Conformal Prediction Tools in R

by Jesus Uriel Diaz-Martinez, Xiaoqian Wang, and Paula Moraga

Abstract Conformal prediction constitutes a flexible family of methods for predictive uncertainty quantification, which has recently drawn significant attention in the statistical literature because of its distribution-free, finite-sample, and model-agnostic coverage guarantees. While numerous methodological advances have extended the original conformal framework to broader settings, the availability of related software in R remains limited. In this paper, we present an overview of several conformal prediction methods and exemplify the use of available R packages, with a focus on regression for continuous responses given covariates and its extensions to time series forecasting. We further identify current limitations and outline opportunities for future improvements.

1 Introduction

Conformal prediction is a family of methods to construct prediction intervals for point predictions. These methods have been popularized recently in the statistical literature due to their model-agnostic and finite-sample coverage guarantees, which require mild assumptions on the distribution of the data. Such properties make the framework appealing for a wide range of tasks such as regression and time series forecasting. The model-agnostic property allows conformal prediction to be wrapped around several models. On the other hand, the finite-sample property is related to conservatively marginal coverage guarantees. Specifically, for a future observation of interest Y_{n+1}, the resulting prediction interval, denoted by Γ^{α}, will satisfy $P(Y_{n+1} \in \Gamma^{\alpha}) \geq 1 - \alpha$ for any significance level $\alpha \in (0, 1)$. These notable features are particularly valuable for providing predictive uncertainty quantification in models that inherently lack it, in particular black-box prediction models. Standard references on conformal prediction include the book *Algorithmic Learning in a Random World* by Vovk et al. (2022) with a recent edition (Vovk et al., 2022), and a tutorial by Shafer and Vovk (2008).

Conformal prediction has emerged as a reliable framework for uncertainty quantification, with applications spanning a wide range of domains. In the medical sciences, it has been applied in breast cancer diagnosis, survivability prediction, and symptom-based condition identification (Vazquez and Facelli, 2022). In the energy sector, conformal prediction has been used to measure prediction uncertainty in simulation-based forecasting, energy and gas demand prediction, and electricity price prediction (e.g., Mendil et al., 2022). More broadly, conformal prediction has also been used to construct prediction intervals for the Direction-of-Arrival (DOA) predictions in geophysics and underwater acoustics, as well as enhancing reliability in aircraft landing systems (Vilfroy et al., 2024).

On the methodological advances, significant progress has been made to extend the scope and applicability of conformal prediction. Lei et al. (2018) demonstrated that conformal prediction methods are not only conservatively valid but also admit model-independent upper bounds on coverage under additional assumptions on the residuals. Many attempts have been made to broaden the framework to more general settings. For example, Tibshirani et al. (2019) proposed a method to address covariate shift, while Romano et al. (2019) tackled the problem of variance heterogeneity in regression settings using flexible quantile regression. Barber et al. (2023) proposed a general theory based on weighted quantiles, extending conformal prediction to settings previously far from the scope of the original formulation. Chen et al. (2018) investigated computationally efficient methods to implement conformal prediction in regression tasks. In the context of time series forecasting, specialized methods based on online updating of prediction intervals have been pioneered by Gibbs and Candes (2021), Zaffran et al. (2022), Bhatnagar et al. (2023) and recently by Wang and Hyndman (2026).

In the R ecosystem, several CRAN packages provide implementations of conformal prediction for both regression and time series forecasting. Available packages for regression include **pintervals** (Randahl, 2026), **marginaleffects** (Arel-Bundock et al., 2024), and **probably** (Kuhn et al., 2025). For time series forecasting, the **caretForecast** (Akay, 2026) and **conformalForecast** (Wang and Hyndman, 2025) are two packages for conformal prediction of time series forecasting. These packages generally adopt a flexible framework that can be applied to diverse underlying models.

Other available packages offer model-specific implementations, among them are **conformalInference.multi** (Diquigiovanni et al., 2025), designed for multivariate regression; and **ConformalSmallest** (Yang, 2021), which optimizes conformal prediction intervals generated with quantile and ridge regression. While these packages provide valuable contributions, they also present some limitations. For instance, some implementations restrict conformal prediction to models from specific packages due to dependencies on custom objects generated by those ecosystems.

In other languages, like Python, libraries such as **MAPIE** (Cordier et al., 2023), **crepes** (Boström, 2024) and **torchCP** (Huang et al., 2025) provide implementations of conformal prediction algorithms that augment regression and classification algorithms available in **scikit-learn** (Pedregosa et al., 2011) and **pytorch** (Paszke et al., 2019), which are two major machine learning libraries.

In this paper, we demonstrate the use of the **pintervals**, **marginaleffects**, **probably** and **conformalForecast** packages in R to the problem of constructing uncertainty intervals in two settings: regression with covariates and time series forecasting. In the regression context, the methods are applied to predict a continuous response variable using a number of covariates, exemplified through a wind farm energy production dataset. For the time series domain, where multi-step-ahead forecasting is of particular interest, we demonstrate the approach using data on the number of active cases of Dengue in Brazil.

The remainder of the paper is as follows. In Section 2 we review the basics of conformal prediction, from the classical perspective, and its adaptations to regression and time series forecasting. In Sections 3 and 4, we provide code examples for regression and time series forecasting, respectively. Section 5 provides a discussion from the perspective of our examples, while Section 6 states our conclusions.

2 Background on conformal prediction

In this section, we provide an overview of the technical aspects of conformal prediction required to understand the examples and the code. For an in-depth overview, the readers are encouraged to consult the work by Vovk et al. (2022), Shafer and Vovk (2008), Lei et al. (2018), Romano et al. (2019), and Wang and Hyndman (2026), just to mention a few.

2.1 The setting for conformal prediction

The basic setting for conformal prediction assumes that there exists an *example space* $\mathbf{Z}$, from which data points $Z_1, Z_2, Z_3, \ldots$ can be drawn according to a data-generating process. In practice, only a finite sample $\{Z_1, Z_2, \ldots, Z_n\}$ is available; this collection will be denoted by $\mathbf{Z}^{(*)}$. The goal of conformal prediction methods is to construct prediction sets, Γ^α, for a future observation $Z_{n+1} \in \mathbf{Z}$ such that the coverage guarantee $P(Z_{n+1} \in \Gamma^\alpha) \geq 1 - \alpha$ holds for a predefined significance level $\alpha \in (0, 1)$. This guarantee is known as *conservative validity*. A key distributional assumption to achieve this goal is that the data generating process is exchangeable, i.e., the process produces an exchangeable collection of random variables. Specifically, a collection of (possibly vector-valued) random variables, $(Z_1, Z_2, \ldots, Z_n)$, is said to be exchangeable if for any permutation π of the set $\{1, \ldots, n\}$, the joint distribution of $(Z_{\pi(1)}, Z_{\pi(2)}, \ldots, Z_{\pi(n)})$ is the same as the joint distribution of the original collection. This assumption is weaker, and hence more flexible, than the usual i.i.d. assumption since exchangeable random variables are not necessarily independent.

The practical construction of the prediction set relies on the so-called *nonconformity measure*, which is a function $S : \mathbf{Z}^{(*)} \times \mathbf{Z} \to [-\infty, +\infty]$, to quantify how different a new example $z \in \mathbf{Z}$ is relative to the available examples in $\mathbf{Z}^{(*)}$. From this function, one obtains nonconformity scores; the corresponding score for each observation Z_i is defined as $V_i := S(\mathbf{Z}^{(*)}, Z_i)$. In several tasks, such as regression and time series forecasting, there are *natural* choices of the nonconformity measure, for example the absolute residuals for regression and the signed residuals for time series forecasting. This does not imply that they must be the unique choice, as the choice usually depends on the context of the problem (Vovk et al., 2022). A detailed discussion and review of nonconformity measures in the context of regression is provided in Kato et al. (2023).

2.2 Full conformal prediction

One of the earliest and more general approaches is full conformal prediction, which serves as a conceptual starting point for other methods in this framework for uncertainty quantification. In this method, the collection of available examples is augmented with a potential value, $z \in \mathbf{Z}$, for Z_{n+1}. Let $\mathbf{Z}_z^{(*)} := \{Z_1, Z_2, \ldots, Z_n, z\}$ be the collection of augmented examples; note that for each potential value in $\mathbf{Z}$ the augmented set of available examples will be different. Define $V_j := S\left(\mathbf{Z}_z^{(*)}, Z_j\right)$ for $j = 1, 2, \ldots, n$, $V_{n+1}^z := S(\mathbf{Z}_z^{(*)}, z)$, and let α be a significance level. The p-value for the new potential observation will be defined as

$$p_{n+1}^z = \frac{|\{j = 1 \ldots, n+1 : V_j \geq V_{n+1}^z\}|}{n+1}, \tag{1}$$

and the full conformal prediction set is defined as

$$\Gamma^{\alpha} := \left\{ z \in \mathbf{Z} : p^{z}_{n+1} > \alpha \right\}. \tag{2}$$

Intuitively, full conformal prediction evaluates how plausible a candidate future observation would be if it were included together with the training data. Candidate values that produce nonconformity scores similar to those observed in the data will have larger p-values and will be included in the prediction set, whereas those that appear atypical relative to the observed data will have smaller p-values and will be excluded. Therefore, p-values play a crucial role in conformal prediction, determining whether a candidate value is included in the prediction set Γ^{α}. As noted by Vovk et al. (2022), the choice of nonconformity measure directly affects the resulting prediction set.

