i.e. electricity demand lags and temperature lags);
- $\boldsymbol{\alpha}_j$, $j = 1, \ldots, q$, are the index coefficient vectors;
- $g_1, \ldots, g_q$ are unknown nonparametric functions;
- $\mathbf{DOW}_k$, $k = 1, \ldots, 6$, are the six dummy variables representing the seven days of the week;
- θ_k, $k = 1, \ldots, 6$, are the corresponding coefficients;
- $\mathbf{Holiday}_l$, $l = 1, \ldots, 3$, are the three dummy variables representing the four levels of the *Holiday* variable;
- δ_l, $l = 1, \ldots, 3$, are the corresponding coefficients; and
- $\boldsymbol{\varepsilon}$ is the error term.

When estimating a nonparametric additive model with backward elimination, all of the above index variables are considered as single non-linear predictors, as no indices are estimated in the model. The treatment of the predictors in a GAIM will be similar to that of the SMI model (except for the fact that the predictors are grouped in advance). The two linear variables (i.e. the calendar variables) will not be included in estimating a PPR model, as there is no possibility of including separate linear variables in `stats::ppr()`, where all the provided predictors are considered as index variables, which is an inappropriate treatment for categorical variables.

4.3.3 Implementation

SMI model estimation

Here, we demonstrate how to estimate SMI models using the `smimodel` package, with simultaneous penalty parameter tuning.

First, we fit a SMI model with the `"ppr"` initialisation option (*SMI-P*), which is the default option in the `greedy_smimodel()` function.

```
# Fit SMI model with "ppr" initialisation
smimodel_ppr <- greedy_smimodel(data = train,
                                val.data = val,
                                yvar = "Demand_lag_000",
                                index.vars = colnames(vic_elec_daily)[5:19],
                                linear.vars = c("DOW", "Holiday"),
                                nlambda = 20,
```

```
                                    recursive = TRUE,
                                    recursive_colRange = 5:11)
```

In this case, the argument `data` is used to specify the training data set, while the argument `val.data` is used for the validation data set, which is employed to tune the penalty parameters. The response variable is specified using the `yvar` argument. The predictors that may enter the indices—the seven lagged demand variables, current temperature, and the seven lagged temperature variables (columns 5 to 19 in the data set)—are specified using the `index.vars` argument. The calendar variables, *DOW* and *Holiday*, are specified as linear predictors using the `linear.vars` argument.

Since we use the default initialisation option, there is no need to specify it explicitly in the function call. In the `"ppr"` initialisation option, the SMI model estimation algorithm is initialised with a PPR model. When fitting a PPR model, it is necessary to specify how many indices are to be estimated for the model. The `num_ind` argument to `greedy_smimodel()` accepts this input, with a default value of five. In this example, we use the default number of indices (i.e. five) for the initial PPR model fitted, so the argument does not need to be explicitly specified.

Given that the lags of the response variable itself are used as predictors in the model, autoregressive forecasting must be performed in the validation set. Therefore, we indicate the requirement for autoregressive forecasting by setting `recursive = TRUE`, and provide the range of column numbers in the data set corresponding to the seven lagged demand variables in `recursive_colRange`.

The `greedy_smimodel()` function determines the maximum value for the penalty parameter λ_0, say λ_0`.max`, using the scale of the first term of the SMI model objective function (refer to Section 2.2.1). It determines a pseudo minimum value, as a fraction of λ_0`.max`. The default fraction used is 0.0001, which can be adjusted by changing the value of the argument `lambda.min.ratio`. Once the maximum and pseudo minimum values are determined, a sequence of values for λ_0 is constructed on a logarithmic scale, with zero added as the minimum of the λ_0 sequence. The argument `nlambda` determines the length of this sequence (excluding zero), and, in this case, we use 20 values.

Next, we look at the estimated model.

```
# Returned smimodel object
smimodel_ppr
```

```
Fitted SMI Model(s):
# A tibble: 1 x 2
  key   fit
```

```
  <chr> <list>
1 1     <named list [6]>
```

`greedy_smimodel()` returns an object of class `smimodel`, which is a `tibble` with two columns: `key` indicates the level of the grouping variable (if multiple models are fitted), and `fit` contains the information of the fitted model corresponding to the `key`. As we fit a single model for the entire data set (no grouping variable), the returned `smimodel` object is a `tibble` with a single row.

Each element in the column `fit` (here we have only one element) contains a `list` with six elements: `initial`, a `list` containing information on the model initialisation; `best`, a `list` containing information on the optimised/final SMI model fit; `best_lambdas`, a `numeric` vector of selected penalty parameter values; `lambda0_seq`, the sequence of values for λ_0 used to construct the initial grid; `lambda2_seq`, the sequence of values for λ_2 used to construct the initial grid; and `searched`, a `tibble` containing the penalty parameter combinations searched during the greedy search and the corresponding validation set MSEs.

Here, we can see that the selected penalty parameter combination is $(\lambda_0, \lambda_2) = (0, 10^4)$, indicating that no ℓ_0 penalty was applied in model estimation.

```
# Best penalty parameter combination
smimodel_ppr$fit[[1]]$best_lambdas
```

```
[1]     0 10000
```

The details of the estimated SMI model can be obtained as follows, where the sparse matrix in the first part of the printed output shows the estimated index structure of the SMI model.

```
# Estimated model
smimodel_ppr$fit[[1]]$best
```

```
SMI Model Fit:
Index coefficients:
15 x 3 sparse Matrix of class "dgCMatrix"
                    index1      index2     index3
Demand_lag_001  0.71662585  .          .
Demand_lag_002  .           .          0.15415451
Demand_lag_003  0.11385967  .          .
Demand_lag_004  .           .          0.08883482
Demand_lag_005  0.04379256  .          .
```

```
Demand_lag_006  .           .           0.06654211
Demand_lag_007  0.04513242  .           .
Temp_lag_000    .          -0.82643919  .
Temp_lag_001    .           .           0.29927157
Temp_lag_002   -0.08058950  .           .
Temp_lag_003    .          -0.05198032  .
Temp_lag_004    .          -0.04805902  .
Temp_lag_005    .          -0.07352147  .
Temp_lag_006    .           .           0.11919403
Temp_lag_007    .           .           0.27200296

Response variable:
[1] "Demand_lag_000"

Index variables:
 [1] "Demand_lag_001" "Demand_lag_002" "Demand_lag_003" "Demand_lag_004"
 [5] "Demand_lag_005" "Demand_lag_006" "Demand_lag_007" "Temp_lag_000"
 [9] "Temp_lag_001"   "Temp_lag_002"   "Temp_lag_003"   "Temp_lag_004"
[13] "Temp_lag_005"   "Temp_lag_006"   "Temp_lag_007"

Spline variables (non-index):
NULL

Linear variables:
[1] "DOW"     "Holiday"

GAM Fit:
Family: gaussian
Link function: identity

Formula:
Demand_lag_000 ~ s(index1, bs = "cr") + s(index2, bs = "cr") +
    s(index3, bs = "cr") + DOW + Holiday
```

```
Estimated degrees of freedom:
4.95 6.55 4.83  total = 26.33


REML score: 2575.443
```

According to the output above, the SMI Modelling algorithm has estimated three indices and has not dropped any predictor from the model.

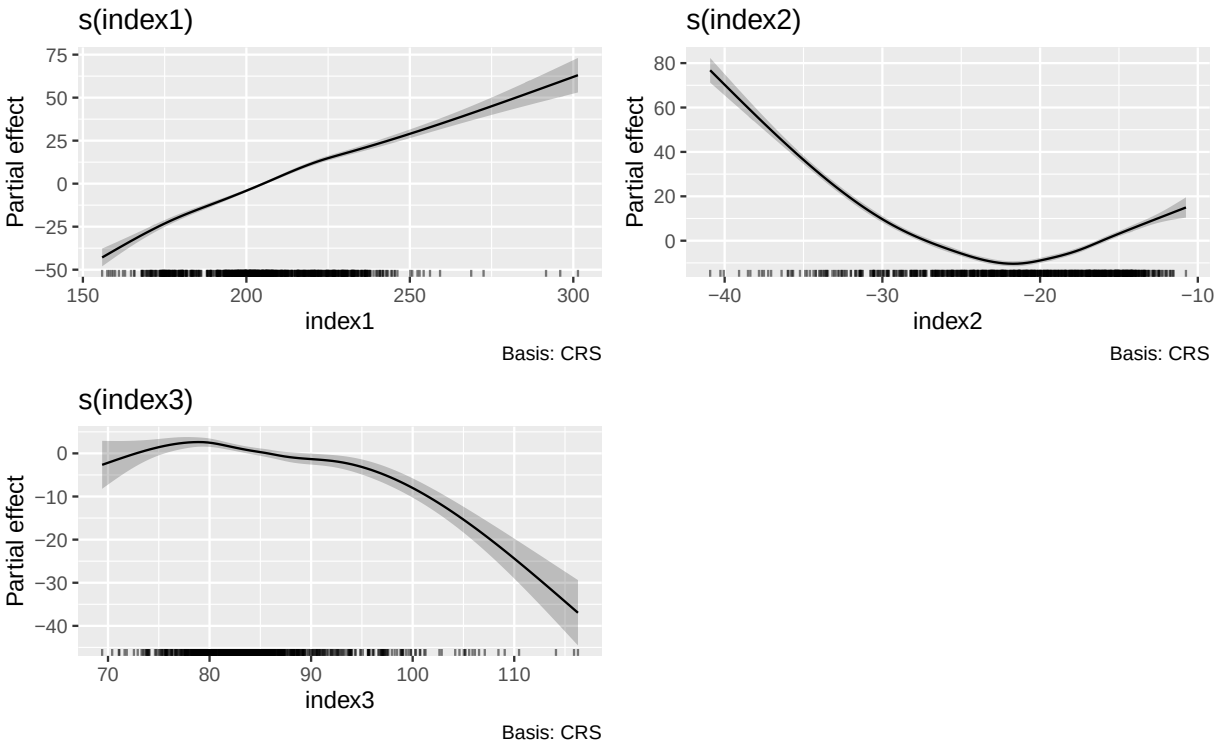
To obtain the fitted values and residuals of the estimated SMI model in an organised format as a `tibble`, we can use the `augment()` function, which will call the `augment.smimodel` method to generate the output.

```
# Using augment function
augment(smimodel_ppr)
```

```
# A tibble: 783 x 3
   Index      .resid .fitted
   <date>      <dbl>   <dbl>
 1 2012-01-08 -2.86     196.
 2 2012-01-09 -6.93     221.
 3 2012-01-10 -2.77     218.
 4 2012-01-11 -0.769    218.
 5 2012-01-12  0.799    218.
 6 2012-01-13  2.34     214.
 7 2012-01-14  8.74     186.
 8 2012-01-15  7.15     182.
 9 2012-01-16 10.9      242.
10 2012-01-17 24.1      265.
# i 773 more rows
```

