pintervals: an R package for model-agnostic prediction intervals

David Randahl
Dep. of War Studies
Swedish Defense University

Anders Hjort
Eiendomsverdi AS

Jonathan P. Williams
Dep. of Statistics
North Carolina State University

Abstract

The **pintervals** package aims to provide a unified framework for constructing prediction intervals and calibrating predictions in a model-agnostic setting using set-aside calibration data. It comprises routines to construct conformal as well as parametric and bootstrapped prediction intervals from any model that outputs point predictions. Several R packages and functions already exist for constructing prediction intervals, but they often focus on specific modeling frameworks or types of predictions, or require manual customization for different models or applications. By providing a consistent interface for a variety of prediction interval construction approaches (all model-agnostic), **pintervals** allows researchers to apply and compare them across different modeling frameworks and applications.

Keywords: Prediction intervals, Conformal prediction, Bootstrapping, R.

1. Introduction

Forecasting has in recent years become an increasingly central component of quantitative research in the social sciences. From predicting civil conflict and mass atrocities to anticipating electoral outcomes, economic downturns, and policy adoption, social scientists are increasingly engaged in producing predictive models for a range of empirical phenomena (see for instance Hegre, Metternich, Nygård, and Wucherpfennig 2017; Nanlohy, Butcher, and Goldsmith 2017; Lyu, Nie, and Yang 2021; Hegre, Vesco, Colaresi, Vestby, Timlick, Kazmi, Lindqvist-McGowan, Becker, Binetti, Bodentien *et al.* 2024; Petropoulos, Apiletti, Assimakopoulos, Babai, Barrow, Taieb, Bergmeir, Bessa, Bijak, Boylan *et al.* 2022). This shift reflects a growing recognition that forecasting can complement traditional causal inference by providing actionable, forward-looking insights to policymakers, civil society, and researchers alike (Nanlohy *et al.* 2017).

Most of these predictive models provide a single estimate of an outcome of interest, such as the expected number of conflict-related fatalities in a country-month (e.g. Hegre, Allansson, Basedau, Colaresi, Croicu, Fjelde, Hoyles, Hultman, Högbladh, Jansen *et al.* 2019; Hegre, Bell, Colaresi, Croicu, Hoyles, Jansen, Leis, Lindqvist-McGowan, Randahl, Rød *et al.* 2021; Hegre *et al.* 2024), or the probability of electoral violence in a certain election (e g. Randahl, Leis, Gåsste, Fjelde, Hegre, Lindberg, and Wilson 2025a). While such point predictions can offer valuable insights they do not reflect the inherent uncertainty of the predictions, limiting utility and potentially leading to overconfidence in the forecasts. In many applications, understanding the uncertainty around point predictions is crucial for effective decision-making, as it provides a more complete picture of the risks and potential outcomes associated with different scenarios (Raftery 2016). For instance, in conflict forecasting, it may be more important for stakeholders, in order to effectively prioritize their resources, to know the likelihood that a conflict exceeds a certain threshold of fatalities, rather than just the expected number of fatalities. Similarly, the width of a prediction interval informs decision-makers about the level of confidence they may place in a particular point prediction, which is essential for risk assessment and planning.

These limitations have led to a growing consensus that point predictions alone are not sufficient for effective communication with stakeholders or for providing actionable support to decision-makers, especially in high-stakes domains such as conflict forecasting. When forecasts inform real-world interventions, it is not enough to know what the most likely outcome is; decision-makers must also understand how much uncertainty encompasses the most likely outcome, and how much confidence they can place in it (Raftery 2016; Hegre *et al.* 2024; Brandt, Freeman, and Schrodt 2014; Chadefaux 2017). For example, a forecast suggesting a moderate risk of violence may warrant very different policy responses depending on whether the associated uncertainty is low or high. Transparent communication of predictive uncertainty can therefore help ensure that responses are proportionate to the level of risk, improving both reliability and accountability.

To quantify the uncertainty around predictions, researchers commonly turn to prediction intervals. A variety of techniques exist for constructing such intervals, including parametric methods (i.e. assuming a specific distribution for the prediction errors), bootstrapping (resampling errors from a calibration set), quantile regression, and Bayesian credible intervals. Each approach has strengths and limitations. Parametric intervals, when available analytically or approximately, may be easy to compute but rely on strong distributional or asymptotic assumptions. Bootstrap methods are flexible, but they assume that the distribution of errors is constant across the data, and they tend to underestimate variance (due to sampling from an empirical distribution function rather than the true cumulative distribution function). Quantile regression provides asymmetric intervals but requires careful specification and is model-specific. Bayesian methods allow for full posterior distributions but are also model-specific and require careful prior specification (for a review of these methods, see Tian, Nordman, and Meeker 2022; Zhang, Zimmerman, Nettleton, and Nordman 2020; Hesterberg 2015). Second, Bayesian uncertainty is epistemic rather than aleatory, meaning that credible sets cannot necessarily be interpreted as [frequentist] confidence sets; and third, Bayesian posterior inference is subject to the *false confidence theorem* (Balch, Martin, and Ferson 2019; Martin 2019; Carmichael and Williams 2018), meaning that there always exists sets not containing the true value-to-be-predicted, that are assigned arbitrarily large posterior probability, arbitrarily often (i.e. over repeated sampling of data sets). The limitations of these standard methods may result in unreliable/misleading uncertainty quantification, particularly in complex or heterogeneous data settings common in social science applications. There has thus been a demand for more-flexible, model-agnostic approaches to uncertainty quantification that offer verifiable reliability guarantees, and that can be applied across a wide range of predictive models and data types. To meet this demand, conformal prediction (CP) is proving to be a viable approach to uncertainty quantification.

Under the assumption of exchangeable data observations[1], CP produces prediction sets with

[1] Exchangeability, requiring only that the joint distribution of the data be invariant to permutations, is a slightly weaker assumption than the data being independent and identically distributed.

guaranteed coverage for any fixed sample size, at any chosen confidence level. A key advantage of CP is its model-agnostic nature: it can be applied on top of any prediction model, from simple linear regression models to complex machine learning algorithms, without requiring model-specific modifications (Shafer and Vovk 2008; Vovk, Gammerman, and Shafer 2022). Recent extensions, including clustered CP (Ding, Angelopoulos, Bates, Jordan, and Tibshirani 2023; Hjort, Williams, and Pensar 2024), weighted CP (Hjort, Hermansen, Pensar, and Williams 2025), and bin-conditional CP (Randahl, Williams, and Hegre 2025b), have further enhanced its applicability by addressing local calibration challenges in settings with skewed or imbalanced outcome distributions.

This article serves as an introduction to the **pintervals** package, and as an expository on the theoretical and computational foundations of the supported methods for uncertainty quantification. We also showcase the package using data on county election turnout in the United States presidential election of 2016. The package is designed to be flexible and extensible, allowing users to apply it to a wide range of predictive modeling tasks in the social sciences and beyond.