A practical difficulty with full conformal prediction is that, to construct the prediction set, one needs to compute the nonconformity scores for all candidate elements in the set $\mathbf{Z}$, as every element can be potentially included. In terms of computations, there are particular algorithms for which the expensive computations can be avoided due to the structure of the prediction algorithm, such as conformalized ridge regression and nearest neighbor regression (Vovk et al., 2022). Nevertheless, as highlighted by Lei et al. (2018), the construction of full conformal prediction sets remains computationally expensive in general. This challenge has motivated the development of alternative methods with lower computational cost, a widely used alternative is described in the following subsection.

2.3 Inductive conformal prediction

A widely used and computationally efficient variant of conformal prediction is inductive conformal prediction, also known as split conformal prediction. It is conceptually similar to the full conformal framework; however, the key difference lies in how p-values are constructed. In split conformal prediction, model fitting and prediction set construction are separated, eliminating the need to recompute scores for each candidate value, thereby substantially reducing the computational cost. Instead of augmenting the available examples $\mathbf{Z}^{(*)}$ with potential values from the example space, the available data is split into two disjoint subsets: a proper training set $\mathbf{Z}^{(*)}_{\text{Train}}$ and a proper calibration set $\mathbf{Z}^{(*)}_{\text{Cal}}$, containing l and m elements, respectively, such that $l + m = n$. The key idea in split conformal is to use only the data in the proper training set to fit the model and then use the examples in the calibration set to construct the prediction intervals. Therefore, the nonconformity scores V_j are only defined for calibration samples, with $V_j := S(\mathbf{Z}^{(*)}_{\text{Train}}, Z_j)$, $Z_j \in \mathbf{Z}^{(*)}_{\text{Cal}}$, and for a candidate $z \in \mathbf{Z}$ representing a potential value of Z_{n+1}, $V^{z}_{n+1} := S(\mathbf{Z}^{(*)}_{\text{Train}}, z)$. Using these nonconformity scores, nonconformity of a candidate value is assessed by comparing its score against the empirical distribution of calibration scores. The p-value for inductive (split) conformal prediction is then defined by

$$p^{z}_{n+1} = \frac{|\{j \in I_{\text{Cal}} : V_j \geq V^{z}_{n+1}\}|}{m+1}, \tag{3}$$

where I_{Cal} is the index set of the calibration set, with $|I_{\text{Cal}}| = m$. The corresponding inductive prediction interval is constructed as

$$\Gamma^{\alpha} := \{ z \in \mathbf{Z} : p^{z}_{n+1} > \alpha \}. \tag{4}$$

For split conformal prediction, an alternative expression to Equation (4) exists in terms of the empirical quantiles induced by the nonconformity scores on the calibration set. The prediction interval is constructed according to Equation (5), which is particularly advantageous for computer implementations and theoretical analysis.

$$\Gamma^{\alpha} = \left\{ z \in \mathbf{Z} : V^{z}_{n+1} \leq \text{Quantile}\left(1-\alpha; \{V_j\}_{j \in I_{\text{Cal}}}\right) \right\}. \tag{5}$$

2.4 Applications to regression and time series forecasting

In general, the regression setting can be described as the task of estimating a function $\hat{f} : X \to Y$, where X corresponds to the space where the predictors take values, typically a subset of $\mathbb{R}^d$ with d representing the number of *predictors*; the space Y, which is usually a subset of $\mathbb{R}$, contains the *response* of interest. For the application of conformal prediction to regression, the example space is defined as $\mathbf{Z} = X \times Y$. It is assumed that a future value of the predictor X_{n+1} is observed in order to produce a prediction for the response Y_{n+1} given by $\hat{Y}_{n+1} = \hat{f}(X_{n+1})$.

In the particular case of full conformal prediction, the data augmentation step will be performed as $\mathbf{Z}_y^{(*)} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n), (x_{n+1}, y)\}$, for some $y \in Y$. The nonconformity scores will be computed as $V_i = S(\mathbf{Z}_y^{(*)}, y_i)$ and $V_{n+1}^y = S(\mathbf{Z}_y^{(*)}, y)$; then the p-value and the prediction interval will follow a similar definition as in Equations (1) and (2), respectively, with minor notational changes to make explicit the dependence only on the potential value y for Y_{n+1}:

$$\Gamma^\alpha := \{y \in Y : p_{n+1}^y \geq \alpha\}. \tag{6}$$

The theoretical properties of this approach have been investigated by Lei et al. (2018), who established an upper bound on the coverage probability in finite sample settings. This property, also known as anti-conservativeness, is independent of the predictive model or any distributional assumptions, depending only on the sample size of the dataset. The result is stated below.

Result 1 *Under the exchangeability assumption the conformal prediction method for regression satisfies that for a new pair of observation* (X_{n+1}, Y_{n+1}) *it holds that* $1 - \alpha \leq P(Y_{n+1} \in \Gamma^\alpha) \leq 1 - \alpha + \frac{1}{n+1}$.

It is important to remark that this result was proven using the absolute deviation in Equation (7) as nonconformity measure, although the result holds with any valid nonconformity measure as long as the data is exchangeable and the algorithm is symmetric.

$$S(Y_i, \hat{Y}_i) := |Y_i - \hat{Y}_i|. \tag{7}$$

Split conformal prediction extends naturally to the regression setting by using the same ideas as full conformal prediction, but with nonconformity scores computed exclusively on the calibration set. This yields the following coverage guarantee.

Result 2 *Under the exchangeability assumption the split conformal prediction method for regression, constructed with a calibration set with m examples, satisfies that for a new pair of observations* (X_{n+1}, Y_{n+1}), $1 - \alpha \leq P(Y_{n+1} \in \Gamma^\alpha) \leq 1 - \alpha + \frac{1}{m+1}$.

Using Result 2, one can solve for the size of the calibration set m to get

$$m \geq \frac{2 - (\nu + \alpha)}{\nu + \alpha - 1}, \tag{8}$$

where ν is a positive scalar that satisfies $1 - \alpha < \nu \leq 1$. The equation above does not imply that using the derived calibration set size in practice we will observe an empirical coverage around ν, as this upper bound is derived based on a worst-case analysis and not a coverage calibration. However, we can use it to choose the size of the calibration set and we will use it in the next section.

When the nonconformity measure defined in Equation (7) is used in the split conformal prediction framework, the resulting prediction intervals take the following simple form:

$$[\hat{Y}_{n+1} - \hat{Q}_\alpha, \hat{Y}_{n+1} + \hat{Q}_\alpha], \tag{9}$$

where $\hat{Q}_\alpha$ is the $\lceil (m+1)(1-\alpha) \rceil$-th empirical quantile of the nonconformity scores computed on the calibration set. The prediction interval from Equation (9) therefore has a constant width across all values of the predictors.

The coverage guarantees of the methods described above rely on the validity of exchangeability in the data and on the prediction algorithm being symmetric. However, in the context of time series, the exchangeability is not a realistic assumption for the data and hence coverage guarantees of usual conformal prediction methods do not apply. One of the main features of time series is the existence of temporal dependence, with observations ordered in time and often displaying autocorrelation. In addition, several methods for time series modeling and prediction are not symmetric, again, because of the temporal structure inherent to the data. In practice, applications of classic conformal prediction methods to time series forecasting can lead to systematic under coverage, meaning the obtained prediction interval does not achieve the nominal coverage level. In time series forecasting, two scenarios are of particular interest: one-step ahead and multi-step ahead forecasting. In the following, time series will be denoted by $\{Y_t\}$, where each $Y_t \in Y$. An h-step ahead forecast is denoted by $\hat{Y}_{t+h}$, where $H \geq 1$ denotes the forecasting horizon. If covariates are used to produce the forecast, they will be denoted by $\mathbf{x}_t \in X$.

Several methods have been developed to extend conformal prediction to time series forecasting. Of particular interest has been to address the violation of data exchangeability and to allow for non-symmetric prediction algorithms. For example, the NexCP (Non Exchangeable Conformal Prediction)

method of Barber et al. (2023) extends full and split conformal prediction by introducing weights to the construction of quantiles, thereby allowing more "trusted" observations to play a greater role in the prediction step. The approach provides approximate coverage when exchangeability does not hold and exact coverage when it does. This method is general and applies not only to time series but other instances where dependence plays a major role in the data and must not be ignored. Other approaches exist such as the EnbPI method proposed by Xu and Xie (2021), which employs ensembles of bootstrap models to construct prediction intervals, one notable advantage of this method is the fact that it is suitable for multi-step forecasting tasks.

Beyond weighted or bootstrap-based extensions, another line of research known as online conformal prediction, is inspired by the sequential nature of time series. Online conformal prediction methods produce intervals dynamically as new data becomes available, refining the significance level over time to adapt to changes in data distribution or updates to the model based on the most recent information. The first such method was developed by Gibbs and Candes (2021), under the assumption that an optimal significance level exists for constructing the desired $1-\alpha$ prediction interval for the point forecast $\hat{Y}_{t+1}$. Hence the authors propose a method to estimate this time-varying significance level α_{t+1}, and use it to construct prediction intervals. Further enhancements of the update rule include Zaffran et al. (2022), Bhatnagar et al. (2023), Angelopoulos et al. (2023), and recently Wang and Hyndman (2026) to construct prediction intervals for multi-step ahead predictions. Wang and Hyndman (2026) also developed the **conformalForecast** R package (Wang and Hyndman, 2025), providing a practical implementation of their approaches.