The `autoplot()` function can be used to plot the fitted smooths/splines corresponding to each of the estimated indices of the fitted SMI model, which calls the `autoplot.smimodel` method.

```
# Plot the estimated smooths
autoplot(smimodel_ppr)
```



Next, we estimate two more SMI models with two other initialisation options: `"additive"` (*SMI-A*) and `"linear"` (*SMI-L*), for comparison purposes. Here, we have to explicitly pass the initialisation option we want to use in the argument `initialise`.

```
# Fit SMI model with "additive" initialisation
smimodel_additive <- greedy_smimodel(data = train,
                                     val.data = val,
                                     yvar = "Demand_lag_000",
                                     index.vars = colnames(vic_elec_daily)[5:19],
                                     linear.vars = c("DOW", "Holiday"),
                                     initialise = "additive",
                                     nlambda = 20,
                                     recursive = TRUE,
                                     recursive_colRange = 5:11)
```

```
# Best penalty parameter combination
smimodel_additive$fit[[1]]$best_lambdas
```

```
[1]   169.5129 10000.0000
```

```
# Estimated index structure
smimodel_additive$fit[[1]]$best$alpha
```

```
15 x 3 sparse Matrix of class "dgCMatrix"
                   index1     index2     index3
Demand_lag_001 0.7022630 .          .
Demand_lag_002 0.1912722 .          .
Demand_lag_003 .         .          0.01673196
Demand_lag_004 .         0.02023022 .
Demand_lag_005 .         .          0.02410018
Demand_lag_006 0.1064648 .          .
Demand_lag_007 .         0.01194447 .
Temp_lag_000   .         0.80060888 .
Temp_lag_001   .         .          0.24389897
Temp_lag_002   .         .          0.34066982
Temp_lag_003   .         0.07773119 .
Temp_lag_004   .         .          0.09775953
Temp_lag_005   .         0.08948524 .
Temp_lag_006   .         .          0.10591822
Temp_lag_007   .         .          0.17092132
```

For *SMI-A*, the selected penalty parameter combination is $(\lambda_0, \lambda_2) = (169.5129, 10^4)$. The fitted model includes all the available predictors, and three indices are estimated.

```
# Fit SMI model with "linear" initialisation
smimodel_linear <- greedy_smimodel(data = train,
                                   val.data = val,
                                   yvar = "Demand_lag_000",
                                   index.vars = colnames(vic_elec_daily)[5:19],
                                   linear.vars = c("DOW", "Holiday"),
                                   initialise = "linear",
                                   nlambda = 20,
                                   recursive = TRUE,
                                   recursive_colRange = 5:11)
```

```
# Best penalty parameter combination
smimodel_linear$fit[[1]]$best_lambdas
```

```
[1] 104.3942   0.0000
```

```
# Estimated index structure
smimodel_linear$fit[[1]]$best$alpha
```

```
15 x 2 sparse Matrix of class "dgCMatrix"
                     index1      index2
Demand_lag_001  0.21796644  .
Demand_lag_002 -0.02691696  .
Demand_lag_003  .           0.02821929
Demand_lag_004  0.01227509  .
Demand_lag_005  .          -0.02429662
Demand_lag_006  .           0.20999853
Demand_lag_007  .          -0.09609086
Temp_lag_000    0.40407551  .
Temp_lag_001   -0.23444288  .
Temp_lag_002    .          -0.44926630
Temp_lag_003    .           0.05941088
Temp_lag_004   -0.04408303  .
Temp_lag_005    .          -0.13271751
Temp_lag_006   -0.06024010  .
Temp_lag_007    .           .
```

For *SMI-L*, the selected penalty parameter combination is $(\lambda_0, \lambda_2) = (104.3942, 0)$. The fitted model estimates two indices, and the seventh temperature lag has been dropped.

Benchmark models

Next, we look at how to estimate a few other nonparametric models using `smimodel` as benchmarks for comparing the forecasting performance of the estimated SMI models.

First, we estimate a nonparametric additive model with backward elimination (Fan & Hyndman 2012) (*Backward*) using the function `model_backward()`.

```
# Fit nonparametric additive model with backward elimination
backward <- model_backward(data = train,
                           val.data = val,
                           yvar = "Demand_lag_000",
                           s.vars = colnames(vic_elec_daily)[5:19],
                           linear.vars = c("DOW", "Holiday"),
```

```
                              recursive = TRUE,
                              recursive_colRange = 5:11)
```

Note that we have included the lagged demand and lagged temperature variables in the `s.vars` argument, as opposed to `index.vars` when estimating the SMI models, because, in this case, we are estimating an additive model, not an index model, and these predictors enter the model as individual non-linear variables (not grouped into indices). All other arguments remain the same as for the SMI model estimation.

Now, we examine the estimated model.

```
# Returned backward object
backward
```

```
Fitted Model(s):
# A tibble: 1 x 2
  key   fit
  <chr> <list>
1 1     <gam>
```

The model object returned is of the class `backward`, which is again a `tibble` with two columns: `key` and `fit`. In this case, each element in the `fit` column is a `gam` object (Wood 2023, 2011), providing information about the fitted additive models corresponding to each `key`. As we are fitting a single model to the entire data set, the `backward` object returned in this case is a single-row `tibble`.

We can access the fitted `gam` object as follows.

```
# Fitted additive model
backward$fit[[1]]
```

```
Family: gaussian
Link function: identity

Formula:
Demand_lag_000 ~ +s(Demand_lag_003, bs = "cr") + s(Demand_lag_004,
    bs = "cr") + s(Demand_lag_006, bs = "cr") + s(Demand_lag_007,
    bs = "cr") + s(Temp_lag_000, bs = "cr") + s(Temp_lag_001,
    bs = "cr") + s(Temp_lag_002, bs = "cr") + s(Temp_lag_003,
```

```
    bs = "cr") + s(Temp_lag_007, bs = "cr") + DOW + Holiday

Estimated degrees of freedom:
3.24 1.00 1.39 1.00 6.94 5.89 1.00
4.01 3.84  total = 38.31

REML score: 2754.274
```

According to the above output, we can observe that the *Backward* model has dropped the first, second and fifth lags of electricity demand, as well as the fourth, fifth and sixth lags of temperature.

Next, we estimate a groupwise additive index model (*GAIM*) using the function `model_gaim()`. As this model does not perform variable selection or parameter tuning, a separate validation set is not required. To ensure comparability of forecasting performance with the SMI models and the *Backward* model (which are re-fitted on the data set combining the training and validation sets after penalty parameter tuning or model selection), we estimate the *GAIM* directly on the combined training and validation data set.

```
# Data set combining training and validation
combinedData <- bind_rows(train, val)

# Fit groupwise additive index model
gaim <- model_gaim(data = combinedData,
                   yvar = "Demand_lag_000",
                   index.vars = colnames(vic_elec_daily)[5:19],
                   index.ind = c(rep(1, 7), rep(2, 8)),
                   linear.vars = c("DOW", "Holiday"))
```

The new argument used here is `index.ind`. Since predictors must be pre-grouped when fitting a *GAIM*, we group all the lagged electricity demand variables into one group and all the lagged temperature variables into another, resulting in two indices for the model. This grouping of the variables should be specified using the `index.ind` argument, which should be a `numeric` vector assigning a group index to each predictor in the `index.vars` argument. Accordingly, we pass a vector containing seven ones (indicating that the first seven variables—i.e. the seven demand lags—should enter the first index) and eight twos (indicating that the next eight variables—i.e. the current temperature and the seven temperature lags—should enter the second index) in the `index.ind` argument.

As no validation set forecasting occurs when estimating a *GAIM*, the function call does not include the `recursive` or `recursive_colRange` arguments.

```
# Returned gaimFit object
gaim
```

```
Fitted Model(s):
# A tibble: 1 x 2
  key   fit
  <chr> <list>
1 1     <cgaim>
```

The model object returned from `model_gaim()` is of the class `gaimFit`, which is again a `tibble` with the two usual columns. Each element in the `fit` column is a `cgaim` object (Masselot et al. 2023), which can be accessed as follows.

```
# Fitted gaim model
gaim$fit[[1]]
```

```
Formula:
Demand_lag_000 ~ g(Demand_lag_001, Demand_lag_002, Demand_lag_003,
    Demand_lag_004, Demand_lag_005, Demand_lag_006, Demand_lag_007) +
    g(Temp_lag_000, Temp_lag_001, Temp_lag_002, Temp_lag_003,
        Temp_lag_004, Temp_lag_005, Temp_lag_006, Temp_lag_007) +
    DOW + Holiday

Coefficients:
   (Intercept) Demand_lag_001   Temp_lag_000            DOW        Holiday
    225.712161      21.821048       8.056363      14.871267       5.677247

Indices weights:
Demand_lag_001
Demand_lag_001 Demand_lag_002 Demand_lag_003 Demand_lag_004 Demand_lag_005
    0.66380425    -0.12333619     0.07650766     0.03304612    -0.02171711
Demand_lag_006 Demand_lag_007
    0.07156957    -0.01001911
Temp_lag_000
```

```
Temp_lag_000 Temp_lag_001 Temp_lag_002 Temp_lag_003 Temp_lag_004 Temp_lag_005
  0.41375636  -0.14374285  -0.09616561   0.04855670  -0.13736280   0.05307141
Temp_lag_006 Temp_lag_007
 -0.03257577  -0.07476850

Residual sum of squares: 72.65604
```

Finally, we fit a projection pursuit regression (*PPR*) model using the function `model_ppr()`.

```
# Fit projection pursuit regression model
ppr <- model_ppr(data = combinedData,
                 yvar = "Demand_lag_000",
                 index.vars = colnames(vic_elec_daily)[5:19],
                 num_ind = 5)
```

As mentioned earlier, when estimating a *PPR* model, we need to specify the number of indices to be estimated, which in this case is done using the argument `num_ind`. Here, we set the number of indices as five to match the number of indices in the *PPR* model used to initialise *SMI-P*.

```
# Returned pprFit object
ppr
```

```
Fitted Model(s):
# A tibble: 1 x 2
  key   fit
  <chr> <list>
1 1     <ppr.form>
```