Several R packages and functions already exist for constructing prediction intervals. For example, the **forecast** package (Hyndman, Athanasopoulos, Bergmeir, Caceres, Chhay, O'Hara-Wild, Petropoulos, Razbash, Wang, and Yasmeen 2025) provides methods for time series forecasting, while the **tidymodels** and **mlr3** packages (Kuhn and Wickham 2020; Lang, Binder, Richter, Schratz, Pfisterer, Coors, Au, Casalicchio, Kotthoff, and Bischl 2019) offer tools for machine learning workflows. Parametric prediction intervals can be constructed using the **stats** package (for linear and generalized linear regression models; R Core Team 2025), or by manually computing intervals (when analytically available) based on standard distribution functions (e.g. `qt` for t-distribution, `qnorm` for normal distribution, etc.). Bootstrapped prediction intervals can be constructed using the **boot** package (namely, it provides functions for resampling; Angelo Canty and B. D. Ripley 2024), or by manually implementing bootstrapping procedures using the **stats** package. Conformal prediction intervals can also be obtained through a range of packages, such as the **conformalbayes** package (McCartan 2025) for Bayesian conformal prediction, the **conformalForecast** package (Wang and Hyndman 2025) for time series forecasting, and the **LCP** package (Guan 2021) for localized conformal prediction. Beyond these packages, there are also various tutorials on how to obtain conformal prediction intervals manually using functions available in packages such as **tidymodels** (Kuhn and Wickham 2020), and **marginaleffects** (Arel-Bundock, Greifer, McDermott, Bacher, and Heiss 2025; Arel-Bundock 2025).

While existing packages offer valuable tools for constructing prediction intervals, they generally lack a unified, model-agnostic interface. Most are tied to specific modeling frameworks or require extensive manual adjustments to accommodate different models or applications, making it challenging for researchers to apply and compare them consistently across datasets and prediction tasks. The **pintervals** package addresses this limitation by providing a coherent and model-agnostic framework for generating prediction intervals. It implements several widely used modern and classical approaches, including CP (with subcategories such as weighted, clustered, and bin-conditional CP), bootstrap-based intervals, and parametric methods, and is compatible with any model that produces point predictions. **pintervals** utilizes a structure where separate calibration data are used to estimate the uncertainty around predictions, allowing for flexible and robust uncertainty quantification.

2. Prediction intervals

Prediction intervals quantify the uncertainty around model predictions by providing a range within which the true outcome is stated to be contained at a given confidence level, e.g. a 90% prediction interval. Unlike confidence intervals, which characterize uncertainty about fixed but unknown model parameters, prediction intervals capture the uncertainty in subsequent data realizations, making them an essential tool for evaluating and communicating the reliability of forecasting models. In applied research, particularly in the social sciences, this distinction is crucial, as stakeholders and policymakers are often more interested in the uncertainty of predicted outcomes than in parameter inference.

The **pintervals** package implements three classes of model-agnostic prediction intervals: conformal, bootstrapped, and parametric. These methods differ in their underlying assumptions and estimation strategies but share a common interface and workflow. Both the conformal and bootstrap approaches rely on a separate calibration set containing observed outcome values, used to calibrate prediction intervals based on the empirical distribution of prediction errors. Parametric intervals, in contrast, assume a specified distribution for the prediction errors of which the parameters can either be estimated from calibration data or supplied directly. Together, the software package provides a flexible and coherent framework for quantifying predictive uncertainty across a wide range of models and applications.

2.1. Conformal prediction

CP is a general, distribution-free method that produces prediction intervals with guaranteed marginal coverage, with minimal assumptions on the underlying prediction model. The method relies on the construction of a *non-conformity score* to quantify a degree of dissimilarity of a given datum point versus a collection of data examples. Under the assumption that non-conformity scores are exchangeable, their empirical distribution can be used to construct a p-value for testing the null hypothesis that the non-conformity score of a test point is exchangeable with the observed data scores, e.g. implying whether or not to rule out the given test point as a plausible predicted outcome. The conformal p-value is a usual frequentist p-value, in that it is stochastically no smaller than a uniform(0,1) random variable, it gives finite-sample control of type 1 errors, and it corresponds to a prediction interval. Namely, a CP interval at level $1-\alpha$ contains all outcome values for which their conformal p-value exceeds α. For a comprehensive background on CP, see Vovk *et al.* (2022), and for its connections to other statistical paradigms, see Williams (2023); Williams and Liu (2024).

The CP algorithm constructs a prediction interval for y_{N+1} given a corresponding feature x_{N+1}, by comparing to a training set of feature-response pairs $(x_1, y_1), \ldots, (x_N, y_N)$. The non-conformity score s_i may measure the dissimilarity between a feature-response pair (x_i, y_i) and the remaining data points $(x_1, y_1), \ldots, (x_{i-1}, y_{i-1}), (x_{i+1}, y_{i+1}), ..., (x_{N+1}, y_{N+1})$. Although the distribution of the data and of the non-conformity scores are both unknown, assuming exchangeability implies that s_{N+1} is equally likely to take any rank among the $N+1$ scores, i.e. $\text{rank}(s_{N+1}) \sim \text{uniform}\{1, \ldots, N+1\}$. Consequently, if $\hat{q}_{1-\alpha}$ denotes the $1-\alpha$ empirical quantile of the calibration scores, then, assuming exchangeability, $\mathsf{P}(s_{N+1} \leq \hat{q}_{1-\alpha}) \geq 1-\alpha$. In other words, $\text{rank}(s_{N+1})/(N+1)$ is a *p-value* of the null hypothesis that s_{N+1} is indeed exchangeable with the $s_1, \ldots, s_N$. A CP set, at any level $1-\alpha \in [0,1]$, can be assembled by considering all values that y_{N+1} could take and only including those for which $\text{rank}(s_{N+1})/(N+1) > \alpha$.

In a supervised setting, we typically create a prediction model $\hat{f}$ such that $\hat{f}(x_{N+1})$ represents the model's best guess of y_{N+1}. In a regression setting, the canonical non-conformity score is perhaps the absolute residual, $s_i := |y_i - \hat{f}(x_i)|$, while in classification a standard choice is *one minus the softmax probability* assigned to the true class. In either case, the CP procedure yields a marginal prediction set

$$C_{1-\alpha}(x_{N+1}) := \{y \in \mathcal{Y} \,:\, s_{N+1}(y) \leq \hat{q}_{1-\alpha}\},$$

where $s_{N+1}(y)$ denotes the non-conformity score corresponding to $y_{N+1} = y$. This construction ensures the following finite-sample marginal coverage guarantee,

$$\mathsf{P}\{Y_{N+1} \in C_{1-\alpha}(X_{N+1})\} \geq 1 - \alpha,$$

which holds with virtually no assumptions on the underlying prediction model or choice of non-conformity score.

There are two main variants of the CP algorithm: the transductive (or *full CP*) and inductive (or *split CP*) algorithm. The transductive utilizes the entire training data, making it maximally statistically efficient, but (typically) requires re-training of the prediction model for each test value y, making it computationally costly. The inductive version trades off some share of statistical efficiency for a gain in computational efficiency, by setting aside a portion of the data as a calibration set, untouched during training, but requiring only a single training of the prediction model. The resulting prediction intervals constructed are typically evaluated based on empirical coverage on a test set, with reference to the nominal $1-\alpha$ confidence level, for each $\alpha \in [0, 1]$; if the *coverage gap* exceeds usual Monte Carlo simulation variation, then the implication is that the non-conformity scores are *not* exchangeable. In the latter case, alternative formulations of the non-conformity measure and/or prediction model may better account for structural components of the data, to yield exchangeable scores. In practical applications, the size of the prediction intervals are also of interest, as "tighter" intervals are more informative to the user.