Online conformal prediction methods take into account the temporal ordering of the data by introducing the idea of sequential splits. At each time point $t = N$ where $N >> H$, the available data are split into a proper training set of size t_r and a proper calibration set of size $N - t_r > H$. For notational convenience, the data at time t is denoted by z_t, to accommodate both the univariate case, where only the time series of interest is used to fit the model (hence $z_t = y_t$), and the multivariate case, where covariates are included so that $z_t = (\mathbf{x}_t, y_t)$. In the latter, the covariates may include not only information up to the current time but also information of the subsequent h steps. Within this framework, the training set is defined as a moving window of size t_r and the nonconformity measure is defined for each h-step-ahead forecast at time $t + h$ as follows

$$S(\{Z_j\}_{j=t-t_r+1}^{t}, Y_{t+h}), \quad \text{for } t_r \leq t \leq N;\ h = 1, \ldots, H,$$

with corresponding score expressed as

$$V_{t+h|t} := S(\{z_j\}_{j=t-t_r+1}^{t}, y_{t+h}), \quad \text{for } t_r \leq t \leq N;\ h = 1, \ldots, H.$$

In the particular case of the work by Wang and Hyndman (2026), the nonconformity measure is given by

$$S(\{Z_j\}_{j=t-t_r+1}^{t}, Y_{t+h}) := Y_{t+h} - \hat{Y}_{t+h|t},$$

where $\hat{Y}_{t+h|t}$ is the prediction of Y_{t+h} using data in the training window corresponding to time t, i.e, $\{Z_j\}_{j=t-t_r+1}^{t}$.

The steps for the multi-step-ahead conformal prediction are outlined as follows:

1. Initialization: Obtain the initial proper training set $\{z_k\}_{k=1+t-t_r}^{t}$ with $t = t_r$, and fit the forecasting model. Then obtain H-step-ahead point forecasts $\{\hat{y}_{t+h|t}\}_{h=1}^{H}$ and compute the corresponding nonconformity scores $\{V_{t+h|t}\}_{h=1}^{H}$.
2. Rolling procedure: Roll the training window of size t_r forward by one observation, i.e., set $t \to t+1$, for $t_r \leq t \leq N$. Then repeat Step 1 until the collection of nonconformity scores, $\{V_{t+h|t}\}_{t=t_r}^{N-H}$, $h = 1, \ldots, H$, have been computed across the entire calibration set.
3. Prediction interval construction: Use the nonconformity scores obtained from Step 2 to compute quantile estimates $\hat{q}_{t+h|t}$ for each forecasting horizon $h = 1, \ldots, H$. Conformal prediction intervals are then given by $\Gamma_{t+h|t}(\hat{q}_{t+h|t}) := \{y \in Y : V_{t+h|t}^{y} \leq \hat{q}_{t+h|t}\}$, with $t = N$ and for each horizon $h = 1, \ldots, H$.
4. Online update: As new observations arrive, update the calibration and the training sets by rolling the data forward one data point at a time. Recompute the nonconformity scores based on the new calibration set and then repeat Step 3 to update prediction intervals across time.

The quantile $\hat{q}_{t+h|t}$ in Step 3 can be estimated using the methods proposed by Wang and Hyndman (2026). In particular, the update rule for the proposed Autocorrelated Multi-step Conformal Prediction (AcMCP) method is defined as

$$\hat{q}_{t+h|t} = \hat{q}_{t+h-1|t-1} + \eta\left(\text{err}_{t|t-h} - \alpha\right) + r_t\left(\sum_{i=h+1}^{t}\left(\text{err}_{i|i-h} - \alpha\right)\right) + \tilde{e}_{t+h|t}, \tag{10}$$

where $\tilde{e}_{t+h|t}$ is a term that accounts for autocorrelation in the nonconformity scores and is the combination of a MA($h-1$) model trained on the h-step-ahead forecast errors and a linear regression model trained to regress the h-step-ahead error from the past $h-1$ errors. $\text{err}_{t|t-h}$ is defined as

$$\text{err}_{t|t-h} := \begin{cases} 1, & \text{if } y_t \notin \Gamma_{t|t-h}(\hat{q}_{t|t-h}) \\ 0, & \text{otherwise} \end{cases}.$$

$\eta > 0$ is a constant learning rate, and r_t is a saturation function that satisfies the conditions

$$x \geq c \cdot g(t-h) \Rightarrow r_t(x) \geq b, \quad \text{and} \quad x \leq -c \cdot g(t-h) \Rightarrow r_t(x) \leq -b,$$

for some constant $b, c > 0$ and a function g that is sublinear, nonnegative and nondecreasing. These conditions are required to establish the asymptotic coverage guarantees.

In the following sections, we show how to use available implementations of conformal prediction methods for regression and time series forecasting. It is important to note that for the regression part, the space $Y = \{y \in \mathbb{R} : y \geq 0\}$, and the covariate space is $X = \{(1, x) : x \geq 0\}$. For time series forecasting, the response Y remains the same.

3 Conformal prediction methods for regression in R

In this section, split conformal prediction will be illustrated using the SCADA dataset obtained from Kaggle (Erisen, 2019). SCADA, short for Supervisory Control and Data Acquisition systems, is widely used to monitor wind energy production. The dataset consists of 39, 692 registries of electricity production (KWh), wind speed (m/s), theoretical electricity production (KWh), and wind direction collected from a wind turbine's SCADA system in Turkey. Our objective is to predict the production of a windmill using wind speed as a covariate. Such datasets present challenges for modeling, as system failures can generate anomalous observations that deviate from expected behavior. Therefore, the models used for prediction should be robust and capable of quantifying uncertainty in the presence of outliers, which may otherwise compromise the reliability of prediction intervals.

3.1 Data preparation

First, we load the dataset and inspect its first rows. Then, we create a data frame `d_scada` for the analysis, containing the response `y`, representing active power (`active_power`), and the covariate `x`, representing wind speed (`wind_speed`). A simple summary of the data indicates that active power (`y`) ranges from 0 to 3618.7 (kW), while wind speed ranges from 1.2 to 25.2 (m/s).

```
library(readr)
library(curl)
url_scada <- "https://raw.githubusercontent.com/uriel278/tutorials-cp/main/data/scada.csv"
ds_scada <- read_csv(curl(url_scada))
head(ds_scada)
```

```
#> # A tibble: 6 x 4
#>   active_power wind_speed theoretical_power wind_direction
#>          <dbl>      <dbl>             <dbl>          <dbl>
#> 1         380.       5.31              416.           260.
#> 2         454.       5.67              520.           269.
#> 3         306.       5.22              391.           273.
#> 4         420.       5.66              516.           271.
#> 5         381.       5.58              492.           266.
#> 6         402.       5.60              499.           265.
```

```
d_scada <- data.frame(y = ds_scada$active_power, x = ds_scada$wind_speed)
```

As split conformal prediction requires a calibration set and we need a proper training set for model fitting, we generate the data splits by using the function `split(d_scada, idx)`, where the second argument `idx` is a vector of the same length as the number of observations and contains one of the

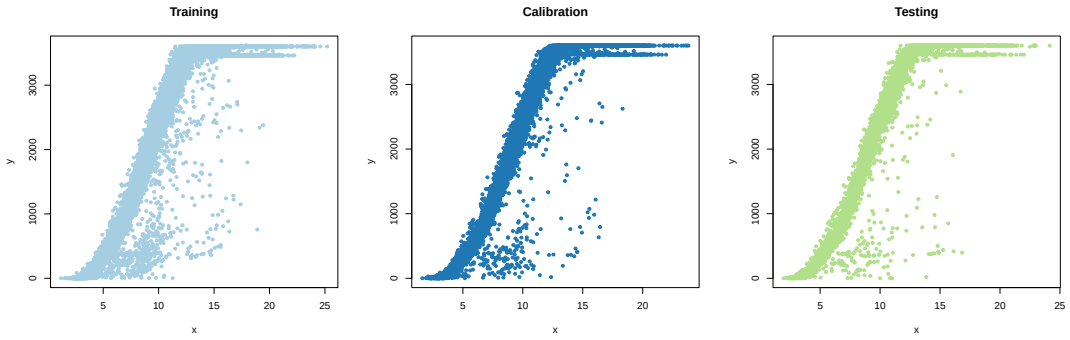

Figure 1: Splits of the original dataset. Left: Proper training data to fit each model. Middle: Proper calibration data for computation of nonconformity scores. Right: Test split to evaluate empirical coverage of each prediction interval.

strings from the vector `c("train", "calib", "test")` selected randomly by the `sample()` function with arguments `prob = c(0.6, 0.252, 0.148)`, `size = nrow(d_scada)`, and `replace = TRUE`; this results in a split that allocates 60% of the data to the proper training set, 25.2% to the proper calibration set, and the remaining 14.8% to the testing set. In this example, the choice of allocating 60% of the data for training is expected to preserve the response-covariate relationship for reliable model training, the size of the calibration set was determined by setting $\nu = 0.95001$ in Equation (8).

```
set.seed(141019)
idx <- sample(x = c("train", "calib", "test"), prob = c(0.6, 0.252, 0.148),
  size = nrow(d_scada), replace = TRUE)
scada <- split(d_scada, idx)
```

Next, we visualize in Figure 1 the obtained data splits.

```
cols <- c(
  palette.colors(3, palette = "Paired"),
  palette.colors(2, palette = "Okabe-Ito")
)
par(mfrow = c(1, 3))
with(scada, {
  plot(y ~ x, data = train,
     col = cols[1], xlab = "x", ylab = "y", main = "Training", pch = 20)
plot(y ~ x, data = calib,
     col = cols[2], xlab = "x", ylab = "y", main = "Calibration", pch = 20)
plot(y ~ x, data = test,
     col = cols[3], xlab = "x", ylab = "y", main = "Testing", pch = 20)
})
```

3.2 Modeling

In this subsection the models to perform predictions for the data set will be introduced. Three models will be fitted to the data to exemplify different workflows for conformal prediction. The first model is a linear model with quadratic term and will be implemented with the well known `lm()` function, the second model will be a monotone generalized additive model fitted using the `scam()` function from the **scam** package, and for users familiar with the tidymodels framework, the third model will be a random forest regression model fitted using functions from the ecosystem.