The model object returned from `model_ppr()` is of the class `pprFit`, which is once again a `tibble` with two columns. Each element in the `fit` column is a `ppr.form` object (see `stats::ppr()`), which is a `list` containing information about the fitted model. The estimated index coefficients can be found as follows.

```
# Estimated index coefficients
ppr$fit[[1]]$alpha
```

```
                      term 1        term 2       term 3       term 4
Demand_lag_001  0.2711403542 -0.0215396033 -0.043452028 -0.142853293
Demand_lag_002 -0.1484608094  0.0278252642 -0.017158869  0.168480327
```

```
Demand_lag_003  0.0788787900 -0.0081368260 -0.032387660 -0.029176754
Demand_lag_004 -0.0056658336  0.0044392121 -0.003955462  0.073975586
Demand_lag_005 -0.0646555587  0.0228034834 -0.038391788  0.032850229
Demand_lag_006  0.1071655387  0.0215746246  0.017709905 -0.003293433
Demand_lag_007  0.1140027983  0.0482778325  0.046710673  0.053517469
Temp_lag_000    0.7189727625 -0.9169970372 -0.749596364  0.176321522
Temp_lag_001   -0.1892982047  0.0136004812  0.313072502 -0.095062901
Temp_lag_002    0.0160559732 -0.0982028035  0.148056240 -0.521506459
Temp_lag_003    0.0008285177  0.0088597218  0.026484296  0.165651261
Temp_lag_004   -0.2049374216 -0.0006327645  0.056418348  0.153989309
Temp_lag_005    0.2493395900 -0.2673720004  0.044845986 -0.525606536
Temp_lag_006   -0.2300761660  0.1703408385  0.234601165  0.450248530
Temp_lag_007   -0.3992595084  0.2099158410  0.500020363  0.314158086
                     term 5
Demand_lag_001  0.06799731
Demand_lag_002 -0.06938484
Demand_lag_003  0.12344298
Demand_lag_004 -0.16036131
Demand_lag_005  0.06733398
Demand_lag_006 -0.02939108
Demand_lag_007  0.02888241
Temp_lag_000   -0.40678190
Temp_lag_001    0.40792519
Temp_lag_002    0.50351755
Temp_lag_003   -0.39333495
Temp_lag_004    0.27181367
Temp_lag_005    0.02165478
Temp_lag_006   -0.29916427
Temp_lag_007   -0.19851613
```

Obtaining forecasts

Next, we look at how to obtain forecasts on the test set from the models we fitted above. We use the `predict()` function for this purpose, which will call the relevant S3 method for each corresponding model class.

For demonstration purposes, we obtain forecasts on the test set using *SMI-P* as follows. In this case, the `predict()` function calls the `predict.smimodel` method.

```
# Obtaining forecasts of the test set using SMI-P
predictions_smimodel_ppr <- predict(object = smimodel_ppr,
                                    newdata = test,
                                    recursive = TRUE,
                                    recursive_colRange = 5:11)
```

Here, the `object` argument takes the fitted model object, and the `newdata` argument takes the data set on which we need to obtain forecasts. The `recursive` and `recursive_colRange` arguments function similarly to those in the model-fitting functions, as we also perform autoregressive forecasting on the test set. The use of the `predict()` function is the same for all the other models we fitted above.

This call to the `predict()` function returns a `tsibble` that contains the `newdata`, along with additional columns generated during the forecasting process. The `.predict` column in this `tsibble` stores the generated forecasts. We will extract and print the response variable data and the generated forecasts below.

```
# Forecasts
predictions_smimodel_ppr[ , c("Demand_lag_000", ".predict")]
```

```
# A tibble: 306 x 2
   Demand_lag_000 .predict
            <dbl>    <dbl>
 1           194.     186.
 2           184.     177.
 3           228.     214.
 4           255.     243.
 5           235.     237.
 6           217.     226.
 7           216.     219.
 8           195.     199.
 9           201.     205.
10           214.     218.
# i 296 more rows
```

Once we have forecasts on the test set, we can compute forecast error measures to evaluate the forecasting performance of the estimated models. In the `smimodel` package, we currently have two functions, `MSE()` and `MAE()`, which correspond to two commonly used error measures, Mean Squared Error and Mean Absolute Error, respectively. If other error measures are required, users of `smimodel` currently need to rely on other packages.

We demonstrate the usage of the aforementioned functions using *SMI-P* as follows. The forecast errors on the test set must be provided in the `residuals` argument.

```
# Forecast error measures
actual <- predictions_smimodel_ppr$Demand_lag_000
predictions <- predictions_smimodel_ppr$.predict
MSE(residuals = actual - predictions)
```

```
[1] 80.63169
```

```
MAE(residuals = actual - predictions)
```

```
[1] 6.563263
```

We can calculate the forecast errors for all the models we estimated above in the same way, and Table 4.1 below summarises the results.

Table 4.1: *Out-of-sample point forecast results for daily electricity demand. The columns* Model, Predictors *and* Indices *indicate the name of the fitted model, the number of predictor variables used/selected (if predictor selection is performed), and the number of indices estimated by the fitted model, respectively.*

			Test Set	
Model	Predictors	Indices	MSE	MAE
SMI-P	17	3	80.632	6.563
SMI-A	17	3	80.999	6.789
SMI-L	16	2	310.327	14.731
Backward	11		**72.166**	**6.331**
GAIM	17	2	13990.017	104.170
PPR	15	5	477.330	16.259

Based on the results above, the *Backward* model demonstrates the strongest forecasting performance in this example. However, the SMI models estimated using both “ppr” and “additive” initialisations

(i.e. *SMI-P* and *SMI-A*) produce forecast errors that are close to those of the best-performing model. This finding is consistent with our observations in the empirical applications discussed in Chapter 2, where the *Backward* model again achieved the best performance, and at least one of the SMI models—estimated with different initialisations—performed similarly well.

Here, the *GAIM* exhibits notably large forecast errors. Throughout our experiments in this research, we have observed that the forecasting performance of *GAIM* is inconsistent. In Chapter 2, it delivered the second-best performance among the three benchmark models and outperformed *SMI-A* and *SMI-L* in the mortality forecasting task. However, it yielded the poorest results in the solar intensity forecasting application. This inconsistency is particularly evident at longer forecast horizons, where forecasts from *GAIM*s can become unstable and errors accumulate rapidly—especially in autoregressive settings. This may explain the current result, as we use a relatively large test set here (forecasting 10 months ahead) in an autoregressive context.

Uncertainty estimation

Next, we explore the functionalities that `smimodel` provides for generating prediction intervals to estimate the uncertainty of the point forecasts obtained using the nonparametric additive models.

As mentioned in Section 4.2.3, the `smimodel` package implements the block bootstrap (BB) and conformal bootstrap (CB) methods discussed in Chapter 3 for generating prediction intervals.

First, we demonstrate the usage of the `bb_cvforecast()` function to apply the BB method.

In the `bb_cvforecast()` function, the `object` argument takes the fitted model object, and the `data` argument takes the data set. The full set of predictors used when fitting the model (i.e. index variables, non-linear variables and linear variables) should be specified in the `predictor.vars` argument, and the forecast horizon is passed to the `h` argument.

The `season.period` argument specifies the length of the seasonal period if the residual series of the fitted model exhibits seasonality. This value is used to determine the block size in the BB method. As explained in Section 3.2.2, if the residual series is non-seasonal, the last significant lag of the ACF of the residuals, subject to a maximum number of lags considered, is used as the block size, which should be passed to `season.period`.

Here, we use the above estimated *SMI-P* as an example, and use the `feasts::ACF()` (O'Hara-Wild, Hyndman & Wang 2024) function below to obtain the ACF plot of the residual series of the fitted model.

```
# Preparing a tsibble with the residuals series of SMI-P
res <- tibble(
  index = seq(1, length(smimodel_ppr$fit[[1]]$best$gam$residuals)),
  SMIModel_PPR = smimodel_ppr$fit[[1]]$best$gam$residuals) |>
  as_tsibble(index = index)

# ACF of the residuals series of SMI-P
acf_smimodel_ppr <- feasts::ACF(.data = res,
                                y = res$SMIModel_PPR,
                                lag_max = 790)
```

```
# ACF plot
acf_smimodel_ppr |>
  autoplot() +
  labs(x = "Lag", y = "ACF") +
  theme_bw()
```

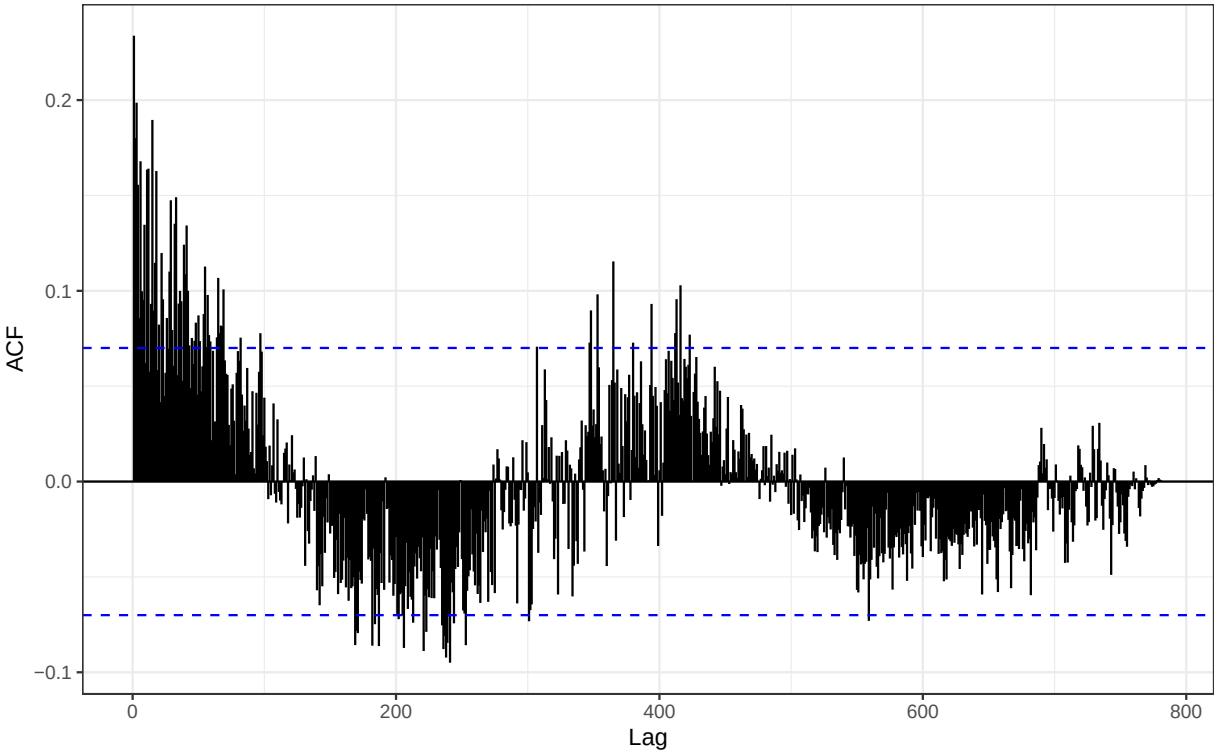


Figure 4.2: *The ACF plot of the residual series of SMI-P*

According to the ACF plot above, it appears that the estimated *SMI-P* has not sufficiently captured the autocorrelation present in the data, as the ACF remains significant for a large number of lags, including the first few. This also leads to some seasonal variation remaining in the residuals.