Mondrian conformal prediction

While standard CP provides valid coverage guarantees marginally, it does not account for potential heterogeneity in the data, which can lead to over- or under-coverage in certain regions of the data. To address this, Mondrian CP (MCP), or label-conditional CP, extends the conformal framework by partitioning the calibration set into groups based on covariates or other characteristics. The disjoint groups enable the construction of group-conditional prediction intervals, i.e. resulting in a CP set for each group. Supposing the feature space $\mathcal{X}$ can be partitioned into disjoint subsets $\mathcal{X}_1, \ldots, \mathcal{X}_K$, MCP guarantees, for each $k \in \{1, \ldots, K\}$ and $\alpha \in [0, 1]$, that

$$\mathsf{P}\{Y_{N+1} \in C_{1-\alpha}(x_{N+1}) \mid x_{N+1} \in \mathcal{X}_k\} \geq 1 - \alpha.$$

In MCP, the calibration step is performed separately per group rather than across the entire calibration data, involving the computation (for each α) of K, $1 - \alpha$ level quantiles $\hat{q}^{(1)}_{1-\alpha}, \ldots, \hat{q}^{(K)}_{1-\alpha}$. While group-conditional validity may hold in MCP, the limiting factor for efficiency (e.g. narrow intervals) is that a sufficient number of calibration points must be available in each group. Finally, it is known (Lei and Wasserman 2013) that exact conditional coverage guarantees, i.e. achieving valid coverage conditional on the exact value x_{N+1} rather than on a Mondrian group membership, are impossible in the context of CP.

Clustered conformal prediction

Clustered conformal prediction (CCP) is an extension of MCP for situations where the number of groups K is large and/or some groups contain too few calibration points to efficiently estimate the required quantile. The approach, first proposed in Ding *et al.* (2023), is to cluster the K initial groups into $M << K$ clusters, and perform calibration at the cluster level rather than at the group level. By pooling information across similar groups, CCP increases the statistical efficiency and may also help to avoid numerical issues arising from too few examples per group.

The key idea of CCP is to cluster together groups whose empirical distribution of non-conformity scores are similar. As shown in Ding *et al.* (2023), the expected coverage gap for any given group depends directly on the similarity between the groups clustered together (note that CCP guarantees validity at the cluster level). When the empirical cumulative distribution functions clustered together are similar, the coverage gap is small. In practice, clustering can be done, for example, through k-means-clustering or based on the Kolmogorov-Smirnov distance between empirical cumulative distribution functions (Ding *et al.* 2023; Hjort *et al.* 2024).

Distance weighted conformal prediction

Another modification of the CP procedure is a class of methods referred to as *weighted* CP. Rather than calculating the quantile of interest once (or once per group or cluster), weighted CP calculates $\hat{q}_{1-\alpha}$ as a *weighted* quantile, where the scores $s_1, \ldots, s_N$ are weighted according to some weights $w_1, \ldots, w_N$. By calculating a weighted quantile, the method places more emphasis on non-conformity scores that are determined closer to exchangeable with the test instance, e.g. with reference to time or space. For example, a weighted CP framework could take weights of the form $w_i \propto \exp\{-\text{dist}(x_i, x_{N+1})\}$, where $\text{dist}(x_i, x_{N+1})$ is an application-specific distance function quantifying the distance between calibration and test features. The prediction interval for y_{N+1} is then calibrated using the weighted $1-\alpha$ quantile of the scores.

The literature contains several variations of this, including data-driven weights to account for covariate shift (Tibshirani, Foygel Barber, Candes, and Ramdas 2019), fixed weights to handle non-exchangeable data (Foygel Barber, Candès, Ramdas, and Tibshirani 2023), spatial weights (Mao, Martin, and Reich 2023), and feature-distance weights (Guan 2023). The coverage guarantees of the method depend on the choice of weighting function; see Barber and Tibshirani (2025) for a review of the statistical properties of weighted CP methods.

Bin-conditional conformal prediction

It is known that the standard CP methods often under-cover in lower density regions of the outcome space $\mathcal{Y}$, while over-covering in higher density regions of $\mathcal{Y}$ (Guan 2023). While MCP and CCP are useful for addressing under- and over-coverage for different groups in the data, these methods are not easily applicable if the outcome space has no natural or well-defined grouping structure. One example is data arising from skewed distributions. Bin-conditional conformal prediction (BCCP) addresses this issue by partitioning the calibration data into user-defined bins of the outcome space, such as low, medium, and high values (Randahl *et al.*

2025b).

For BCCP, non-conformity scores are calculated on a calibration data set within each user-defined bin. As the true bin is not known for new test points, for a new test point, the BCCP algorithm calculates its bin-specific non-conformity score within each bin. A CP set is then constructed to include all bins for which the bin-specific conformal p-value of the test point exceeds α. For univariate outcomes, the final prediction set is formed by either the union of disjoint intervals corresponding to each bin in the CP set, or by contiguizing the intervals using the left- and right-most endpoints across the bins in the CP set. Both formations achieve *at least* the desired coverage level, but the contiguized intervals may suffer over-coverage.

BCCP is particularly useful in cases where the outcome space is skewed or imbalanced, as it allows for more flexible and adaptive CP sets that can better reflect the distribution of the outcome variable versus standard CP. It also allows the user to define the bins based on their knowledge of the data and the specific application, providing a more tailored approach to uncertainty quantification. Similar to MCP and CCP, BCCP requires carefully balancing the number and specification of bins with the quantity of data within each bin, to maximize statistical efficiency.

2.2. Bootstrapped prediction intervals

Outside the realm of CP, another widely used approach to constructing prediction intervals is based on bootstrapping. The bootstrap is a resampling technique that approximates the sampling distribution of any quantity of interest (e.g. a point prediction) by repeatedly drawing samples with replacement from the observed data. This approach is particularly appealing when the distribution of prediction errors is unknown or complex, as it provides an empirical estimate of predictive uncertainty without requiring parametric assumptions (Beran 1990; Tian *et al.* 2022).

Bootstrapped prediction intervals are expected to achieve nominal coverage under the assumption that the calibration set is representative of the data-generating process, and that the prediction errors are approximately independent and identically distributed. The algorithm proceeds by first computing the prediction errors on the calibration set and then repeatedly resampling these errors with replacement to form an empirical error distribution (Tian *et al.* 2022; Hesterberg 2015). For each new test observation, a large number of resampled errors are added to the model's point prediction, generating a simulated distribution of possible outcomes. The lower and upper bounds of the prediction interval are then obtained by taking $\hat{q}_{\alpha/2}$ and $\hat{q}_{1-\alpha/2}$.

Several variants of the bootstrap exist, including the residual bootstrap, wild bootstrap, and parametric bootstrap. Variants differ both in how the resampling is performed, and in the assumptions they require about the underlying model or error structure.

2.3. Parametric prediction intervals

Parametric prediction intervals are constructed under the assumption that the prediction errors follow a specific probability distribution. This approach models uncertainty analytically rather than empirically, allowing the interval bounds to be derived directly from the assumed distribution and its estimated parameters. Common choices include the normal, log-normal, Poisson, or negative binomial distributions, depending on the nature of the outcome variable and the model's residuals.

For example, if the prediction errors are assumed to follow a centered normal distribution, their standard deviation can be estimated from a calibration set, and the prediction interval for a new observation is obtained by adding and subtracting the appropriate quantiles of the normal distribution around the point prediction. More generally, any distribution for which quantile functions are available can be used to construct prediction intervals, including user-specified or mixture distributions.