Model: Linear model

For simplicity, we first model the response variable using a simple linear regression (SLR) model with a quadratic term for the covariate using the training data.

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \epsilon, \quad \epsilon \sim N(0, \sigma^2).$$

In R, the following call fits the model described above.

```
linear_model <- lm(y ~ x + I(x^2), data = scada$train)
```

Model: SCAM

Given the nonlinear shape of the relationship between the response and the covariate of interest, we also fit an alternative model, namely a Shape Constrained Additive Model (SCAM), which offers greater flexibility in capturing the form of the regression function and includes the option to add monotonicity constraints to the fitted regression function. The model is of the form

$$Y = f(X) + \epsilon, \quad \epsilon \sim N(0, \sigma^2),$$

where $f(X) = \sum_{j=1}^{k} b_j(X)\beta_j$ and $b_j(\cdot)$ denotes the basis function, taken here to be cubic splines.

To fit the model above, we use the functions `s()`and `scam()` from the **scam** package (Pya, 2025) to specify the smooth term and fit the model to the training data, respectively. To specify the monotone increasing smooth term, we call the `s(x, bs = "mpi")` in the right hand side of the formula; here the argument `bs = "mpi"` indicates the monotonicity constraint. Additionally, we have specified the argument `family = Gamma(link = "identity")` to select the Gamma distribution as the likelihood for the data with identity link, as energy production is non-negative.

```
library(scam)
scam_model <- scam(y ~ s(x, bs = "mpi"), family = Gamma(link = "identity"),
                   data = scada$train)
```

Model: Random forest regression

As a non-parametric alternative that makes fewer assumptions about the underlying distribution and functional form, we also consider a Random Forest (RF) regression model. Random Forest is an ensemble learning method that captures complex, non-linear interactions by aggregating the predictions of multiple decision trees. This model can be expressed as

$$Y = \frac{1}{B}\sum_{b=1}^{B} T_b(X, \theta_b) + \epsilon$$

Here, the regression function has the form $f(X) = \frac{1}{B}\sum_{b=1}^{B} T_b(X, \theta_b)$ where each $T_b(\cdot)$ denotes an individual decision tree fitted using a bootstrap sample of the training data. To fit this model, we will illustrate the usage of the tidymodels framework for those users familiar with it. In this framework, we will use the **parsnip** package to specify the model, we will set the engine as ranger to specify such package as backend for model fitting.

```
library(tidymodels)
library(parsnip)
rf_spec <- rand_forest(trees = 154, min_n = 813) %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("regression")
rf_wf <- workflow() %>%
  add_model(rf_spec) %>%
  add_formula(y ~ x)
rf_fit <- rf_wf %>% fit(data = scada$train)
```

The parameters `trees = 154` and `min_n = 813` were selected via 5-fold cross-validation on the training split. The argument `min_n` was optimized first by setting the `trees` argument to its default (`trees = 500`). After the optimal value was found, the `trees` parameter was optimized by setting `min_n = 813`, the optimal value from the previous step.

Prediction intervals

For the computation of prediction intervals powered by conformal prediction, we will use the packages **pintervals** and **marginaleffects** for all the fitted models, while the **probably** package will be used for the model fitted with the tidyverse framework. We will focus on split conformal prediction to reduce the computational load. This will help us illustrate the differences between the two packages.

The **pintervals** package follows a package agnostic implementation for the computation of the intervals. The function for split conformal prediction is `pinterval_conformal()`, which requires as arguments a vector of predictions made by a model on the data for which the intervals will be computed, a vector of predictions on the calibration set, the vector of true values of the response on

the calibration set, and the confidence level. These arguments correspond to the argument `pred`, `calib`, `calib_truth`, and `alpha`. The `pinterval_conformal` handles the computation of the nonconformity scores and returns the prediction intervals as a tibble.

```
# This step takes on average 2 minutes on a MacBook Pro machine with 16 GB of
# RAM and a 2.3 GHz 8-Core Intel(R) Core(TM) i9-9889H CPU.
library(pintervals)
lm_pintervals <- pinterval_conformal(
  pred = predict(linear_model, newdata = scada$test),
  calib = predict(linear_model, newdata = scada$calib),
  calib_truth = scada$calib$y
)
scam_pintervals <- pinterval_conformal(
  pred = predict(scam_model, newdata = scada$test),
  calib = predict(scam_model, newdata = scada$calib),
  calib_truth = scada$calib$y
)
rf_pintervals <- pinterval_conformal(
  pred = predict(rf_fit, new_data = scada$test)$.pred,
  calib = predict(rf_fit, new_data = scada$calib)$.pred,
  calib_truth = scada$calib$y
)
```

When using the **marginaleffects** package to compute the prediction intervals, we need to use the `predictions` and `inferences` functions from the package. The arguments to the first function should be a fitted model and covariate values at which predictions will be computed in our case they should be specified by `model = scam_model` and `newdata = scada$test`. The `inferences` function is used to specify the method employed to compute uncertainty estimates around the predictions from the model. The three required arguments by this function are the method, in our case `method = "conformal_split"`, the training data and the calibration data specified by `data_train = scada$train` and `data_calib = scada$calib`.

```
library(marginaleffects)
linear_meint <- predictions(model = linear_model, newdata = scada$test) |>
  inferences(
    method = "conformal_split",
    data_train = scada$train,
    data_calib = scada$calib
  )

scam_meint <- predictions(model = scam_model, newdata = scada$test) |>
  inferences(
    method = "conformal_split",
    data_train = scada$train,
    data_calib = scada$calib
  )
```

Finally, the **probably** package can be used to obtain prediction intervals for models fitted with the tidymodels framework. The nonconformity scores for split conformal prediction are computed using the `int_conformal_split()` function. The arguments of this function include the fitted model workflow and the calibration split. Then, to retrieve the prediction intervals we use the `predict()` function by providing the nonconformity scores obtained by `int_conformal_split()` function, the predictor values for which we want to get prediction intervals, and the desired confidence level.

```
library(probably)
rf_printervals <- rf_fit |>
  int_conformal_split(cal_data = scada$calib) |>
  predict(new_data = scada$test, level = 0.95)
```

Plot predictions and prediction intervals

After the results for each method have been obtained, plots of the estimated regression lines together with their corresponding prediction intervals are shown in Figure 2. The code used to generate the figure is provided below.

```
plot_pred <- function(pred, pred.low, pred.high, newdata, label ="Split", model = "LM") {
  p <- data.frame("x" = newdata$x, "y" = newdata$y, "pred" = pred,
                  "pred.low" = pred.low, "pred.high" = pred.high)

  p <- p[order(p$x), ]
  with(p, {
    coverage <- mean(y >= pred.low & y <= pred.high) * 100
    width <- median(pred.high - pred.low)
    caption <- sprintf("%s + %s\nCoverage = %.2f%%\nMedian width = %.2f",
                       model, label, coverage, width)
    plot(y~x, col = cols[3], pch = 19, cex =.5, main = caption)
    lines(pred~x)
    lines(x, pred.high, lwd = 1.5, col = cols[5])
    lines(x, pred.low, lwd = 1.5, col = cols[5])
  })
}
```

4 Conformal prediction methods for time series forecasting

The R package **conformalForecast** (Wang and Hyndman, 2025) implements the AcMCP method developed in Wang and Hyndman (2026) and some extensions of several other popular conformal prediction methods for multi-step forecasting. An attractive feature of the package design is the attempt to separate the model specification and fitting from the conformal prediction step. This structure reflects the core principle of conformal prediction— its flexibility and ability to wrap around almost any model. In particular, the functions for conformal prediction require the user to provide a function capable of generating predictions (a model). This function is then used to obtain predictions on the (online) calibration set, from which conformal prediction intervals are constructed using any of the methods implemented in the package.

In this example, a time series of the weekly number of dengue cases in São Paulo, Brazil is considered. The data has been obtained from the surveillance system InfoDengue (Codeco et al., 2018) and contains the number of dengue cases spanning 678 weeks from 2010 to 2022 (52 weeks per year, with the exception of 2012 and 2017, which each contains 53 weeks).

4.1 Data preparation

We begin by reading the dengue time series data with `read.csv()`, and then we create a time series object with the `ts()` function using the weekly number of cases. As mentioned above, there are two years in which the number of observations is 53 instead of 52. For visualization purposes, we specify `frequency = 52` in the `ts()` function, which simplifies the visualization despite these irregularities. An alternative approach could involve using `frequency = 52.28571` to account for leap years, or imputing the 53rd week's data by averaging it with the 52nd week, if appropriate for the context of the problem and data at hand. For this illustration, however, we prioritize simplicity and clarity in illustrating how to calculate intervals, so we proceed with `frequency = 52`. Importantly, in the model building step (see section Modeling and prediction), the `frequency` argument is not used, as seasonality is explicitly defined in the model. If a model were to depend on this argument, the user would need to adjust the frequency accordingly to ensure valid results.