In this particular example, this observation is understandable, as the selected ℓ_2 penalty parameter, λ_2, for the model was 10000—a very large value that caused the estimated coefficients for the lagged electricity demand variables to be heavily shrunk. The excessive shrinkage imposed has resulted in smaller coefficient estimates, limiting the ability of the lagged variables to effectively capture the autocorrelation structure.

Therefore, in this case, it is not practical to determine the block size based on the significant lags of the ACF plot. If we attempt to set the block length as the seasonal period length, it would be too long, preventing us from constructing a sufficient number of blocks from the available data to generate many simulated residual series. Thus, applying the BB method in this example is not very meaningful.

However, simply to demonstrate the use of arguments, we present the call for the `bb_cvforecast()` function below.

```
# Set a seed
set.seed(12345)
# BB prediction intervals through time series cross-validation
bb <- bb_cvforecast(object = smimodel_ppr,
                    data = vic_elec_daily,
                    yvar = "Demand_lag_000",
                    predictor.vars = colnames(vic_elec_daily)[c(5:19, 2:3)],
                    h = 30,
                    season.period = 1,
                    num.futures = 1000,
                    level = c(80, 95),
                    window = NROW(train) + NROW(val),
                    recursive = TRUE,
                    recursive_colNames = colnames(vic_elec_daily)[5:11])
```

Note that, in this case, we specify `h = 30`, indicating that we perform 30-step-ahead forecasting. Moreover, we have specified `season.period = 1`, in which case the BB method reduces to standard bootstrapping. The `num.futures` argument specifies the number of possible future sample paths (or the number of bootstrap series) to generate, while the `level` argument sets the confidence level(s) for the prediction intervals.

The `window` argument specifies the rolling window length used in time series cross-validation. If set to `NULL`, an expanding window approach is employed instead. In this case, we use a rolling window strategy, with the rolling training window length set equal to that of the initial training set used for estimating the SMI model (including penalty parameter tuning—that is, the original training plus validation sets). It is important to note that at each cross-validation step, there is a training window used for model fitting, and a separate test window of length 30 (as `h = 30`) which is not used in model training. Prediction intervals are generated for this test window, as described in Section 3.2.2.

The column names of the lagged response variables in the data set are passed to the `recursive_colNames` argument. (See the package documentation for details on other arguments not explicitly mentioned in the function call above, as default values are used.)

It is also necessary to set a seed before calling the `bb_cvforecast()` function to ensure reproducibility, as random resampling is used in the block bootstrap procedure.

The `bb_cvforecast()` function returns an object of class `bb_cvforecast`, which is a list containing 10 elements. The `mean` element contains the point forecasts generated at each cross-validation step as a multivariate time series, where the h^{th} column contains the point forecasts for the forecast horizon h. The time index corresponds to the period for which the forecast is produced. The `lower` and `upper` elements are lists containing the lower and upper bounds, respectively, for the prediction intervals generated at each confidence level and can be accessed as follows.

```
# Lower bounds of the 95% prediction intervals
bb$lower$`95%`
# Upper bounds of the 95% prediction intervals
bb$upper$`95%`
```

Each of these is a multivariate time series with the same dimensional characteristics as `mean`. For details on the other elements in a `bb_cvforecast` object, refer to the package documentation.

Next, we explore how to generate conformal bootstrap (CB) prediction intervals using the `cb_cvforecast()` function.

```
# Set a seed
set.seed(12345)
# CB prediction intervals through time series cross-validation
cb <- cb_cvforecast(object = smimodel_ppr,
                    data = vic_elec_daily,
                    yvar = "Demand_lag_000",
```

```
                    predictor.vars = colnames(vic_elec_daily)[c(5:19, 2:3)],
                    h = 30,
                    ncal = 100,
                    num.futures = 1000,
                    level = c(80, 95),
                    window = NROW(train) + NROW(val),
                    recursive = TRUE,
                    recursive_colNames = colnames(vic_elec_daily)[5:11])
```

The key additional argument in `cb_cvforecast()` is `ncal`, which specifies the length of the rolling calibration window used to compute non-conformity scores (i.e. out-of-sample forecast errors) at each cross-validation step. Note that each cross-validation step involves three distinct windows: a training window (`window`) for model fitting, a calibration window (`ncal`) for calculating non-conformity scores, and a test window (`h`) for generating prediction intervals. The calibration window is not used for model training, and the test window is used solely for evaluating prediction intervals—it is not involved in either model fitting or the calculation of non-conformity scores.

As with `bb_cvforecast()`, it is important to set a seed before calling the `cb_cvforecast()` function to ensure reproducibility.

The `cb_cvforecast()` function returns an object of the class `cb_cvforecast`, which has a similar structure to `bb_cvforecast`. See the package documentation for details.

To calculate the mean (actual) coverage and the mean width of the constructed prediction intervals, we can use the `avgCoverage()` and `avgWidth()` functions, respectively. The `mean` element in the output of both functions provides the mean coverage and mean width, respectively. For details on the other elements of the output, see the package documentation. Note that, as previously mentioned, `avgCoverage()` and `avgWidth()` are wrappers for `conformalForecast::coverage()` and `conformalForecast::width()` (Wang & Hyndman 2025), respectively.

```
# Mean coverage
coverage <- avgCoverage(object = cb, level = 95)
coverage$mean
# Mean width
width <- avgWidth(object = cb, level = 95)
width$mean
```

4.4 Comparison of computational performance

As mentioned in Section 4.1, several R packages are available for estimating nonparametric additive models and distributed lag models, both of which are related to the SMI model—the main focus of the `smimodel` package. In this section, we compare the computational performance of the main SMI model estimation function `smimodel::greedy_smimodel()`, with three different models from three commonly used packages: `mgcv` (Wood 2023, 2011), `dLagM` (Demirhan 2020), and `dlnm` (Gasparrini 2011). We continue using the electricity demand forecasting example for this comparison as well.

First, we consider the estimation of SMI models with the three initialisation options (`"ppr"`, `"additive"`, and `"linear"`) we presented previously, as the run time and memory usage may differ based on the initialisation option used.

Next, we estimate a generalised additive model (GAM) using the `mgcv::gam()` function, where we treat the seven electricity demand lags, current temperature, and seven temperature lags as non-linear variables, and the two calendar variables, *DOW* and *Holiday*, as linear variables.

Then, an autoregressive distributed lag model (ARDL) is estimated using the `dLagM::ardlDlm()` function, which is a linear model.

Next, we fit a distributed lag non-linear model (DLNM) using the `dlnm` package. The `dlnm` package does not provide a single function to fit a DLNM directly. Therefore, we first use the `dlnm::crossbasis()` function to generate two cross-basis matrices: one for the first lag of electricity demand and six lags of it (i.e. seven demand lags are included as predictors), and another for the current temperature and its seven lags (i.e. current temperature and seven temperature lags are included in the model as predictors). Then, we use the constructed cross-basis matrices to fit a GAM (using `mgcv::gam()`), while taking the two calendar variables as linear variables. We combine the above two steps into a new function as follows for this comparison. Note that we use penalised cubic regression splines in both `mgcv::gam()` and `dlnm::crossbasis()` (option `"cr"`) to match the type of splines we use in estimating SMI models.

```
# A new function to estimate a DLNM using `dlnm` package
dlnm_function <- function(){
  # Basis matrix for demand lags
  cb.demand <- dlnm::crossbasis(x = combinedData$Demand_lag_001,
                                lag = 6,
                                argvar = list(fun = "cr"),
                                arglag = list(fun = "cr"))
```

```
  # Basis matrix for temperature lags
  cb.temp <- dlnm::crossbasis(x = combinedData$Temp_lag_000,
                              lag = 7,
                              argvar = list(fun = "cr"),
                              arglag = list(fun = "cr"))
  mgcv::gam(formula = Demand_lag_000 ~ cb.demand + cb.temp + DOW + Holiday,
            data = combinedData,
            method = "REML")
}
```

In addition, we also compare `smimodel::greedy_smimodel()` with a groupwise additive index model (GAIM) fitted using the `cgaim::cgaim()` (Masselot et al. 2023), although the `cgaim` package was not originally designed for forecasting or prediction purposes. Similarly to the GAIM fitted in Section 4.3.3, we group the seven electricity demand lags into one group, and the current temperature and the seven temperature lags into another group, resulting in two indices for the GAIM estimated. The two calendar variables are treated as linear variables, as usual.