The main advantage of parametric prediction intervals lies in their computational efficiency and smooth, easily interpretable behavior. However, their validity depends critically on the appropriateness of the assumed error distribution: when this assumption is violated the resulting intervals may exhibit biased or unreliable coverage.

3. The pintervals package and main functions

The **pintervals** package provides a unified and model-agnostic interface for constructing prediction intervals using the methods described above. All functions in the package share a consistent syntax and workflow, allowing users to easily switch between different approaches and apply them to a wide range of predictive models. Each function takes as input a vector of predicted values for the test set, as well as predicted and true outcomes from calibration data[2], either as a two-column tibble or as separate vectors, and returns a tibble with the lower and upper bounds of the prediction intervals, along with the corresponding calibrated or uncalibrated predictions. MCP- and CCP-based functions also return the group or cluster assignment for each observation.

For methods that rely on partitioning the data, such as MCP, CCP, or BCCP, an additional grouping variable must be specified. This can be provided either as a separate vector or as a column in the calibration tibble.

Prediction intervals are constructed using the `pinterval_*()` function family, which includes the following main functions:

- `pinterval_conformal()` for standard conformal prediction intervals
- `pinterval_mondrian()` for Mondrian conformal prediction intervals
- `pinterval_ccp()` for clustered conformal prediction intervals
- `pinterval_bccp()` for bin-conditional conformal prediction intervals
- `pinterval_bootstrap()` for bootstrapped prediction intervals
- `pinterval_parametric()` for parametric prediction intervals

All functions follow the same general structure and support additional arguments to control key aspects of interval construction—such as the desired coverage level, the number of bootstrap replications, or the distributional parameters for parametric intervals. This unified design facilitates flexible experimentation and direct comparison across different interval estimation methods within a common analytical framework. The output of each function is a

[2] Calibration data are optional for parametric prediction intervals, as parameters can be supplied directly

tibble containing the point predictions along with the lower and upper bounds of the prediction intervals and any additional relevant information (e.g., group or cluster assignments for MCP and CCP).

3.1. Conformal prediction intervals

The `pinterval_conformal()` function implements standard CP intervals. It takes the predicted values from the test set and the calibration data as input, and returns a tibble with the lower and upper bounds of the prediction intervals. The calibration data should be a two-column tibble of predicted (first column) and true (second column) values from the calibration set; alternatively the predicted values can be passed as a vector for the argument `calib` and the true values as a vector for the argument `calib_true`.

As additional arguments, the user can specify the desired coverage level and the non-conformity score function to be used. Absolute errors are used by default, with relative, zero-adjusted relative, heterogeneous, and raw errors as the alternatives. The user can also specify a custom real-valued function that takes the predicted and true values as input and returns a vector of non-conformity scores. This allows for flexibility in defining the non-conformity score, based on the specific application or data characteristics.

Mondrian and clustered conformal prediction intervals

The functions `pinterval_mondrian()` and `pinterval_ccp` implement MCP and CCP, which allows for partitioning the calibration set into disjoint groups based on a grouping variable.

The syntax for MCP and CCP builds on `pinterval_conformal()`, but requires an additional column in the calibration tibble for the grouping variable (third column). The grouping variable can also be passed as a vector for the argument `calib_group`.

For CCP, the user can also define the number of clusters to be used, or choose to optimize the number of clusters by minimizing the Caliński-Harabasz index of the proposed clustering structure (Caliński and Harabasz 1974). To avoid overfitting of the clusters, the calibration data can be further divided into one partition which is used to cluster the classes, and one partition used to calculate the resulting prediction intervals.[3]

Bin-conditional conformal prediction intervals

The function `pinterval_bccp()` implements BCCP intervals, which allows for partitioning the calibration set into user-defined bins of the outcome space. The syntax is similar to `pinterval_mondrian()` and `pinterval_ccp()`, in that it uses an additional column, `bins`, in the calibration tibble to specify the bins, or a vector for the argument `calib_bins`. The user can also specify whether to contiguize the intervals using the argument `contiguize`, which is set to `FALSE` by default. If set to `TRUE`, the function will return contiguous intervals by taking the left- and right-most endpoints across the bins, otherwise it will contiguize adjacent bin-specific intervals. When `contiguize` is `FALSE` the function outputs a tibble where the prediction intervals with corresponding lower- and upper bounds are in a list-column called `intervals`, otherwsie the function outputs a standard tibble with point predictions and the contiguized lower- and upper bounds of the intervals. The user can also specify the desired coverage level and the non-conformity score function to be used, as in `pinterval_conformal()`.

[3] For more details on guidlines for how to divide the data, see Ding *et al.* (2023).

Distance weighted conformal prediction intervals

All CP methods in **pintervals** also support distance-weighted CP (DWCP), which gives more weight to non-conformity scores from observations in the calibration set with values close to the new test observation.

DWCP is enabled by setting `distance_weighted_cp = TRUE`. DWCP additionally requires calibration and test covariates, supplied through the arguments `distance_features_calib` and `distance_features_pred`. Distances are calculated as either the Mahalanobis distance (default) or the Euclidean distance between the rows of the calibration and test covariate data. Optionally, the distances can be normalized after calculation (preferred for Euclidean distances), either through min-max normalization or by dividing by the standard deviation of the distances (z-score normalization). When distance-weighted CP is used, the non-conformity scores from the calibration set are weighted according to their distance to the new test observation before computing the quantiles for the prediction interval.

In addition, the user can specify a weight function, which translates the distances into weights for the non-conformity scores. By default, the function uses the Gaussian kernel: $w = \exp(-d^2)$, where d is the euclidean distance between the predicted value of the new test observation and the predicted values in the calibration set. Built in alternatives are the Cauchy kernel: $w = \frac{1}{1+d^2}$, the logistic kernel: $w = \frac{1}{1+\exp(d)}$, and the reciprocal linear kernel: $w = \frac{1}{1+d}$. The user can also specify a custom distance weighting function using the argument `distance_weighting_function`. This function should take a vector of distances as input and return a vector of non-negative weights, which will be used to weight the non-conformity scores. This allows for considerable flexibility in defining the distance weighting function based on the specific application or data characteristics.

3.2. Bootstrapped prediction intervals

The `pinterval_bootstrap()` function implements bootstrapped prediction intervals. It uses the prediction errors from the calibration set to mimic the distribution of errors via resampling. The function takes as input the predicted values for the test set and the calibration data, either as a two-column tibble (predicted values in the first column and true values in the second), or as separate vectors supplied to the arguments `calib` and `calib_truth`.

As additional arguments, the user can specify the desired coverage level (default is 0.9), the number of bootstrap samples (default is 1000), and the type of error to bootstrap from. The argument `error_type` can be set to `"raw"` to use raw, signed, errors, or `"absolute"` to use absolute errors with random signs. The user can also specify whether to use distance-weighted resampling by setting the argument `distance_weighted` to `TRUE`. This will give more weight to errors from observations in the calibration set with predicted values close to the new test observation, which can help improve the accuracy of the prediction intervals. The distance weighted bootstrap procedure follow the same principles as distance weighted CP described above, being controlled by the arguments `distance_features_calib`, `distance_features_pred`, `distance_type`, `normalize_distance`, and `weight_function`.