A summary of the data shows weekly dengue cases range from 38 to 12, 367 cases.

```
library(readr)
library(curl)
url_dengue <- "https://raw.githubusercontent.com/uriel278/tutorials-cp/main/data/dengue.csv"
d_dengue <- read_csv(curl(url_dengue))
d_dengue <- ts(d_dengue$x, start = c(2010, 1), frequency = 52)
summary(d_dengue)

#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    38.0   126.5   232.0   563.8   459.0 12367.0
```

For this time series, the data from 2010-2013 will play the role of the initial training set, the data from 2014-2017 will be used for calibration, and the data from 2018-2022 will serve as the testing set for evaluation of empirical coverage. The **conformalForecast** package splits the data automatically, as

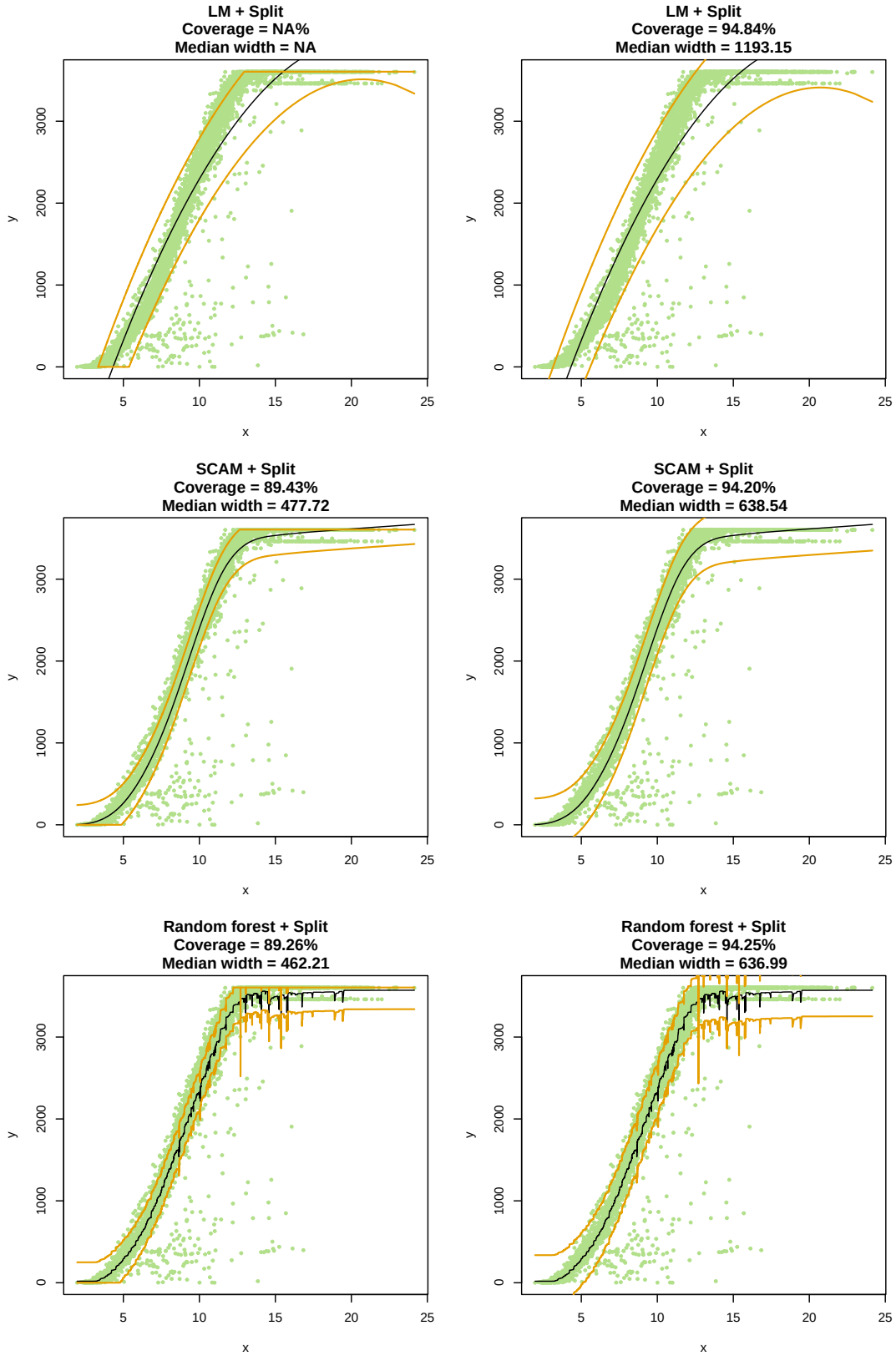

Figure 2: Point predictors depicted with black color and its corresponding 95% prediction intervals in dark orange. Top row: Prediction intervals computed with the pintervals package for the linear model with quadratic term (left), and the marginaleffects package (right). Middle row: Conformal prediction intervals for the SCAM model using pintervals (left), and marginaleffects (right). Bottom row: Prediction intervals for the random forest model produced by the pintervals package (left), and the probably package (right).

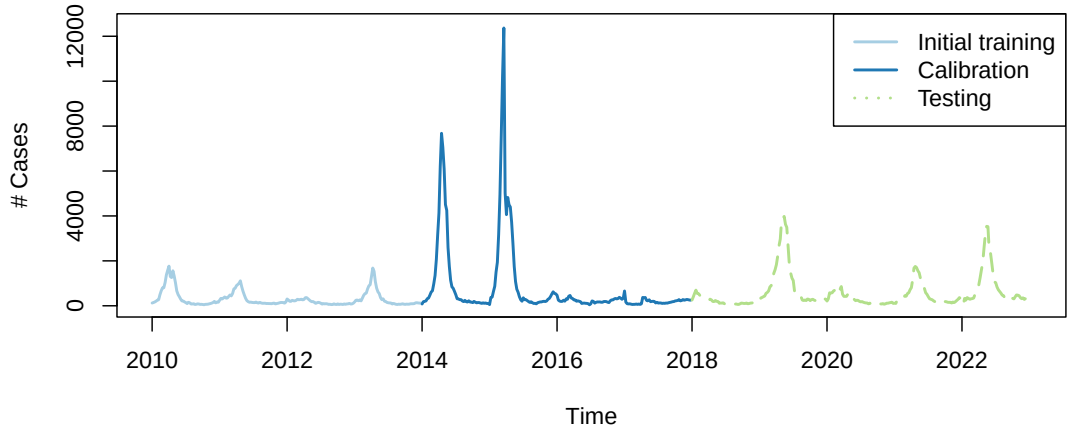

Figure 3: Splits of the dengue time series for the example. The initial training split (in light blue), calibration set (in dark blue) and the testing set in dashed green lines.

described in section Prediction intervals below, ensuring consistency with the setting illustrated in Figure 3.

For visualization purposes, the initial training, calibration, and testing splits can be created manually using the `window()` function by providing the start and end dates of each segment. The resulting initial splits are displayed in the Figure 3, each represented with a different color.

```
dtrain_dengue <- window(d_dengue, start = c(2010, 1), end = c(2013, 52))
dcal_dengue <- window(d_dengue, start = c(2014, 1), end = c(2017, 52))
dtest_dengue <- window(d_dengue, start = c(2018, 1))
```

4.2 Modeling and prediction

The **conformalForecast** package separates the model specification and fitting from the process of computing conformal prediction intervals. Users provide a forecasting function that generates point forecasts for the calibration set. These forecasts are then used to compute nonconformity scores and to construct conformal prediction intervals.

It is important to remark that the main function from the **conformalForecast** package, `cvforecast()`, does not require the user to pass pre-defined data splits. The splits generated above are only for visualization purposes. Instead, the `cvforecast()` function generates splits following the sequential split framework described in Section Multi-step Conformal Prediction.

The `cvforecast()` function

The `cvforecast()` function is the main function of the **conformalForecast** package. As arguments it requires a time series object via the `y` argument, a forecasting function with the `forecastfun` argument, the forecasting horizon `h`, and the confidence level(s) `level` for the prediction interval. The `level` argument accepts either a single numerical value or a vector of numerical values to compute prediction intervals at multiple confidence levels. However, caution is required when providing a vector, as some conformal prediction methods available in the package cannot handle multiple significance values and may return `NA` values for prediction intervals without warning. All methods work correctly when a single numerical input is provided, although this limitation is not documented. Additional arguments are available, and readers are encouraged to see the package documentation for details.

The `cvforecast()` function applies the `forecastfun` to the data, computes predictions for each forecasting horizon given sequential information based on sequential splits, and computes nonconformity scores (in this case, residuals). The returned object is of class `cvforecast` and `forecast`.

This function only computes predictions that are required by the conformal prediction methods available in the package, it does not compute any conformal prediction intervals directly. However, if the forecasting function supplied via `forecastfun` produces its own prediction intervals, these will be stored in the returned object.

An example of a call to the `cvforecast()` function is as follows.

```
pred <- cvforecast(y = d_dengue, forecastfun = forecastfunction,
```

```
                h = 4, level = 95, window = 52*4)
```

Here, the data `d_dengue` contains the dengue time series. The forecasting function is provided through the argument `forecastfun` (details are explained in the following subsection). The horizon `h = 4` requests four-step-ahead forecasts, and `level = 95` specifies 95% prediction intervals from the forecasting function. The argument `window = 52*4` restricts the training window to four years of data.