We use the `bench::mark()` function to perform the intended comparison, through 10 runs of each model-fitting expression as follows. This is run on an Apple Silicon MacBook Pro laptop with an Apple M1 Pro chip and 16 GB of memory.

```
# Using `bench;:mark()`
comparison <- mark(
  # smimodel::greedy_smimodel()
  # SMI-P
  greedy_smimodel(data = train,
                  val.data = val,
                  yvar = "Demand_lag_000",
                  index.vars = colnames(vic_elec_daily)[5:19],
                  linear.vars = c("DOW", "Holiday"),
                  nlambda = 20,
                  recursive = TRUE,
                  recursive_colRange = 5:11),
  # SMI-A
  greedy_smimodel(data = train,
                  val.data = val,
```

```
                yvar = "Demand_lag_000",
                index.vars = colnames(vic_elec_daily)[5:19],
                linear.vars = c("DOW", "Holiday"),
                initialise = "additive",
                nlambda = 20,
                recursive = TRUE,
                recursive_colRange = 5:11),
# SMI-L
greedy_smimodel(data = train,
                val.data = val,
                yvar = "Demand_lag_000",
                index.vars = colnames(vic_elec_daily)[5:19],
                linear.vars = c("DOW", "Holiday"),
                initialise = "linear",
                nlambda = 20,
                recursive = TRUE,
                recursive_colRange = 5:11),
# mgcv::gam()
gam(formula = Demand_lag_000 ~ s(Demand_lag_001, bs = "cr") +
      s(Demand_lag_002, bs = "cr") + s(Demand_lag_003, bs = "cr") +
      s(Demand_lag_004, bs = "cr") + s(Demand_lag_005, bs = "cr") +
      s(Demand_lag_006, bs = "cr") + s(Demand_lag_007, bs = "cr") +
      s(Temp_lag_000, bs = "cr") + s(Temp_lag_001, bs = "cr") +
      s(Temp_lag_002, bs = "cr") + s(Temp_lag_003, bs = "cr") +
      s(Temp_lag_004, bs = "cr") + s(Temp_lag_005, bs = "cr") +
      s(Temp_lag_006, bs = "cr") + s(Temp_lag_007, bs = "cr") +
      DOW + Holiday,
    data = combinedData,
    method = "REML"),
# cgaim::cgaim()
cgaim(formula = Demand_lag_000 ~ g(Demand_lag_001, Demand_lag_002,
                                   Demand_lag_003, Demand_lag_004,
                                   Demand_lag_005, Demand_lag_006,
                                   Demand_lag_007) +
```

```
          g(Temp_lag_000, Temp_lag_001, Temp_lag_002, Temp_lag_003,
            Temp_lag_004, Temp_lag_005, Temp_lag_006, Temp_lag_007) +
          DOW + Holiday,
        data = combinedData),
  # dLagM::ardlDlm()
  ardlDlm(formula = Demand_lag_000 ~ Temp_lag_000 + DOW + Holiday,
          data = combinedData,
          p = 7, q = 7,
          remove = list(p = list(DOW = 1:7, Holiday = 1:7))),
  # dlnm
  dlnm_function(),

  min_time = Inf,
  iterations = 10,
  max_iterations = 10,
  check = FALSE
)
```

The output from the above function call provides measurements of run time and memory usage for the models compared, and is summarised in Table 4.2.

Table 4.2: *Summarised information on the computational performance of smimodel::greedy_smimodel() and functions from other packages chosen for comparison.*

	Run Time			
Expression	Minimum	Median	Total	Memory Allocation
smimodel::greedy_smimodel() - PPR	13.7m	13.95m	2.36h	148.8GB
smimodel::greedy_smimodel() - Additive	29.49m	30.1m	5.02h	200.6GB
smimodel::greedy_smimodel() - Linear	3.5m	3.54m	36.11m	48.5GB
mgcv::gam()	3.4s	3.41s	34.12s	283.4MB
cgaim::cgaim()	1.49s	1.54s	16s	249.4MB
dLagM::ardlDlm()	28.37ms	28.89ms	302.1ms	28.2MB
dlnm_function()	677.91ms	680.58ms	6.81s	107.3MB

According to Table 4.2, the estimation of the SMI model with simultaneous penalty parameter tuning using the `smimodel::greedy_smimodel()` function is computationally expensive compared to other models, recording the highest run time and total memory allocation over 10 runs. This outcome is expected due to the limited input information provided on the model structure, the large number of MIP iterations performed, and the estimation of several non-linear functions. Here, it is also important to note that the inclusion of the ℓ_0 penalty renders the SMI model estimation problem NP-hard. As a result, `smimodel::greedy_smimodel()` undertakes a particularly complex and computationally intensive estimation task.

Notably, as observed previously in Chapter 2, SMI model estimated with `"additive"` initialisation requires more time and memory than other initialisation options. The computational cost of SMI model estimation depends on the number of index variables used and the number of indices in the model. Since the "additive" initialisation constructs the model using a nonparametric additive structure, where each variable is assigned to a separate index, the number of indices to be estimated is greater than that of the `"ppr"` or `"linear"` options, thereby leading to higher computational costs.

4.5 Conclusion

This chapter has introduced the R package `smimodel`, a new open-source software designed for nonparametric forecasting and prediction. While there are several R packages available for estimating nonparametric additive models, to the best of our knowledge, none support principled predictor selection and grouping, simultaneously with model estimation. Typically, achieving this requires expert input, especially in high-dimensional settings. `smimodel` addresses this gap by providing an automated implementation of the *Sparse Multiple Index (SMI) Modelling* algorithm, which automatically selects and groups predictors, simultaneously with the estimation of a nonparametric additive index model, through a user-friendly interface. Thus, `smimodel`—as a statistical forecasting/prediction tool—has the potential to improve the quality and the objectivity of nonparametric model estimation. It has broad applicability in various domains, from academic research to industry, and can be effective even for forecasters without specialised domain knowledge in their particular application of interest.

In addition, the package provides implementations of two prediction interval generation methods: block bootstrap (BB) and conformal bootstrap (CB). These methods enable uncertainty estimation for point forecasts generated by the nonparametric models available in `smimodel`. We believe that this combination of model estimation and prediction interval generation capabilities offers a comprehensive workflow for time series forecasting. It is particularly useful for addressing challenging

real-world forecasting problems that involve a large number of predictor variables with non-linear and complex interactions, using a statistical approach.

One of the main limitations of `smimodel` is the computational cost involved in SMI model estimation, due to the challenging estimation task it faces with the limited prior information about model structure. As the `smimodel` package is actively developed and maintained, we are continuously working on optimising the functions and researching improvements to the algorithm to reduce the computational cost, especially in high-dimensional settings.

As demonstrated in Section 4.3.3, the model fitting functions in `smimodel` currently require users to specify the names of the response and predictor variables as character strings through separate arguments. To improve usability and consistency with existing regression-type modelling packages in R, we plan to adopt a formula interface that accepts a standard `formula` object, replacing the current argument-based approach.

Moreover, when lags of the response variable are used as predictors in time series forecasting problems, users of `smimodel` currently need to explicitly include these lagged variables in the data set provided to the model fitting functions. We plan to enhance `smimodel` by enabling the automatic generation of lagged variables within the model fitting functions, taking away this additional task from the users in the future.

The current version of `smimodel` offers limited functionality for visualising various aspects of the estimated models, apart from the `autoplot.smimodel` method for plotting estimated smooth terms in a SMI model. To improve model analysis and interpretation, we plan to introduce additional functions to generate useful visualisations.

Chapter 5

Conclusion

This thesis investigates nonparametric additive index models as a flexible and comprehensible alternative to traditional time series forecasting methods, which typically rely on strong parametric assumptions and time series-specific structures. The proposed models capture non-linear relationships by expressing the forecast target as a function of one or more linear combinations of predictors, without relying on strict distributional assumptions or the underlying characteristics of the time series. Although these models are based on a regression framework and not inherently designed for time series forecasting, we demonstrate their potential in real-world applications. This final chapter provides an overview of the thesis, critically evaluating the proposed methodologies and key findings. It also discusses the limitations of the study and suggests avenues for future research.

5.1 Discussion of results and contributions

The challenges outlined at the beginning of this thesis—the presence of non-linear predictor-response relationships, high-dimensionality, and the need for transparency in model structure and estimation—have shaped the direction and contributions of this research.

In recent years, nonparametric additive models have been used as a predictive tool in several studies across different application areas (Fan & Hyndman 2012; Du, Cheng & Liang 2012; Goude, Nedellec & Kong 2014; Ho, Chen & Hwang 2020; Boente & Martínez 2023). However, existing approaches to model estimation often depend on manual intervention, where decisions about which variables to include and how to model their interactions are largely informed by expert intuition, rather than by objective, data-driven methods.

Therefore, the primary objective of Chapter 2 of this thesis was to introduce a modelling framework based on nonparametric additive index models that is capable of capturing complex predictor interactions while automating both variable selection and grouping. To this end, we proposed the *Sparse Multiple Index (SMI) Modelling* algorithm. This is an iterative procedure that employs mixed integer programming to solve an ℓ_0- and ℓ_2-regularised non-linear least squares optimisation problem, subject to linear constraints, for the estimation of a nonparametric additive index model.

The proposed SMI Modelling algorithm is distinguished by its structural flexibility. Unlike traditional statistical models, where the structure of the model is predetermined, the SMI model does not require the user to predefine its index structure—namely, the number of indices and the assignment of predictor variables to each index. Instead, the algorithm automatically identifies the optimal index structure in conjunction with model estimation. To initiate the iterative procedure, the user simply provides a set of candidate predictors and an initial model structure. Another notable feature of the algorithm is its ability to perform automatic variable selection without manual input. Given a pool of potential predictors entering indices, the algorithm determines the most relevant subset for inclusion in the final model.

Thus, this study offers three fundamental contributions. First, the proposed SMI Modelling algorithm introduces automatic predictor selection for nonparametric additive index model estimation, reducing reliance on user domain expertise. To our knowledge, this is the first instance of such a capability being incorporated into a nonparametric additive index model estimation algorithm. Second, owing to its flexible structure, the SMI model encompasses a wide range of models—from single index models (where all predictors are combined into a single index) to additive models (where each predictor has its own index). Furthermore, it accommodates additional non-linear and linear predictors, enabling semiparametric model estimation. Hence, the SMI Modelling algorithm serves as a general estimation tool for the broader class of nonparametric and semiparametric additive models. Third, although employed as a forecasting tool in this study, the SMI model does not rely on time series-specific features. As a nonparametric regression model, it can be applied to a variety of prediction tasks, including those involving cross-sectional data. This broad applicability positions the SMI model as a versatile solution for numerous real-world predictive problems.

In Chapter 2, we demonstrated the use and performance of the proposed SMI model through two empirical applications—forecasting heat exposure-related daily mortality and daily solar intensity. We compared the forecasting accuracy of the estimated SMI models against three benchmark models. In both cases, the SMI models were unable to outperform the nonparametric additive model estimated using backward elimination for variable selection (Backward), while the other benchmark models did

not exhibit consistent performance across the two applications. However, in each case, at least one of the SMI models (among those estimated with different initialisation options) achieved forecasting performance close to that of the Backward model.