3.3. Parametric prediction intervals

The `pinterval_parametric()` function implements parametric prediction intervals based on a user-specified probability distribution. The user supplies the predicted values for the test set, a desired distribution (e.g. normal, log-normal, Poisson, negative binomial), and either a calibration set with predicted and true values *or* the parameters of the distribution.

Natively supported distributions include common continuous and count distributions such as the normal, log-normal, Poisson, and negative binomial distributions. The user specifies the distribution using the argument `dist`. For native distributions, the relevant parameters of the distribution (e.g. standard deviation from the normal distribution and dispersion parameter from the negative binomial distribution) are estimated from the calibration set. Alternatively, the user can provide the parameters directly by passing a named list with the parameters through the argument `pars`. Unlike the other `pinterval_*()` functions, `pinterval_parametric()` does not require calibration data if the parameters of the distribution are provided directly. Similarly, for one-parameter distributions (e.g. the Poisson and Chi-square distributions), no calibration data are needed as the single parameter can be estimated directly from the predicted values.

Beyond the natively supported distributions, the user can also use any arbitrary distribution by supplying a custom quantile function to the argument `dist`. This allows for considerable flexibility, including custom or mixture distributions using for instance the **mistr** package (Sablica and Hornik 2023, 2020). When specifying a custom distribution, the user needs to provide the parameters of the distribution as a named list to the argument `pars`. Ideally, these parameters should be estimated on a calibration data set. The function will then use the specified quantile function to compute the lower and upper bounds of the prediction intervals.

Parametric intervals are simple, efficient and interpretable, but rely on correct specification of the underlying distribution. Users are encouraged to verify distributional assumptions using the calibration errors before applying this method.

4. Using pintervals: An example with predicting county-level turnout

To illustrate the practical use of the **pintervals** package, we demonstrate how it can be applied to generate prediction intervals for county-level voter turnout in the 2016 U.S. presidential election. This example provides a concrete, real-world application of the methods introduced above, using a typical regression problem workflow where predictions are obtained as point-predictions from a fitted model and prediction intervals are constructed using the methods available in **pintervals** to quantify the uncertainty around these predictions.

The data are bundled with the **pintervals** package and contains county-level information on voter turnout in the 2016 U.S. presidential election, along with various demographic and socioeconomic covariates that may influence turnout rates from the MIT Election Data and Science Lab (2018) dataset. The dataset includes variables such as population size, median income, education levels, as well as racial, age, and gender composition of the counties. The outcome variable of interest is the voter turnout rate, defined as the proportion of eligible voters who cast a ballot in the election. The dataset also includes a point prediction of turnout rates obtained from a random forest regression model using all available covariates, except geographic features such as state and latitude/longitude coordinates. The point predictions were generated using a leave-one-out cross-validation procedure to ensure that all the predictions are out-of-sample.[4]

The goal of the analysis is to construct prediction intervals around the point predictions of

[4] See the documentation of the dataset in Randahl and Williams (2025) for more details.

voter turnout rates using the different methods implemented in **pintervals** and evaluate their coverage and width. We will compare the performance of the different CP methods (standard, Mondrian, clustered, and bin-conditional), as well as bootstrapped prediction intervals and parametric prediction intervals assuming normal and beta distributions of errors.

We begin the example below by loading the necessary packages, the county turnout dataset, and splitting the data into calibration and test sets. We use a 50/50 random split of the data into calibration and test sets to illustrate the procedure. In practice, one would typically use a larger calibration set to ensure accurate estimation of the prediction intervals. Importantly, the point predictions used for calibration need to be generated from out-of-sample from the model to ensure valid coverage guarantees.[5]

```
R> library(pintervals)
R> library(dplyr)
R> library(tidyr)
R> data("county_turnout", package = "pintervals")
R> # Split data into calibration and test sets
R> set.seed(20260106)
R> nobs <- nrow(county_turnout)
R> calib_indices <- sample(nobs, size = 0.5 * nobs)
R> calib_data <- county_turnout[calib_indices, ]
R> test_data <- county_turnout[-calib_indices, ]
```

We can now proceed to construct prediction intervals using the different methods available in **pintervals**. For each method, we will use the point predictions from the random forest model and the true turnout rates from the calibration set to generate the intervals for the test set. We will then evaluate the empirical coverage and average width using the `interval_coverage()` function from the package.

```
R> # Standard CP intervals
R> conformal_intervals <- pinterval_conformal(
 +   pred = test_data$predicted_turnout,
 +   calib = calib_data$predicted_turnout,
 +   calib_truth = calib_data$turnout,
 +   alpha = 0.1
 + )
R> # Evaluate coverage and width
R> conformal_coverage <- interval_coverage(
 +   truth = test_data$turnout,
 +   lower_bound = conformal_intervals$lower_bound,
 +   upper_bound = conformal_intervals$upper_bound)
R> conformal_coverage

[1] 0.9086229
```

[5] The point predictions in the dataset are generated using leave-one-out cross-validation to ensure they are out-of-sample.

We can repeat this procedure for the parametric and bootstrapped prediction intervals, adjusting the function calls as needed to accommodate the specific requirements of each method.

```
R> # Parametric prediction intervals assuming normal distribution
R> norm_intervals <- pinterval_parametric(
 +   pred = test_data$predicted_turnout,
 +   calib = calib_data$predicted_turnout,
 +   calib_truth = calib_data$turnout,
 +   dist = "norm",
 +   alpha = 0.1
 + )
 +
R> # Evaluate coverage
R> norm_coverage <- interval_coverage(
 +   truth = test_data$turnout,
 +   lower_bound = norm_intervals$lower_bound,
 +   upper_bound = norm_intervals$upper_bound)
 +
R> # Parametric prediction intervals assuming logistic distribution
R> logis_intervals <- pinterval_parametric(
 +   pred = test_data$predicted_turnout,
 +   calib = calib_data$predicted_turnout,
 +   calib_truth = calib_data$turnout,
 +   dist = "logis",
 +   alpha = 0.1
 + )
 +
R> # Evaluate coverage
R> logis_coverage <- interval_coverage(
 +   truth = test_data$turnout,
 +   lower_bound = logis_intervals$lower_bound,
 +   upper_bound = logis_intervals$upper_bound)
 +
R> # Bootstrapped prediction intervals
R> bootstrap_intervals <- pinterval_bootstrap(
 +   pred = test_data$predicted_turnout,
 +   calib = calib_data$predicted_turnout,
 +   calib_truth = calib_data$turnout,
 +   alpha = 0.1,
 +   n_bootstrap = 1000
 + )
 +
R> # Evaluate coverage
R> bootstrap_coverage <- interval_coverage(
 +   truth = test_data$turnout,
 +   lower_bound = bootstrap_intervals$lower_bound,
 +   upper_bound = bootstrap_intervals$upper_bound)
 +
R> c(norm_coverage, logis_coverage, bootstrap_coverage)

[1]  0.9105534 0.9079794 0.9066924
```

Here we can see that all methods achieve coverage close to the nominal 90% level. In the appendix, we present a more extensive simulation study comparing the performance of these methods across multiple calibration-test splits. These results show that across simulations, the SCP method achieve the most consistent coverage close to the nominal level, while the parametric methods slightly over-cover on average and the bootstrapped intervals slightly under-cover.