Model specification and the `forecastfun` argument

Here, we specify the function used to obtain point forecasts. In this illustration example, we use a Seasonal ARIMA (SARIMA) model, as in the plots we can observe some temporal patterns that repeat over time. The SARIMA model for a time series Y_t is of the form:

$$\Phi_P(B)\phi_p(B)\nabla^d\nabla_S^D Y_t = \Theta_Q(B)\theta_q(B)W_t,$$

where B is a lag operator, $\Phi(\cdot)$ and $\phi(\cdot)$ are the seasonal autoregressive and autoregressive polynomials of orders P and p, respectively; $\Theta_Q(\cdot)$ and $\theta_q(\cdot)$ are the seasonal moving average and moving average polynomials of orders Q and q, respectively; ∇_S^D and ∇^d are the seasonal difference and difference operators of orders (D, S) and d, respectively, and W_t is a white noise process.

The code below shows how to define a forecasting function, `forecastfunction()`, using the `Arima()` function of the **forecast** R package. This function will later be passed to `cvforecast()` to compute forecasts as the `forecastfun` argument. To be compatible, any forecasting function defined here must accept three arguments: a time series `x`, the forecasting horizon `h`, and the confidence level `level`. Failure to do so will result in errors when calling the `cvforecast()` function. Note that this requirement does not pose a limitation in the forecasting models that can be used, as the user can always create a wrapping function mapping these arguments to the corresponding inputs of the model of interest. In our example, the wrapper combines the `Arima()` function, which estimates the model given a dataset and specification arguments, with the `forecast` function, which produces multi-step forecasts and confidence intervals. Furthermore, note that before the prediction/forecast step we fit the model. This is not a mandatory step, we may have an already fitted model and can call only the corresponding prediction function. A complete example of the `forecastfunction()` is shown below.

```
library(forecast)
forecastfunction <- function(x, h, level){
  Arima(x, order = c(1, 0, 1),
    seasonal = list(order = c(0, 1, 0), period = 52),
    method = "ML") |>
    forecast(h = h, level = level)
}
```

We can verify that this function is working well by applying it to the training set `dtrain_dengue` to obtain four-step-ahead forecasts with 95% prediction intervals. These intervals are generated by the `forecast()` function of the **forecast** package, applied to the object returned by `Arima()`, and are not conformal prediction intervals.

```
forecastfunction(x = dtrain_dengue, h = 4, level = 95)
```

```
#>          Point Forecast      Lo 95    Hi 95
#> 2014.000       173.5596   11.67358 335.4457
#> 2014.019       249.0351  -48.40245 546.4726
#> 2014.038       259.5708 -110.87368 630.0154
#> 2014.058       226.1600 -192.66605 644.9861
```

Despite being very flexible, the **conformalForecast** requires objects of type `forecast` with a certain structure to be able to work properly. In the following, using the **caretForecast**, we will illustrate how to go over the issues by extracting the required fields from a model, and creating a similar object for the **conformalForecast** package to work with. The second model will be fitted using nonnegative least squares, which restrict the fitted regression function to be nonnegative; in the **caretForecast** package all the models have an autoregressive structure.

To fit the model, we specify the `predfun` as follows

```
library(caretForecast)
```

```
nnlsreg <- function(x, h, level = 0.95){
  fit <- ARml(x, max_lag = 52, caret_method = "nnls",
              verbose = FALSE, seasonal = FALSE, fixed_window = TRUE,
              cv_horizon = h, initial_window = 52)
  fc <- forecast(fit, h = h, level = level)
  caretToforecast(fc)
}
```

The function `caretToforecast()` is specified in the following, where the fields of the forecast object produced by the resulting fit from the **caretForecast** are parsed to the expected format of the **conformalForecast**, in this case the `lower` and `upper` fields must have as column names the coverage specified by the user.

```
caretToforecast <- function(caretForecastObj){
  level <- caretForecastObj$level
  colnames(caretForecastObj$lower) <- paste0(level, "%")
  colnames(caretForecastObj$upper) <- paste0(level, "%")
  standardize_fc <- structure(
    list(
      "method" = caretForecastObj$method,
      "level" = caretForecastObj$level,
      "mean" = caretForecastObj$mean,
      "lower" = caretForecastObj$lower,
      "upper" = caretForecastObj$upper,
      "fitted" = caretForecastObj$fitted
    ),
    class = "forecast"
  )
  return(standardize_fc)
}
```

4.3 Prediction intervals

Now that the `cvforecast()` function and the `forecastfun` argument have been introduced, we proceed to obtain prediction intervals with conformal prediction. The first step is to call the `cvforecast()` function with the required arguments, as shown below.

```
# This step takes on average 4 minutes and 20 seconds on a MacBook Pro machine with 16 GB of
# RAM and a 2.3 GHz 8-Core Intel(R) Core(TM) i9-9889H CPU.
library(conformalForecast)
pred_arima <- cvforecast(d_dengue, forecastfunction,
                   h = 4, level = 95, initial = 1, window = 52*4)
pred_nnls <- cvforecast(d_dengue, nnlsreg,
                        h = 4, level = 95, window = 52*4)
```

Once forecasts have been obtained using `cvforecast()`, prediction intervals can be generated using conformal prediction methods implemented in **conformalForecast** package, see Wang and Hyndman (2026) for more details and a complete list of available algorithms.

In this tutorial, we focus on the AcMCP method developed by the authors, implemented via the `acmcp()` function, and the extension of the Adaptive Conformal Prediction (ACP) method implemented in the function `acp()`.

The AcMCP method applies the update rule described in Equation (10). Its first argument is the object of class `cvforecast` returned by the `cvforecast()` function, which in our case is stored as `pred`. The calibration set size should also be specified; we set `ncal = 52*4` to match the four years of calibration data (2014–2017). In addition, we set `rolling = TRUE` to use a rolling window for quantile construction. With these settings, the initial training set covers 2010-2013, while the initial calibration set corresponds to 2014-2014, as described earlier in subsection Data preparation.

The `acmcp()` function also accepts several additional parameters, which are documented in detail in Wang and Hyndman (2026) and in the package documentation. Briefly, the learning rate parameter `lr` controls the step size of the update and is set here to `lr = 0.1`. The parameter `KI` — specific to this conformal method — is a positive constant in order to place the integrator on the same scale as the nonconformity scores. If prior knowledge about the scale of the scores is available, it can be used to set

KI; in this example, we set KI = 2000. Finally, the parameter Csat is a positive constant ensuring that by time T, the absolute coverage guarantee is at least $1 - \alpha - \delta$, where δ reduces the desired coverage guarantee $1 - \alpha$ because, unlike traditional conformal prediction methods that are conservative in finite samples, the coverage guarantees here are asymptotic. This reduction δ is *controlled* by the argument delta and can be interpreted as the amount of coverage one is willing to forgo given that we cannot observe the process for an infinite period of time. The expression for Csat in the acmcp function is defined in Appendix C of Angelopoulos et al. (2023).

```
# This step takes on average 16 seconds on a MacBook Pro machine with 16 GB of RAM and a
# 2.3 GHz 8-Core Intel(R) Core(TM) i9-9889H CPU.
arima_acmcp <- acmcp(pred_arima, ncal = 52*4, rolling = TRUE,
              lr = 0.1, Tg = length(d_dengue), delta = 0.01, KI = 2000)
nnls_acmcp <- acmcp(pred_nnls, ncal = 52*4, rolling = TRUE,
              lr = 0.1, Tg = length(d_dengue), delta = 0.01, KI = 2000)
```

For the ACP method, we require fewer parameters. In addition to the nonconformity scores, only two parameters are necessary: a learning rate gamma, similar to lr, and the size of the calibration set ncal.

```
# This step takes on average 7 seconds on a MacBook Pro machine with 16 GB of RAM and a
# 2.3 GHz 8-Core Intel(R) Core(TM) i9-9889H CPU.
arima_acp <- acp(pred_arima, gamma = 0.1, ncal = 52*4)
nnls_acp <- acp(pred_nnls, gamma = 0.1, ncal = 52*4)
```

Plot predictions and prediction intervals

We plot the prediction intervals for each forecasting horizon as follows, and the result is displayed in Figure 4. We can observe that the width of the prediction intervals grows as the forecasting horizon increases, this is an expected behavior as extending the forecast horizon generally increases forecast uncertainty.

```
plot_cpForecast <- function(obj, plot_h, ts_train, ts_cal, ts_test, ts,
                            model, label = "AcMCP"){
  coverage_mean <- coverage(obj, level = 95, window = 52)
  width_mean <- width(obj, window = 52)
  for(h in plot_h){
    caption <- sprintf("%s + %s,h = %s\nCoverage = %.2f%%\nMean width = %.2f",
                       model, label, h, coverage_mean$mean[h], width_mean$mean[h])
    plot(ts*NA, ylim = c(-500, 12500), ylab = "# Cases",
     main = caption)
    lines(ts_train, col = cols[1], lwd = 2)
    lines(ts_cal, col = cols[2], lwd = 2)
    lines(ts_test, col = cols[3], lwd = 2, lty = 3)
    lines(obj$LOWER$`95%`[, h], col = cols[5])
    lines(obj$UPPER$`95%`[, h], col = cols[5])
  }
  }
```

5 Discussion

The examples presented in this work demonstrate how the packages **pintervals**, **marginaleffects**, **probably**, and **conformalForecast** enable the practical implementation of conformal prediction for regression and time series forecasting, respectively. The main strength of these packages is their modular design, which facilitates their integration with existing R packages, and broader modeling ecosystems such as **tidymodels**. This modularity allows practitioners to incorporate conformal prediction into existing workflows with minimal changes, making distribution-free uncertainty quantification more accessible in applied settings.