This raises a point of discussion regarding the possible reasons for the SMI model to exhibit inferior forecasting performance compared to the Backward model. A key difference lies in model structure—the Backward model, being a nonparametric additive model, has a predefined structure, and employs a straightforward backward elimination procedure for variable selection. As a result, the Backward model follows a clear and direct estimation process. In contrast, as explained earlier, the SMI model adopts a much more flexible approach to determining the index structure. This structure is identified automatically through a data-driven algorithm that simultaneously performs predictor selection and model estimation. Consequently, the estimation process for the SMI model is inherently more complex and introduces additional optimisation challenges. This may explain why the SMI model performs comparably but not better than the simpler Backward model.

A second consideration is the impact of the ℓ_0 and ℓ_2 penalty terms in the estimation of the SMI model, as they have a substantial effect on its forecasting accuracy. Accordingly, the tuning of the corresponding penalty parameters, λ_0 and λ_2, plays a critical role. Our experiments revealed that, at times, the selected penalty parameter values—especially λ_2, associated with the ridge penalty—are too large, resulting in excessive shrinkage of the estimated coefficients toward zero. This suggests that the current method for determining the grid of candidate penalty values may be too dispersed. As a result, the greedy search procedure used for tuning may fail to explore the optimal region of the parameter space effectively. We believe that refining this search strategy to better identify the ideal range of penalty parameter values would improve the forecasting performance of the SMI model. We leave this refinement as a direction for future research.

Once a forecasting model capable of generating point forecasts is in place, the next step in the forecasting workflow is to construct prediction intervals to quantify forecast uncertainty. Accordingly, Chapter 3 of this thesis focused on exploring prediction interval construction methods for nonparametric additive index models.

This part of the study makes two key contributions. First, it evaluates the applicability of block bootstrapping (BB) and three existing conformal prediction (CP) methods—Split Conformal Prediction (SCP), Weighted Split Conformal Prediction (WSCP), and Adaptive Conformal Prediction (ACP)—for generating prediction intervals in the context of nonparametric additive index models. Second, it proposes a novel method named *Conformal Bootstrap (CB)*, which combines elements of both block bootstrapping and split conformal prediction to construct prediction intervals.

In Chapter 3, we assessed the empirical coverage performance of each method using the same two real-world forecasting problems presented in Chapter 2. For the heat exposure-related daily mortality application, we considered 92-step-ahead forecasts, and for the daily solar intensity application, we considered 30-step-ahead forecasts. In each case, we evaluated the average empirical coverage of the 95% prediction intervals generated by each method across the full forecast horizon. The results showed that BB intervals often led to under-coverage, except for the GAIM and Backward models in the solar intensity forecasting task. In contrast, the three CP methods generally produced the most reliable long-run coverage across both applications.

The proposed CB method, which combines block bootstrap with split conformal prediction, was expected to achieve coverage performance equal to or better than each of its component methods. In the mortality forecasting application, it generally delivered empirical coverage similar to the CP methods at shorter forecast horizons and comparable to the BB method at longer horizons. In the solar intensity forecasting problem, it performed similarly to the CP methods across all horizons for the SMI and Backward models. However, it exhibited notable under-coverage for the PPR model—potentially due to a combination of factors: overfitting arising from the model's dense structure relative to the other, more parsimonious models; the use of only one-step-ahead forecast errors, as opposed to the multi-step errors used in the CP methods that demonstrated better coverage; and the truncation of predictors during the generation of sample paths. Further investigation is required to better understand the factors contributing to the underperformance of the CB method in certain settings.

After establishing the methodological framework for time series forecasting using nonparametric additive index models in Chapter 2 and Chapter 3, the next stage was to facilitate the use of these proposed methods by making them available to potential users. To achieve this, the development of open-source software that automates the SMI Modelling algorithm became the central focus of Chapter 4 in the thesis.

The primary contribution of Chapter 4 is `smimodel`—an open-source R package developed to implement the SMI Modelling algorithm for estimating nonparametric additive index models. While several existing R packages offer tools for estimating various types of nonparametric additive and distributed lag models—two model classes closely related to the SMI model—none provide the integrated capabilities of automated predictor selection, model structure determination, and model estimation. Moreover, `smimodel` incorporates two methods for constructing prediction intervals, BB and CB, which enable uncertainty estimation for point forecasts generated from the nonparametric additive models offered in the package. Although there is room for improvement, as discussed in

Section 4.5, we believe `smimodel` marks a significant step towards making advanced nonparametric modelling accessible to a wider audience, including users without deep statistical expertise.

5.2 Future work

This section explores future research directions informed by the current study. While some of these ideas were briefly discussed at the end of Chapters 2 to 4 to maintain the coherence of each chapter, this section expands and builds upon them.

A key limitation of the proposed SMI Modelling algorithm is its lack of a generalised initialisation option that guarantees optimal performance across diverse data sets and applications. Throughout the experiments conducted in this thesis, it became evident that the effectiveness of the algorithm's initialisation is highly data-dependent. Consequently, the choice of initialisation strategy plays a pivotal role in determining forecasting accuracy. Therefore, an important direction for future research would be to explore and develop an initialisation method that can adapt to various types of data, ensuring that the SMI Modelling algorithm consistently delivers strong forecasting performance across a broad range of applications. Such a study would not only enhance the model's generalisability but also improve its applicability in real-world scenarios, where data characteristics can vary considerably.

Alternatively, another avenue worth exploring is the use of ensembles of SMI models estimated using different initialisations. By estimating multiple models with varying initialisations and combining their forecasts—either through averaging or weighted ensembling—it may be possible to improve the robustness and stability of the model, particularly in data sets with complex structures. This approach could mitigate the algorithm's sensitivity to the chosen initialisation and provide better forecasting performance in practical applications.

Another limitation in estimating a SMI model is the computational cost, especially when dealing with many predictors and indices. The complexity of the model increases with dimensionality, leading to longer processing times and greater demands on computational resources. Thus, addressing these computational challenges is crucial for ensuring the algorithm's practical usability. An important research challenge is to explore enhancements to the proposed algorithm, both methodologically and programmatically, to improve the computational efficiency of the model estimation process. Achieving this would enable the application of the SMI model to larger data sets and more complex applications, thereby increasing its utility in diverse domains where high-dimensional data is prevalent.

We acknowledge that the applications presented in this thesis are not diverse or challenging enough to draw generalised conclusions about the forecasting performance of the proposed SMI model.

Therefore, an interesting direction for future research would be to evaluate the SMI Modelling algorithm across a wide-ranging problem space, where the data sets display a broad spectrum of characteristics (Kang, Hyndman & Smith-Miles 2017). Additionally, it would be valuable to investigate how different types of problems, each with distinct structural properties, impact the forecasting performance of the SMI Modelling algorithm. This would help identify the conditions under which the algorithm outperforms benchmark methods, as well as the circumstances where its limitations become apparent. Conducting an instance space analysis (Smith-Miles et al. 2014) would provide deeper insights into how the algorithm can be further refined, offering potential improvements not only in forecasting accuracy but also in the computational efficiency discussed earlier.

The SMI Modelling algorithm is an iterative method based on mixed integer programming, designed to solve a non-linear least squares optimisation problem with ℓ_0 and ℓ_2 regularisation under linear constraints, where the ℓ_0 penalty introduces sparsity into the estimation process through best subset selection. A potential avenue for future research would be to examine the effectiveness of alternative regularisation strategies, particularly combining ℓ_0 and ℓ_1 penalties, or employing an elastic net penalty (Zou & Hastie 2005)—i.e. a combination of ℓ_1 and ℓ_2 penalties—for variable selection (Mazumder, Radchenko & Dedieu 2023; Hazimeh & Mazumder 2020) in the context of nonparametric additive index models. These alternative strategies were not explored in the present work, as they fell outside the scope of this thesis. Furthermore, it may be worthwhile to consider restructuring the SMI model penalty function—currently defined by a combination of ℓ_0 and ℓ_2 regularisation—in a manner analogous to the elastic net, utilising a single regularisation parameter λ and a mixing parameter α. Such a reformulation could yield improvements towards computational efficiency in the estimation process.

Although the discussions in this thesis are based on the assumption that y_t, the response variable at time t, is a scalar, the need for models capable of handling multivariate responses is evident in many real-world applications. For instance, in electricity demand forecasting, it is often necessary to predict demand simultaneously across different population groups or geographical regions. There are several studies on multivariate *single index* models. For example, Feng, Xiao & Chi (2021) proposed a sparse multivariate single index model that incorporates multiple matrix-level penalties to enable effective variable selection, while Tian & Qiu (2023) developed a multivariate single index framework tailored to analyse longitudinal data with multiple response variables. In light of these developments, extending the SMI Modelling algorithm to accommodate multivariate responses represents a promising direction for future research, with the potential to significantly broaden the model's applicability.

On the other hand, forecasting vectorial targets such as regional electricity demand naturally involves a hierarchical structure, where forecasts at lower levels (e.g. individual regions) must align with aggregated totals (e.g. national demand). Moreover, incorporating probabilistic forecasts in this context—known as probabilistic hierarchical forecasting—is an active area of research, with several applications in energy systems and related fields (Ben Taieb, Taylor & Hyndman 2021; Zheng et al. 2025). Therefore, another important future research direction would be to extend the SMI Modelling framework to support hierarchical structures and probabilistic forecast reconciliation.

This thesis aimed to develop a statistical model capable of effectively capturing complex temporal patterns and non-linear predictor-response relationships in time series forecasting. Unlike deep learning models, the proposed SMI model emphasises transparency in its structure and estimation. In recent years, there has been a natural tendency to explore the potential application of machine learning models in real-world forecasting problems. For example, the use of neural network models in groundwater level forecasting was studied by Clark et al. (2020) as a modern approach to time series regression. The authors compared the performance of classical Autoregressive Integrated Moving Average (ARIMA) models, several Feed-forward Neural Network (FFNN) models, and Long Short-Term Memory Networks (LSTM) in forecasting groundwater levels. Therefore, a compelling direction for future research would be to compare the forecasting performance of the SMI model against "black-box" deep learning methods, potentially uncovering valuable insights.

In addition, it would be worthwhile to explore hybrid modelling approaches that embed SMI-like structures within deep learning architectures. For instance, integrating single-index or additive-index structures into neural networks may offer a balance between interpretability and modelling flexibility, potentially reducing computational costs while enhancing predictive performance.

In Chapter 3 of the thesis, we examined several methods for generating prediction intervals to quantify uncertainty in point forecasts obtained from nonparametric additive models. We introduced the CB method as a natural extension of two existing approaches, which did not consistently achieve the expected performance across all settings. Hence, an important direction for future work is to investigate the conditions under which the CB method provides reliable empirical coverage, as well as the scenarios where its performance declines.