4.1. Subgroup coverage with Mondrian and clustered CP

Next, we demonstrate how to use Mondrian and clustered CP to improve coverage within subsets of the data. In this example, we will use the geographic group partition for the counties as the grouping variable for both methods. We will generate prediction intervals using Mondrian and clustered CP, as well as distance-weighted CP using geographic distances between counties. We will then evaluate the empirical coverage both in aggregate and within each of the four U.S. Census regions. For the CCP, we use the U.S. Census divisions as the grouping variable and optimize the number of clusters based on the Caliński-Harabasz index, setting the maximum number of clusters to 5.

```
R> # MCP intervals
R> mondrian_intervals <- pinterval_mondrian(
 +   pred = test_data$predicted_turnout,
 +        pred_class = test_data$region,
 +   calib = calib_data$predicted_turnout,
 +   calib_truth = calib_data$turnout,
 +   calib_class = calib_data$region,
 +   alpha = 0.1
 + )
R> # Evaluate coverage
R> mondrian_coverage <- interval_coverage(
 +   truth = test_data$turnout,
 +   lower_bound = mondrian_intervals$lower_bound,
 +   upper_bound = mondrian_intervals$upper_bound)
 +
R> # Clustered CP intervals
R> clustered_conformal_intervals <- pinterval_ccp(
 +   pred = test_data$predicted_turnout,
 +         pred_class = test_data$division,
 +   calib = calib_data$predicted_turnout,
 +   calib_truth = calib_data$turnout,
 +   calib_class = calib_data$division,
 +   alpha = 0.1,
 +   optimize_n_clusters = TRUE,
 +   max_n_clusters = 5,
 + )
R> # Evaluate coverage
R> clustered_conformal_coverage <- interval_coverage(
 +   truth = test_data$turnout,
 +   lower_bound = clustered_conformal_intervals$lower_bound,
 +   upper_bound = clustered_conformal_intervals$upper_bound)
 +
R> # Distance-weighted CP intervals using geographic distances
R> dw_cp_intervals <- pinterval_conformal(
 +   pred = test_data$predicted_turnout,
 +   calib = calib_data$predicted_turnout,
 +   calib_truth = calib_data$turnout,
 +   alpha = 0.1,
 +   distance_weighted_cp = TRUE,
 +   distance_features_calib = calib %>%
 +     select(latitude, longitude),
 +   distance_features_pred = test_data %>%
 +     select(latitude, longitude),
 +         normalize_distance = "sd"
 + )
R> # Evaluate coverage
R> dw_cp_coverage <- interval_coverage(
 +   truth = test_data$turnout,
 +   lower_bound = dw_cp_intervals$lower_bound,
 +   upper_bound = dw_cp_intervals$upper_bound))
R> c(mondrian_coverage, clustered_conformal_coverage, dw_cp_coverage)
```

```
[1] 0.8931789 0.9021879 0.9054054
```

We again see that the methods achieve coverage close to the nominal 90% level in aggregate. To evaluate the group-wise coverage within each U.S. Census region, we combine the results from all CP methods and compute the empirical coverage within each group.

```
R> # Adding grouping variable and true turnout values
R> conformal_intervals <- conformal_intervals %>%
 +   mutate(region = test_data$region,
 +          method = "SCP",
 +          turnout = test_data$turnout)
R> mondrian_intervals <- mondrian_intervals %>%
 +   mutate(region = test_data$region,
 +                                         method = "MCP",
 +          turnout = test_data$turnout)
R> clustered_conformal_intervals <- clustered_conformal_intervals %>%
 +   mutate(region = test_data$region,
 +                                         method = "CCP",
 +          turnout = test_data$turnout)
```

```
R> dw_cp_intervals <- dw_cp_coverage %>%
 +   mutate(region = test_data$region,
 +                                        method = "DWCP",
 +          turnout = test_data$turnout)
R> bootstrap_intervals <- bootstrap_intervals %>%
 +   mutate(region = test_data$region,
 +                                        method = "Bootstrap",
 +          turnout = test_data$turnout)
R> norm_intervals <- norm_intervals %>%
 +   mutate(region = test_data$region,
 +                                        method = "Normal",
 +          turnout = test_data$turnout)
R> logis_intervals <- logis_intervals %>%
 +   mutate(region = test_data$region,
 +                                        method = "Logistic",
 +          turnout = test_data$turnout)
 +
R> # Combine all intervals for group-wise coverage evaluation
R> all_conformal_intervals <- bind_rows(
 +   conformal_intervals,
 +   mondrian_intervals,
 +   clustered_conformal_intervals,
 +   dw_cp_intervals,
 +   bootstrap_intervals,
 +   norm_intervals,
 +   logis_intervals
 + )
R> # Evaluate group-wise coverage
R> group_wise_coverage <- all_conformal_intervals %>%
 +   group_by(method, region) %>%
 +   summarise(
 +     coverage = interval_coverage(
 +       truth = turnout,
 +       lower_bound = lower_bound,
 +       upper_bound = upper_bound
 +     ),
 +     .groups = "drop"
 +   )
R> group_wise_coverage %>%
 +    pivot_wider(
 +     names_from = region,
 +     values_from = coverage
 +   )

 A tibble: 4 × 8
  region Bootstrap   CCP  DWCP Logistic   MCP Normal   SCP
  <chr>      <dbl> <dbl> <dbl>    <dbl> <dbl>  <dbl> <dbl>
```

```
1 Midwest    0.952 0.896 0.925    0.950 0.876  0.954 0.950
2 Northeast  0.929 0.902 0.902    0.946 0.902  0.946 0.946
3 South      0.884 0.904 0.890    0.884 0.899  0.886 0.885
4 West       0.862 0.913 0.913    0.867 0.913  0.867 0.867
```

The results show that the Mondrian, clustered, and distance-weighted CP algorithms achieve coverage that is more uniform and close to the nominal level across all U.S. Census regions compared to standard CP, bootstrapped, and parametric prediction intervals.

Analyzing the results further, we below compute the mean absolute error (MAE) of coverage across regions for each method to quantify the uniformity of coverage. The results of this analysis show that the Mondrian, clustered, and distance-weighted CP have MAE values around half of the alternative methods. The simulation study in the appendix further supports these findings, showing that the group-aware and distance-weighted CP methods consistently achieve coverage closer to the nominal level within the U.S. Census regions.

```
R> group_wise_coverage %>%
 +       group_by(method) %>%
 +       summarize(mae_coverage = mean(abs(coverage - 0.9))) %>%
 +   pivot_wider(
 +               names_from = method,
 +               values_from = mae_coverage
 +       )

 A tibble: 1 × 7
  Bootstrap     CCP   DWCP Logistic     MCP Normal    SCP
      <dbl>   <dbl>  <dbl>    <dbl>   <dbl>  <dbl>  <dbl>
1    0.0338 0.00579 0.0122   0.0365 0.00994 0.0368 0.0361
```

4.2. Coverage within ranges of the target with bin-conditional CP

Finally, we illustrate how BCCP can be used to improve coverage within different ranges of the outcome variable. In this example, we will create bins for turnout rates lower than 50%, between 50% and 60%, between 60% and 65%, and above 65%[6] and generate prediction intervals using BCCP. We will then evaluate the empirical coverage both in aggregate and within each bin.

```
R> # Create bins of turnout rates in the calibration set
R> calib_data <- calib_data %>%
 +   mutate(turnout_bin = case_when(
 +     turnout < 0.5 ~ 1,
 +     turnout < 0.6 ~ 2,
 +     turnout < 0.65 ~ 3,
```