This modularity allows practitioners to incorporate conformal prediction into existing workflows with minimal disruption, making distribution-free uncertainty quantification more accessible in applied settings. The **pintervals** package shows progress in this direction by including multiple nonconformity measures and allowing users to define custom ones, providing a pathway toward

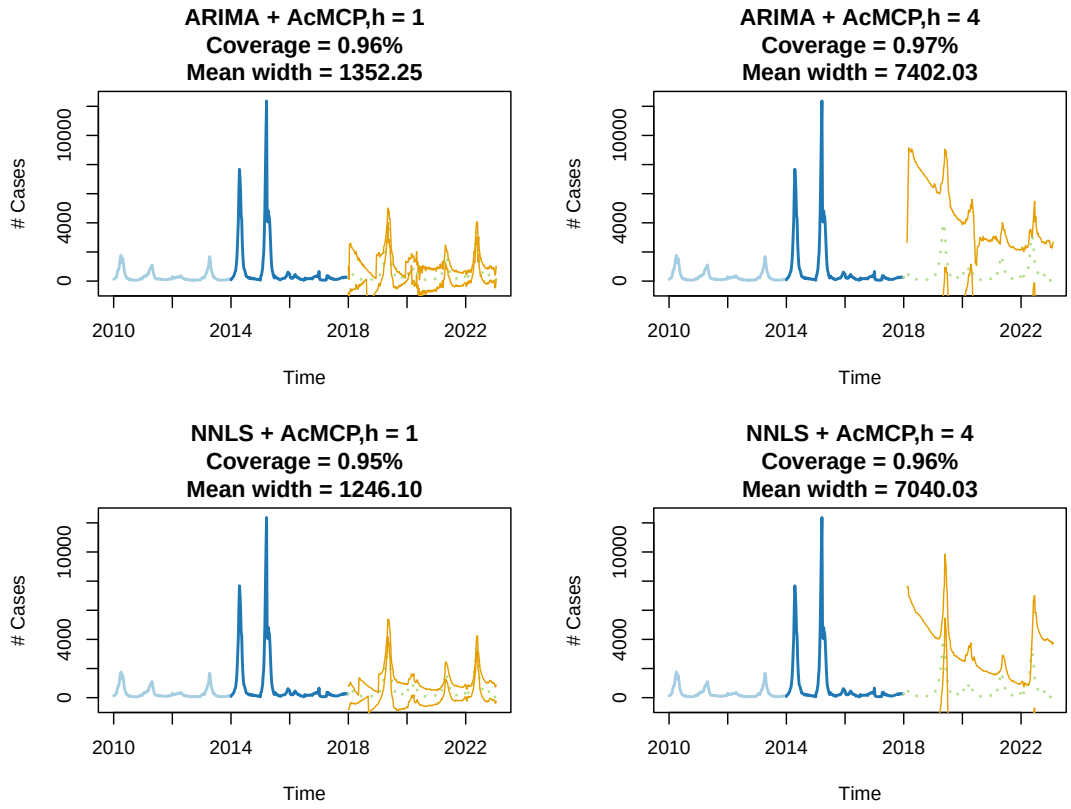

Figure 4: Prediction intervals (in orange) using AcMCP on the dengue data for two forecasting horizon for the ARIMA (first row) and NNLS models (second row).

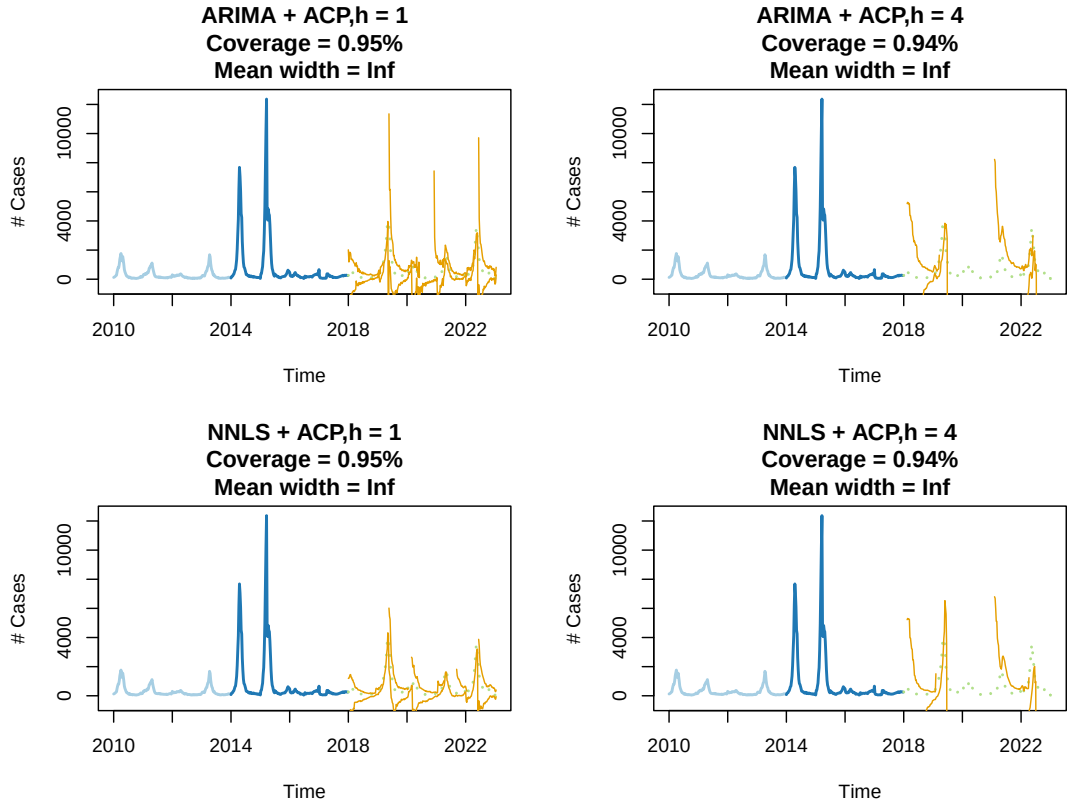

Figure 5: Prediction intervals (in orange) using ACP on the dengue data for two forecasting horizon for the ARIMA (first row) and NNLS models (second row).

more general and adaptable tools, according to their documentation. During the development of this tutorial, however, we identified practical limitations. In particular, computing split conformal prediction intervals can be computationally expensive in some settings, and the package may return missing values for prediction intervals without providing clear warnings or diagnostic messages. Addressing these issues in future releases would further improve reliability for both practitioners and researchers.

The **marginaleffects** and **probably** packages provide efficient implementations of split conformal prediction through functions designed to integrate naturally into their respective frameworks. These tools fill an important gap by enabling conformal prediction within widely used modeling pipelines. The**marginaleffects** package may be particularly well suited for users working in local computing environments, while **probably** may be more suitable for production-oriented workflows. Because both packages are part of larger modeling ecosystems with standardized objects and interfaces, they support a wide range of models and simplify the computation of nonconformity scores and prediction intervals. However, this framework-based design can also introduce limitations. Models outside these ecosystems may require additional development effort to be compatible, which can pose challenges for users without experience extending these frameworks. Future developments could focus on improving extensibility and reducing the technical barrier for integrating custom models.

For time series forecasting, the **conformalForecast** package offers a broad set of algorithms for online conformal prediction. The package includes standardized functions for computing, storing, and retrieving results, facilitating reproducible workflows. Currently, the package relies primarily on the forecast modeling framework. While this does not represent a major practical limitation, as users can construct compatible objects manually as demonstrated in our example, the development of native interfaces to additional ecosystems such as **fable** (O'Hara-Wild et al., 2026) would further enhance usability and adoption.

Beyond the packages considered in this tutorial, several specialized conformal prediction packages are available on CRAN, including **conformalbayes** (McCartan, 2025), **conformalInference.fd** (Diquigiovanni et al., 2022), **conformalInference.multi** (Diquigiovanni et al., 2025), **conformalpvalue** (Tyagi, 2023), **ConformalSmallest** (Yang, 2021), **fabPrediction** (Bersson, 2024), **ClusTorus** (Jung et al., 2022). These packages often provide computationally efficient implementations tailored to specific conformal prediction variants, though sometimes at the expense of model-agnostic flexibility. Together, they reflect the rapid evolution of conformal prediction methodology and software tools.

From a practical standpoint, users seeking seamless integration of conformal prediction into existing regression workflows may benefit most from packages embedded in large modeling ecosystems. In contrast, users requiring specialized conformal prediction variants or optimized implementations for large-scale problems may prefer dedicated conformal prediction packages, even if this reduces model flexibility. The choice of tool therefore depends strongly on the intended application, computational constraints, and existing modeling infrastructure. Looking forward, further progress will likely depend on the development of standardized data structures for storing intermediate results across the multiple stages of conformal prediction. Equally important will be the design of interfaces that clearly define interactions between predictive models, nonconformity measures, score computation, and prediction interval construction. Advances in these areas could significantly improve flexibility while maintaining coherence across software implementations.

6 Conclusions

In this paper, we showcase the **pintervals**, **marginaleffects**, **probably**, and **conformalForecast** packages for regression and time series forecasting, respectively. A key contribution of these tools is the separation of predictive model fitting from conformal inference procedures, allowing users to construct prediction intervals for a wide range of models without relying on distributional assumptions. These packages are designed for practical use and integrate naturally with existing tools in the R ecosystem.