It would also be worthwhile to extend the CB method by incorporating adjustments such as weighting the calculated quantiles, as in WSCP (Barber et al. 2023), or using adaptive significance levels, as in ACP (Gibbs & Candès 2021; Zaffran et al. 2022; Gibbs & Candès 2024). These methods, which have been shown to handle non-exchangeable data effectively, may enhance the performance of CB in time series settings.

Furthermore, in Chapter 3, the investigation of CP methods for uncertainty estimation in nonparametric additive models was limited to SCP, WSCP, and ACP. Future research could benefit from evaluating the performance of more recent conformal prediction approaches tailored to time series data—particularly Autocorrelated Multi-step Conformal Prediction (AcMCP), introduced by Wang & Hyndman (2024)—in this context.

In particular, extending the CB method to incorporate these recent developments related to conformal prediction in time series settings would be a valuable direction for future research, especially in terms of improving the reliability of CB prediction intervals.

Moreover, it would be worthwhile to explore the development of an iterative method for online adaptation of the CB approach, as well as existing CP methods for uncertainty quantification in nonparametric additive models, as re-estimating models in a rolling window framework is computationally intensive.

In Chapter 3, we focused solely on constructing prediction intervals for the conditional mean (i.e. point forecasts). We did not consider a fully distributional approach, as it was beyond the scope of the present work. Thus, an important avenue for future research would be to explore the use of methods such as Generalised Additive Models for Location, Scale, and Shape (GAMLSS) (Rigby & Stasinopoulos 2005; Stasinopoulos et al. 2024) to adopt a direct distributional approach. Furthermore, it would be of interest to extend the SMI Modelling algorithm to a nonparametric quantile regression framework, building on methodologies developed in quantile GAMs—for example, as implemented in the `qgam` R package (Fasiolo et al. 2021).

Bibliography

Allaire, JJ, C Teague, C Scheidegger, Y Xie & C Dervieux (2024). *Quarto*. Version 1.5.57. https://github.com/quarto-dev/quarto-cli.

Angelopoulos, A, E Candes & RJ Tibshirani (2024). Conformal PID Control for Time Series Prediction. *Advances in Neural Information Processing Systems* **36**.

Bakker, M & F Schaars (2019). Solving Groundwater Flow Problems With Time Series Analysis: You may not Even Need Another Model. *Groundwater* **57**(6), 826–833.

Barber, RF, EJ Candès, A Ramdas & RJ Tibshirani (2023). Conformal Prediction Beyond Exchangeability. *The Annals of Statistics* **51**(2), 816–845.

Bellman, R (1957). *Dynamic Programming*. Princeton, New Jersey: Princeton University Press.

Ben Taieb, S, JW Taylor & RJ Hyndman (2021). Hierarchical Probabilistic Forecasting of Electricity Demand With Smart Meter Data. *Journal of the American Statistical Association* **116**(533), 27–43.

Bertsimas, D, A King & R Mazumder (2016). Best Subset Selection via a Modern Optimization Lens. *The Annals of Statistics* **44**(2), 813–852.

Boente, G & AM Martínez (2023). A Robust Spline Approach in Partially Linear Additive Models. *Computational Statistics and Data Analysis* **178**, 107611.

Boyd, SP & L Vandenberghe (2004). *Convex Optimization*. Cambridge University Press.

Breiman, L (1995). Better Subset Regression Using the Nonnegative Garrote. *Technometrics* **37**(4), 373–384.

Chatfield, C (1993). Calculating Interval Forecasts. *Journal of Business and Economic Statistics* **11**(2), 121–135.

Clark, S, RJ Hyndman, D Pagendam & LM Ryan (2020). Modern Strategies for Time Series Regression. *International Statistical Review* **88**(S1), S179–S204.

Demirhan, H (2020). dLagM: An R Package for Distributed Lag Models and ARDL Bounds Testing. *PLoS ONE* **15**(2), e0228812.

Dewolf, N, BD Baets & W Waegeman (2023). Valid Prediction Intervals for Regression Problems. *Artificial Intelligence Review* **56**(1), 577–613.

Du, P, G Cheng & H Liang (2012). Semiparametric Regression Models With Additive Nonparametric Components and High Dimensional Parametric Components. *Computational Statistics and Data Analysis* **56**, 2006–2017.

Efron, B, T Hastie, I Johnstone & R Tibshirani (2004). Least Angle Regression. *The Annals of Statistics* **32**(2), 407–499.

Elsayed, S, D Thyssens, A Rashed, HS Jomaa & L Schmidt-Thieme (2021). Do We Really Need Deep Learning Models for Time Series Forecasting? *arXiv preprint arXiv:2101.02118*.

Fan, J & R Li (2001). Variable Selection via Nonconcave Penalized Likelihood and its Oracle Properties. *Journal of the American Statistical Association* **96**(456), 1348–1360.

Fan, S & RJ Hyndman (2012). Short-Term Load Forecasting Based on a Semi-parametric Additive Model. *IEEE Transactions on Power Systems* **27**(1), 134–141.

Fasiolo, M, SN Wood, M Zaffran, R Nedellec & Y Goude (2021). qgam: Bayesian Nonparametric Quantile Regression Modeling in R. *Journal of Statistical Software* **100**(9), 1–31.

Feng, Y, L Xiao & EC Chi (2021). Sparse Single Index Models for Multivariate Responses. *Journal of Computational and Graphical Statistics* **30**(1), 115–124.

Frank, IE & JH Friedman (1993). A Statistical View of Some Chemometrics Regression Tools. *Technometrics* **35**(2), 109–135.

Friedman, JH & JW Tukey (1974). A Projection Pursuit Algorithm for Exploratory Data Analysis. *IEEE Transactions on Computers* **C-23**(9), 881–890.

Friedman, JH & W Stuetzle (1981). Projection Pursuit Regression. *Journal of the American Statistical Association* **76**(376), 817–823.

Gasparrini, A (2011). Distributed Lag Linear and Non-linear Models in R: The Package dlnm. *Journal of Statistical Software* **43**(8), 1–20.

Gibbs, I & E Candès (2021). Adaptive Conformal Inference Under Distribution Shift. *Advances in Neural Information Processing Systems*, 1660–1672.

Gibbs, I & EJ Candès (2024). Conformal Inference for Online Prediction With Arbitrary Distribution Shifts. *Journal of Machine Learning Research* **25**(162), 1–36.

Godahewa, R, C Bergmeir, GI Webb, RJ Hyndman & P Montero-Manso (2021). Monash Time Series Forecasting Archive. In: *35th Conference on Neural Information Processing Systems (NeurIPS 2021) Track on Datasets and Benchmarks*.

Goude, Y, R Nedellec & N Kong (2014). Local Short and Middle Term Electricity Load Forecasting With Semi-parametric Additive Models. *IEEE Transactions on Smart Grid* **5**(1), 440–446.

Guo, J, M Tang, M Tian & K Zhu (2013). Variable Selection in High-dimensional Partially Linear Additive Models for Composite Quantile Regression. *Computational Statistics and Data Analysis* **65**, 56–67.

Gurobi Optimizer Reference Manual (2023). Gurobi Optimization, LLC. https://www.gurobi.com.

Härdle, W, P Hall & H Ichimura (1993). Optimal Smoothing in Single-Index Models. *The Annals of Statistics* **21**(1), 157–178.

Hastie, T (2024). *gam: Generalized Additive Models*. R package version 1.22-5. https://CRAN.R-project.org/package=gam.

Hazimeh, H & R Mazumder (2020). Fast Best Subset Selection: Coordinate Descent and Local Combinatorial Optimization Algorithms. *Operations Research* **68**(5), 1517–1537.

Hazimeh, H, R Mazumder & P Radchenko (2023). Grouped Variable Selection With Discrete Optimization: Computational and Statistical Perspectives. *The Annals of Statistics* **51**(1), 1–32.

Ho, CC, LJ Chen & JS Hwang (2020). Estimating Ground-Level PM2.5 Levels in Taiwan Using Data From Air Quality Monitoring Stations and High Coverage of Microsensors. *Environmental Pollution* **264**, 114810.

Hornik, K, D Meyer, F Schwendinger & S Theussl (2023). *ROI: R Optimization Infrastructure*. R package version 1.0-1. https://CRAN.R-project.org/package=ROI.

Huang, J, JL Horowitz & F Wei (2010). Variable Selection in Nonparametric Additive Models. *The Annals of Statistics* **38**(4), 2282–2313.

Huber, PJ (1985). Projection Pursuit. *The Annals of Statistics* **13**(2), 435–475.

Hyndman, RJ & S Fan (2010). Density Forecasting for Long-Term Peak Electricity Demand. *IEEE Transactions on Power Systems* **25**(2), 1142–1153.

Hyndman, RJ & G Athanasopoulos (2021). *Forecasting: Principles and Practice*. 3rd edition. Melbourne, Australia: OTexts. https://OTexts.com/fpp3.

Ibrahim, S, P Radchenko, E Ben-David & R Mazumder (2023). Predicting Census Survey Response Rates With Parsimonious Additive Models and Structured Interactions. *arXiv preprint arXiv:2108.11328*.

Jakobs, M & T Liebig (2024). AALF: Almost Always Linear Forecasting. *arXiv preprint arXiv:2409.10142*.

Kang, Y, RJ Hyndman & K Smith-Miles (2017). Visualising Forecasting Algorithm Performance Using Time Series Instance Spaces. *International Journal of Forecasting* **33**(2), 345–358.

Konzen, E & FA Ziegelmann (2016). LASSO-Type Penalties for Covariate Selection and Forecasting in Time Series. *Journal of Forecasting* **35**(7), 592–612.

Kruskal, JB (1969). “Toward a Practical Method Which Helps Uncover the Structure of a Set of Multivariate Observations by Finding the Linear Transformation Which Optimizes a New “Index of Condensation””. In: *Statistical Computation*. Ed. by Roy C. Milton and John A. Nelder. Academic Press, pp.427–440.

Land, AH & AG Doig (1960). An Automatic Method of Solving Discrete Programming Problems. *Econometrica* **28**(3), 497–520.

Lei, J, M G’Sell, A Rinaldo, RJ Tibshirani & L Wasserman (2018). Distribution-Free Predictive Inference for Regression. *Journal of the American Statistical Association* **113**(523), 1094–1111.