[6] These bins were chosen to roughly match the quartiles of turnout rates in the calibration set. In practice, the bins should be defined based on domain knowledge and the specific application at hand.

```
 +     TRUE ~ 4
 +   ))
 +
R> bccp_contiguized_intervals <- pinterval_bccp(
 +   pred = test_data$predicted_turnout,
 +   calib = calib_data$predicted_turnout,
 +   calib_truth = calib_data$turnout,
 +   calib_bins = calib_data$turnout_bin,
 +   alpha = 0.1,
 +   contiguize = TRUE
 + )
 +
R> # Evaluate coverage
R> bccp_contiguized_coverage <- interval_coverage(
 +   truth = test_data$turnout,
 +   lower_bound = bccp_contiguized_intervals$lower_bound,
 +   upper_bound = bccp_contiguized_intervals$upper_bound)
 +
R> bccp_discontigous_intervals <- pinterval_bccp(
 +   pred = test_data$predicted_turnout,
 +   calib = calib_data$predicted_turnout,
 +   calib_truth = calib_data$turnout,
 +   calib_bins = calib_data$turnout_bin,
 +   alpha = 0.1,
 +   contiguize = FALSE
 + )
 +
R> # Evaluate coverage
R> bccp_discontigous_coverage <- interval_coverage(
 +   truth = test_data$turnout,
 +   intervals = bccp_discontigous_intervals$intervals
 + )
 +
R> c(bccp_contiguized_coverage, bccp_discontigous_coverage)

[1] 0.9026227 0.9018326
```

Both contiguized and non-contiguized BCCP intervals achieve coverage close to the nominal 90% level in aggregate. To evaluate the bin-wise coverage within each turnout bin, we combine the results from both methods and compute the empirical coverage within each bin.

```
R> # Adding bin variable to the intervals
R> test_data <- test_data %>%
 +   mutate(turnout_bin = case_when(
 +     turnout < 0.5 ~ 1,
 +     turnout < 0.6 ~ 2,
 +     turnout < 0.65 ~ 3,
 +     TRUE ~ 4
 +   ))
 +
R> bccp_contiguized_intervals <- bccp_contiguized_intervals %>%
 +   mutate(turnout_bin = test_data$turnout_bin,
 +          method = "BCCP(c)",
 +          turnout = test_data$turnout)
 +
R> bccp_discontigous_intervals <- bccp_discontigous_intervals %>%
 +   mutate(turnout_bin = test_data$turnout_bin,
 +          method = "BCCP(d)",
 +          turnout = test_data$turnout)
 +
R> conformal_intervals <- conformal_intervals %>%
 +   mutate(turnout_bin = test_data$turnout_bin)
 +
R> bootstrap_intervals <- bootstrap_intervals %>%
 +   mutate(turnout_bin = test_data$turnout_bin)
 +
R> norm_intervals <- norm_intervals %>%
 +   mutate(turnout_bin = test_data$turnout_bin)
 +
R> logis_intervals <- logis_intervals %>%
 +   mutate(turnout_bin = test_data$turnout_bin)
 +
R> # Combine all intervals for bin-wise coverage evaluation
R> all_intervals_bins <- bind_rows(
 +   bccp_contiguized_intervals,
 +   bccp_discontigous_intervals,
 +   conformal_intervals,
 +   bootstrap_intervals,
 +   norm_intervals,
 +   logis_intervals
 + )
 +
R> # Evaluate bin-wise coverage
R> bin_wise_coverage <- all_intervals_bins %>%
 +   group_by(method, turnout_bin) %>%
 +   summarise(
 +     coverage = interval_coverage(
 +       truth = turnout,
 +       lower_bound = lower_bound,
 +       upper_bound = upper_bound,
 +             intervals = intervals
 +     ),
 +     .groups = "drop"
 +   )
 +
R> bin_wise_coverage %>%
 +    pivot_wider(
 +     names_from = turnout_bin,
 +     values_from = coverage
 +   )
```

```
# A tibble: 6 × 5
  method      `1`   `2`   `3`   `4`
  <chr>     <dbl> <dbl> <dbl> <dbl>
1 BCCP(c)   0.909 0.927 0.921 0.904
2 BCCP(d)   0.909 0.915 0.918 0.904
3 Bootstrap 0.806 0.969 0.944 0.854
4 Logistic  0.814 0.969 0.941 0.856
5 Normal    0.814 0.971 0.944 0.861
6 SCP       0.814 0.969 0.941 0.859
```

In this case, we see that both contiguized and non-contiguized BCCP achieve coverage closer to the nominal level within each turnout bin compared to standard CP, bootstrapped, and parametric prediction intervals.

Analyzing the results further, we below compute the MAE of coverage across bins for each method to quantify their performance. This analysis show that the MAE of coverage across bins is lowest for the contiguized BCCP method, followed closely by the non-contiguized version. The simulation study in the appendix further supports these findings, showing that BCCP consistently achieves coverage closer to the nominal level within different ranges of the outcome variable.

```
R> bin_wise_coverage %>%
 +       group_by(method) %>%
 +       summarize(mae_coverage = mean(abs(coverage - 0.9))) %>%
 +       pivot_wider(
 +               names_from = method,
 +               values_from = mae_coverage
 + )
```

```
# A tibble: 1 × 6
  `BCCP(c)` `BCCP(d)` Bootstrap Logistic Normal    SCP
      <dbl>     <dbl>     <dbl>    <dbl>  <dbl>  <dbl>
1    0.0155    0.0116    0.0631   0.0598 0.0598 0.0592
```

5. Conclusion

This paper has introduced the **pintervals** package for R, a unified and extensible framework for constructing prediction intervals using conformal, bootstrapped, and parametric methods. The package is designed to lower the barrier for applied researchers and practitioners to incorporate formal uncertainty quantification into their predictive analyses. By offering a consistent syntax and flexible interface, **pintervals** allows users to move seamlessly between different approaches to interval construction while maintaining transparent and reproducible workflows.

Through an empirical example predicting county-level voter turnout in the 2016 U.S. presidential election, we illustrated how the package can be applied to real-world data, and how alternative methods differ in coverage performance and interval width. The example further demonstrated how extensions such as Mondrian, clustered, bin-conditional, and distance-weighted CP can address heterogeneity and local calibration challenges across different subgroup dimensions of the data or prediction target. The results confirm that conformal methods achieve empirical coverage close to the nominal level not only overall but also within meaningful subpopulations, an important property in applied research where distributional assumptions may fail or subgroup balance is imperfect. These results are further validated in a simulation study presented in the appendix, showing the robustness and reliability of the CP methods implemented in **pintervals**.

Beyond the example presented here, **pintervals** can be readily integrated into a wide range of predictive modeling workflows, from classical regression to complex machine learning pipelines. The design emphasizes interpretability and extensibility, making it straightforward to evaluate new methods or tailor existing ones to domain-specific needs.

We hope that **pintervals** will provide an accessible, principled, and reproducible way for researchers and analysts to quantify predictive uncertainty. By combining simplicity of use with rigorous statistical foundations, the package aims to bridge the gap between methodological innovation and practical implementation, enabling more-trustworthy inference and decision-making in applied predictive modeling.