Together, these packages highlight the importance of modular software development for statistical methods and conformal prediction naturally lends itself to such design principles, since multiple interacting components — the predictive model, the choice of nonconformity measure, the computation of nonconformity scores, and the construction of prediction intervals — must work together in a flexible yet coherent way, their design sets them apart from other existing packages that restrict the applicability of conformal prediction to a narrow class of models and offers practitioners a straightforward path to add distribution-free uncertainty quantification to their existing prediction and forecasting workflows.

A current limitation shared by these packages, and by many conformal prediction tools more broadly, is the restricted set of built-in nonconformity measures. Future development could focus on expanding the available measures and improving support for user-defined nonconformity functions, further increasing flexibility across application domains.

Overall, these four packages represent an important step toward making reliable, model-agnostic prediction intervals more accessible to the R community. As software ecosystems continue to mature, conformal prediction is likely to become an increasingly standard component of predictive modeling workflows, lowering the barrier for statisticians and data scientists to incorporate principled uncertainty quantification into applied prediction and forecasting tasks.

7 Acknowledgments

Xiaoqian Wang was supported by the Presidential Foundation of the Academy of Mathematics and Systems Science, Chinese Academy of Sciences, China (No. E555930101).

References

R. Akay. *caretForecast: Conformal Time Series Forecasting Using State of Art Machine Learning Algorithms*, 2026. URL https://CRAN.R-project.org/package=caretForecast. R package version 0.1.3. [p202]

A. Angelopoulos, E. Candes, and R. J. Tibshirani. Conformal PID control for time series prediction. *Advances in Neural Information Processing Systems*, 36:23047–23074, 2023. [p206, 216]

V. Arel-Bundock, N. Greifer, and A. Heiss. How to interpret statistical models using marginaleffects for R and Python. *Journal of Statistical Software*, 111:1–32, 2024. [p202]

R. F. Barber, E. J. Candès, A. Ramdas, and R. J. Tibshirani. Conformal prediction beyond exchangeability. *The Annals of Statistics*, 51(2), 04 2023. doi: 10.1214/23-AOS2276. [p202, 206]

E. Bersson. *fabPrediction: Compute FAB (Frequentist and Bayes) Conformal Prediction Intervals*, 2024. URL https://CRAN.R-project.org/package=fabPrediction. R package version 1.0.4. [p218]

A. Bhatnagar, H. Wang, C. Xiong, and Y. Bai. Improved online conformal prediction via strongly adaptive online learning. In *International Conference on Machine Learning*, pages 2337–2363. PMLR, 2023. [p202, 206]

H. Boström. Conformal prediction in python with crepes. *Proceedings of Machine Learning Research*, 230: 236–249, 2024. [p203]

W. Chen, K.-J. Chun, and R. F. Barber. Discretized conformal prediction for efficient distribution-free inference. *Stat*, 7(1):e173, 2018. [p202]

C. Codeco, F. Coelho, O. Cruz, S. Oliveira, T. Castro, and L. Bastos. Infodengue: A nowcasting system for the surveillance of arboviruses in brazil. *Revue d'Épidémiologie et de Santé Publique*, 66:S386, 2018. [p211]

T. Cordier, V. Blot, L. Lacombe, T. Morzadec, A. Capitaine, and N. Brunel. Flexible and systematic uncertainty estimation with conformal prediction via the mapie library. In *Conformal and Probabilistic Prediction with Applications*, pages 549–581. PMLR, 2023. [p203]

J. Diquigiovanni, M. Fontana, A. Solari, S. Vantini, P. Vergottini, and R. Tibshirani. *conformalInference.fd: Tools for Conformal Inference for Regression in Multivariate Functional Setting*, 2022. URL https://CRAN.R-project.org/package=conformalInference.fd. R package version 1.1.1. [p218]

J. Diquigiovanni, M. Fontana, A. Solari, S. Vantini, P. Vergottini, and R. Tibshirani. *conformalInference.multi: Conformal Inference Tools for Regression with Multivariate Response*, 2025. URL https://CRAN.R-project.org/package=conformalInference.multi. R package version 1.1.2. [p202, 218]

B. Erisen. Wind turbine scada dataset, 2019. URL https://www.kaggle.com/datasets/berkerisen/wind-turbine-scada-dataset. [p207]

I. Gibbs and E. Candes. Adaptive conformal inference under distribution shift. volume 34, pages 1660–1672, 2021. [p202, 206]

J. Huang, J. Song, X. Zhou, B. Jing, and H. Wei. Torchcp: A python library for conformal prediction. *Journal of Machine Learning Research*, 26(266):1–25, 2025. [p203]

S. Jung, S. Hong, K. Park, and B. Kim. *ClusTorus: Prediction and Clustering on the Torus by Conformal Prediction*, 2022. URL https://CRAN.R-project.org/package=ClusTorus. R package version 0.2.2. [p218]

Y. Kato, D. M. Tax, and M. Loog. A review of nonconformity measures for conformal prediction in regression. *Conformal and Probabilistic Prediction with Applications*, pages 369–383, 2023. [p203]

M. Kuhn, D. Vaughan, and E. Ruiz. *probably: Tools for Post-Processing Predicted Values*, 2025. URL https://CRAN.R-project.org/package=probably. R package version 1.2.0. [p202]

J. Lei, M. G'Sell, A. Rinaldo, R. J. Tibshirani, and L. Wasserman. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523):1094–1111, 07 2018. [p202, 203, 204, 205]

C. McCartan. *conformalbayes: Jackknife(+) Predictive Intervals for Bayesian Models*, 2025. URL https://CRAN.R-project.org/package=conformalbayes. [p218]

M. Mendil, L. Mossina, M. Nabhan, and K. Pasini. Robust gas demand forecasting with conformal prediction. In *Conformal and Probabilistic Prediction with Applications*, pages 169–187. PMLR, 2022. [p202]

M. O'Hara-Wild, R. Hyndman, and E. Wang. *fable: Forecasting Models for Tidy Time Series*, 2026. URL https://CRAN.R-project.org/package=fable. R package version 0.5.0. [p218]

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019. [p203]

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. [p203]

N. Pya. *scam: Shape Constrained Additive Models*, 2025. URL https://CRAN.R-project.org/package=scam. R package version 1.2-20. [p209]

D. Randahl. *pintervals: Model Agnostic Prediction Intervals*, 2026. URL https://CRAN.R-project.org/package=pintervals. R package version 1.1.1. [p202]

Y. Romano, E. Patterson, and E. Candes. Conformalized quantile regression. *Advances in Neural Information Processing Systems*, 32, 2019. [p202, 203]

G. Shafer and V. Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(12): 371–421, 2008. [p202, 203]

R. J. Tibshirani, R. Foygel Barber, E. Candes, and A. Ramdas. Conformal prediction under covariate shift. *Advances in Neural Information Processing Systems*, 32, 2019. [p202]

C. Tyagi. *conformalpvalue: Computes Conformal p-Values*, 2023. URL https://CRAN.R-project.org/package=conformalpvalue. R package version 0.1.0. [p218]

J. Vazquez and J. C. Facelli. Conformal prediction in clinical medical sciences. *Journal of Healthcare Informatics Research*, 6(3):241–252, 2022. [p202]

S. Vilfroy, L. Bombrun, T. Urruty, F. De Grancey, J.-P. Lebrat, and P. Carré. Conformal prediction for regression models with asymmetrically distributed errors: application to aircraft navigation during landing maneuver. *Machine Learning*, 113(10):7841–7866, 09 2024. [p202]

V. Vovk, A. Gammerman, and G. Shafer. *Algorithmic learning in a random world*. Springer, second edition, 2022. [p202, 203, 204]

X. Wang and R. Hyndman. *conformalForecast: Conformal Prediction Methods for Multistep-Ahead Time Series Forecasting*, 2025. URL https://CRAN.R-project.org/package=conformalForecast. R package version 0.1.0. [p202, 206, 211]

X. Wang and R. J. Hyndman. Online conformal inference for multi-step time series forecasting, 2026. URL https://arxiv.org/abs/2410.13115. [p202, 203, 206, 211, 215]

C. Xu and Y. Xie. Conformal prediction interval for dynamic time-series. In *International Conference on Machine Learning*, pages 11559–11569. PMLR, 2021. [p206]

Y. Yang. *ConformalSmallest: Efficient Tuning-Free Conformal Prediction*, 2021. URL https://CRAN.R-project.org/package=ConformalSmallest. R package version 1.0. [p202, 218]

M. Zaffran, O. Féron, Y. Goude, J. Josse, and A. Dieuleveut. Adaptive conformal predictions for time series. In *International Conference on Machine Learning*, pages 25834–25866. PMLR, 2022. [p202, 206]

Jesus Uriel Diaz-Martinez
King Abdullah University of Science and Technology (KAUST)
Computer, Electrical and Mathematical Sciences and Engineering Division
Department of Statistics
Thuwal, 23955-6900, Kingdom of Saudi Arabia
jesus.diazmartinez@kaust.edu.sa

Xiaoqian Wang
Academy of Mathematics and Systems Science, Chinese Academy of Sciences
The Center for Forecasting Science (corresponding author)
Beijing, China
https://xqnwang.rbind.io/
ORCiD: 0000-0003-4827-496X
xiaoqian.wang@amss.ac.cn

Paula Moraga
King Abdullah University of Science and Technology (KAUST)
Computer, Electrical and Mathematical Sciences and Engineering Division
Department of Statistics
Thuwal, 23955-6900, Kingdom of Saudi Arabia
http://www.paulamoraga.com
ORCiD: 0000-0001-5266-0201
paula.moraga@kaust.edu.sa