Lei, J & L Wasserman (2014). Distribution-Free Prediction Bands for Non-parametric Regression. *Journal of the Royal Statistical Society Series B: Statistical Methodology* **76**(1), 71–96.

Lian, H (2012). Variable Selection in High-dimensional Partly Linear Additive Models. *Journal of Nonparametric Statistics* **24**(4), 825–839.

Lim, B & S Zohren (2021). Time-series Forecasting With Deep Learning: A Survey. *Philosophical Transactions of the Royal Society A* **379**(2194), 20200209.

Little, JDC, KG Murty, DW Sweeney & C Karel (1963). An Algorithm for the Traveling Salesman Problem. *Operations Research* **11**(6), 972–989.

Liu, X, L Wang & H Liang (2011). Estimation and Variable Selection for Semiparametric Additive Partial Linear Models. *Statistica Sinica* **21**(3), 1225–1248.

Masselot, P, F Chebana, C Campagna, É Lavigne, TBMJ Ouarda & P Gosselin (2023). Constrained groupwise additive index models. *Biostatistics* **24**(4), 1066–1084.

Mazumder, R, P Radchenko & A Dedieu (2023). Subset Selection With Shrinkage: Sparse Linear Modeling When the SNR is Low. *Operations Research* **71**(1), 129–147.

Mork, D, S Im & A Wilson (2024). *dlmtree: Bayesian Treed Distributed Lag Models*. R package version 1.0.0. https://CRAN.R-project.org/package=dlmtree.

Müller, K & H Wickham (2023). *tibble: Simple Data Frames*. R package version 3.2.1. https://CRAN.R-project.org/package=tibble.

Nelson, M, R Sen & C Subramaniam (2006). Understanding Open Source Software: A Research Classification Framework. *Communications of the Association for Information Systems* **17**.

Nocedal, J & SJ Wright (2006). *Numerical Optimization*. 2nd edition. New York, NY: Springer.

O’Hara-Wild, M, RJ Hyndman & E Wang (2024). *feasts: Feature Extraction and Statistics for Time Series*. R package version 0.4.1. https://CRAN.R-project.org/package=feasts.

O’Hara-Wild, M, RJ Hyndman, E Wang & R Godahewa (2022). *tsibbledata: Diverse Datasets for ’tsibble’*. R package version 0.4.1. https://CRAN.R-project.org/package=tsibbledata.

Papadopoulos, H (2008). “Inductive Conformal Prediction: Theory and Application to Neural Networks”. In: *Tools in artificial intelligence*. InTech.

Papadopoulos, H, K Proedrou, V Vovk & A Gammerman (2002). Inductive Confidence Machines for Regression. In: *Machine learning: ECML 2002: 13th European conference on machine learning Helsinki, Finland, August 19–23, 2002 proceedings 13*. Springer, pp.345–356.

Park, H & F Sakaori (2013). Lag Weighted Lasso for Time Series Model. *Computational Statistics* **28**(2), 493–504.

Peng, RD (2022). *Advanced Statistical Computing*. Accessed: 2023-5-19. https://bookdown.org/rdpeng/advstatcomp/.

Peterson, TJ & AW Western (2014). Nonlinear Time-series Modeling of Unconfined Groundwater Head. *Water Resources Research* **50**(10), 8330–8355.

Politis, DN (2001). Resampling Time Series With Seasonal Components. In: *Frontiers in data mining and bioinformatics: Proceedings of the 33rd symposium on the interface of computing science and statistics*, pp.13–17.

– (2003). The Impact of Bootstrap Methods on Time Series Analysis. *Statistical Science* **18**(2), 219–230.

R Core Team (2024). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/.

Radchenko, P (2015). High Dimensional Single Index Models. *Journal of Multivariate Analysis* **139**, 266–282.

Rajaee, T, H Ebrahimi & V Nourani (2019). A Review of the Artificial Intelligence Methods in Groundwater Level Modeling. *Journal of Hydrology* **572**, 336–351.

Ravindra, K, P Rattan, S Mor & AN Aggarwal (2019). Generalized Additive Models: Building Evidence of Air Pollution, Climate Change and Human Health. *Environment International* **132**, 104987.

Rigby, RA & DM Stasinopoulos (2005). Generalized Additive Models for Location, Scale and Shape. *Journal of the Royal Statistical Society Series C: Applied Statistics* **54**(3), 507–554.

Rudin, C (2019). Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead. *Nature Machine Intelligence* **1**(5), 206–215.

Ruppert, D, MP Wand & RJ Carroll (2003). *Semiparametric Regression*. Cambridge University Press.

Schwendinger, F (2023). *ROI.plugin.gurobi: ’Gurobi’ Plug-in for the R Optimization Infrastructure*. R package version 0.4-0. http://r-forge.r-project.org/projects/roi.

Shafer, G & V Vovk (2008). A Tutorial on Conformal Prediction. *Journal of Machine Learning Research* **9**(3).

Shmueli, G & J Polak (2024). *Practical Time Series Forecasting With R: A Hands-on Guide*. 2nd edition. Axelrod Schnall Publishers.

Simon, N, J Friedman, T Hastie & R Tibshirani (2013). A Sparse-Group Lasso. *Journal of Computational and Graphical Statistics* **22**(2), 231–245.

Smith-Miles, K, D Baatar, B Wreford & R Lewis (2014). Towards Objective Measures of Algorithm Performance Across Instance Space. *Computers and Operations Research* **45**, 12–24.

Stasinopoulos, MD, T Kneib, N Klein, A Mayr & GZ Heller (2024). *Generalized Additive Models for Location, Scale and Shape: A Distributional Regression Approach, with Applications*. Cambridge University Press.

Stoker, TM (1986). Consistent Estimation of Scaled Coefficients. *Econometrica* **54**(6), 1461–1481.

Stone, CJ (1982). Optimal Global Rates of Convergence for Nonparametric Regression. *The Annals of Statistics* **10**(4), 1040–1053.

The UMass Trace Repository (2023). https://traces.cs.umass.edu/ Accessed: 2024-3-19. University of Massachusetts.

Theußl, S, F Schwendinger & K Hornik (2020). ROI: An Extensible R Optimization Infrastructure. *Journal of Statistical Software* **94**, 1–64.

Thornton, PE, R Shrestha, M Thornton, SC Kao, Y Wei & BE Wilson (2021). Gridded Daily Weather Data for North America With Comprehensive Uncertainty Quantification. *Scientific Data* **8**(1), 190.

Tian, Q, DJ Nordman & WQ Meeker (2022). Methods to Compute Prediction Intervals: A Review and New Results. *Statistical Science* **37**(4).

Tian, Z & P Qiu (2023). Multivariate Single Index Modeling of Longitudinal Data With Multiple Responses. *Statistics in Medicine* **42**(17), 2982–2998.

Tibshirani, R (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology* **58**(1), 267–288.

Tibshirani, R & X Suo (2016). An Ordered Lasso and Sparse Time-Lagged Regression. *Technometrics* **58**(4), 415–423.

Tibshirani, RJ, R Foygel Barber, EJ Candès & A Ramdas (2019). Conformal Prediction Under Covariate Shift. *Advances in Neural Information Processing Systems* **32**.

Tolstov, GP (1976). *Fourier Series*. Trans. by RA Silverman. New York, NY: Dover Publications, Inc.

Vovk, V, A Gammerman & G Shafer (2005). *Algorithmic Learning in a Random World*. New York, NY: Springer.

Wang, E, D Cook & RJ Hyndman (2020). A New Tidy Data Structure to Support Exploration and Modeling of Temporal Data. *Journal of Computational and Graphical Statistics* **29**(3), 466–478.

Wang, H, L Guodong & CL Tsai (2007). Regression Coefficient and Autoregressive Order Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology* **69**(1), 63–78.

Wang, L, L Xue, A Qu & H Liang (2014). Estimation and Model Selection in Generalized Additive Partial Linear Models for Correlated Data With Diverging Number of Covariates. *The Annals of Statistics* **42**(2), 592–624.

Wang, T, P Xu & L Zhu (2015). Variable Selection and Estimation for Semi-parametric Multiple-Index Models. *Bernoulli* **21**(1), 242–275.

Wang, T, J Zhang, H Liang & L Zhu (2015). Estimation of a Groupwise Additive Multiple-Index Model and its Applications. *Statistica Sinica* **25**, 551–566.

Wang, X & RJ Hyndman (2024). Online Conformal Inference for Multi-Step Time Series Forecasting. *arXiv preprint arXiv:2410.13115*.

– (2025). *conformalForecast: Conformal Prediction Methods for Multistep-Ahead Time Series Forecasting*. R package version 0.0.0.9000. https://github.com/xqnwang/conformalForecast.

Wood, SN (2011). Fast Stable Restricted Maximum Likelihood and Marginal Likelihood Estimation of Semiparametric Generalized Linear Models. *Journal of the Royal Statistical Society Series B: Statistical Methodology* **73**(1), 3–36.

– (2017). *Generalized Additive Models: An Introduction With R*. 2nd edition. Chapman & Hall/CRC.

– (2023). *mgcv: Mixed GAM Computation Vehicle With Automatic Smoothness Estimation*. R package version 1.9-1. https://CRAN.R-project.org/package=mgcv.

Yuan, M & Y Lin (2006). Model Selection and Estimation in Regression With Grouped Variables. *Journal of the Royal Statistical Society Series B: Statistical Methodology* **68**(1), 49–67.

Zaffran, M, O Féron, Y Goude, J Josse & A Dieuleveut (2022). Adaptive Conformal Predictions for Time Series. In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. PMLR, pp.25834–25866.

Zhang, CH (2010). Nearly Unbiased Variable Selection Under Minimax Concave Penalty. *The Annals of Statistics* **38**(2), 894–942.

Zhang, X, L Liang, X Tang & HY Shum (2008). L1 Regularized Projection Pursuit for Additive Model Learning. In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp.1–8.

Zheng, K, H Xu, Z Long, Y Wang & Q Chen (2025). Coherent Hierarchical Probabilistic Forecasting of Electric Vehicle Charging Demand. *IEEE Transactions on Industry Applications* **61**(1), 1329–1340.

Zou, H (2006). The Adaptive Lasso and its Oracle Properties. *Journal of the American Statistical Association* **101**(476), 1418–1429.

Zou, H & T Hastie (2005). Regularization and Variable Selection via the Elastic Net. *Journal of the Royal Statistical Society Series B: Statistical Methodology* **67**(2), 301–320.