References

Angelo Canty, B D Ripley (2024). *boot: Bootstrap R (S-Plus) Functions*. R package version 1.3-31.

Arel-Bundock V (2025). *Model to Meaning: How to Interpret Statistical Models with R and Python*. CRC Press.

Arel-Bundock V, Greifer N, McDermott G, Bacher E, Heiss A (2025). *marginaleffects: Predictions, Comparisons, Slopes, Marginal Means, and Hypothesis Tests*. doi:10.32614/CRAN.package.marginaleffects. R package version 0.31.0, URL https://CRAN.R-project.org/package=marginaleffects.

Balch MS, Martin R, Ferson S (2019). "Satellite conjunction analysis and the false confidence theorem." *Proceedings of the Royal Society A*, **475**(20180565).

Barber RF, Tibshirani RJ (2025). "Unifying different theories of conformal prediction." *arXiv preprint arXiv:2504.02292*.

Beran R (1990). “Calibrating prediction regions.” *Journal of the American Statistical Association*, **85**(411), 715–723.

Brandt PT, Freeman JR, Schrodt PA (2014). “Evaluating forecasts of political conflict dynamics.” *International Journal of Forecasting*, **30**(4), 944–962.

Caliński T, Harabasz J (1974). “A dendrite method for cluster analysis.” *Communications in Statistics-theory and Methods*, **3**(1), 1–27.

Carmichael I, Williams JP (2018). “An exposition of the false confidence theorem.” *Stat*, **7**(1), e201.

Chadefaux T (2017). “Conflict forecasting and its limits.” *Data Science*, **1**(1-2), 7–17.

Ding T, Angelopoulos A, Bates S, Jordan M, Tibshirani RJ (2023). “Class-conditional conformal prediction with many classes.” *Advances in neural information processing systems*, **36**, 64555–64576.

Foygel Barber R, Candès E, Ramdas A, Tibshirani RJ (2023). “Conformal prediction beyond exchangeability.” *The Annals of Statistics*, **51**(2), 816 – 845.

Guan L (2021). *LCP: Localized conformal prediction.* R package version 1.0, commit 64e5166864379a5e53d55ca22f5fe01986cde631, URL https://github.com/LeyingGuan/LCP.

Guan L (2023). “Localized conformal prediction: A generalized inference framework for conformal prediction.” *Biometrika*, **110**(1), 33–50.

Hegre H, Allansson M, Basedau M, Colaresi M, Croicu M, Fjelde H, Hoyles F, Hultman L, Högbladh S, Jansen R, *et al.* (2019). “ViEWS: A political violence early-warning system.” *Journal of peace research*, **56**(2), 155–174.

Hegre H, Bell C, Colaresi M, Croicu M, Hoyles F, Jansen R, Leis MR, Lindqvist-McGowan A, Randahl D, Rød EG, *et al.* (2021). “ViEWS2020: revising and evaluating the ViEWS political violence early-warning system.” *Journal of peace research*, **58**(3), 599–611.

Hegre H, Metternich NW, Nygård HM, Wucherpfennig J (2017). “Introduction: Forecasting in peace research.” *Journal of Peace Research*, **54**(2), 113–124.

Hegre H, Vesco P, Colaresi M, Vestby J, Timlick A, Kazmi NS, Lindqvist-McGowan A, Becker F, Binetti M, Bodentien T, *et al.* (2024). “The 2023/24 VIEWS Prediction challenge: Predicting the number of fatalities in armed conflict, with uncertainty.” *Journal of Peace Research*, p. 00223433241300862.

Hesterberg TC (2015). “What teachers should know about the bootstrap: Resampling in the undergraduate statistics curriculum.” *The american statistician*, **69**(4), 371–386.

Hjort A, Hermansen GH, Pensar J, Williams JP (2025). “Uncertainty quantification in automated valuation models with spatially weighted conformal prediction.” *International Journal of Data Science and Analytics*, pp. 1–18.

Hjort A, Williams JP, Pensar J (2024). “Clustered conformal prediction for the housing market.” *Proceedings of Machine Learning Research*, **230**, 1–21.

Hyndman R, Athanasopoulos G, Bergmeir C, Caceres G, Chhay L, O'Hara-Wild M, Petropoulos F, Razbash S, Wang E, Yasmeen F (2025). *forecast: Forecasting functions for time series and linear models*. R package version 8.24.0, URL https://pkg.robjhyndman.com/forecast/.

Kuhn M, Wickham H (2020). *Tidymodels: a collection of packages for modeling and machine learning using tidyverse principles*. URL https://www.tidymodels.org.

Lang M, Binder M, Richter J, Schratz P, Pfisterer F, Coors S, Au Q, Casalicchio G, Kotthoff L, Bischl B (2019). "mlr3: A modern object-oriented machine learning framework in R." *Journal of Open Source Software*. doi:10.21105/joss.01903. URL https://joss.theoj.org/papers/10.21105/joss.01903.

Lei J, Wasserman L (2013). "Distribution-free Prediction Bands for Non-parametric Regression." *Journal of the Royal Statistical Society Series B: Statistical Methodology*, **76**(1), 71–96. ISSN 1369-7412. doi:10.1111/rssb.12021. https://academic.oup.com/jrsssb/article-pdf/76/1/71/49514328/jrsssb_76_1_71.pdf, URL https://doi.org/10.1111/rssb.12021.

Lyu Y, Nie J, Yang SKX (2021). "Forecasting US economic growth in downturns using cross-country data." *Economics letters*, **198**, 109668.

Mao H, Martin R, Reich BJ (2023). "Valid Model-Free Spatial Prediction." *Journal of the American Statistical Association*, **119**(546), 904–914.

Martin R (2019). "False confidence, non-additive beliefs, and valid statistical inference." *International Journal of Approximate Reasoning*, **113**, 39–73.

McCartan C (2025). *conformalbayes: Jackknife(+) Predictive Intervals for Bayesian Models*. doi:10.32614/CRAN.package.conformalbayes. R package version 0.1.4, URL https://CRAN.R-project.org/package=conformalbayes.

MIT Election Data and Science Lab (2018). "U.S. General Elections 2018 - Analysis Dataset." https://github.com/MEDSL/2018-elections-unoffical/blob/master/election-context-2018.csv. Accessed: 2025-11-03.

Nanlohy S, Butcher C, Goldsmith BE (2017). "The policy value of quantitative atrocity forecasting models." *The RUSI Journal*, **162**(2), 24–32.

Petropoulos F, Apiletti D, Assimakopoulos V, Babai MZ, Barrow DK, Taieb SB, Bergmeir C, Bessa RJ, Bijak J, Boylan JE, *et al.* (2022). "Forecasting: theory and practice." *International Journal of forecasting*, **38**(3), 705–871.

R Core Team (2025). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Raftery AE (2016). "Use and communication of probabilistic forecasts." *Statistical Analysis and Data Mining: The ASA Data Science Journal*, **9**(6), 397–410.

Randahl D, Leis M, Gåsste T, Fjelde H, Hegre H, Lindberg SI, Wilson S (2025a). "Forecasting electoral violence." *International Journal of Forecasting*. ISSN 0169-2070. URL https://www.sciencedirect.com/science/article/pii/S0169207025000871.

Randahl D, Williams JP (2025). *pintervals: Model Agnostic Prediction Intervals*. R package version 0.9.2.

Randahl D, Williams JP, Hegre H (2025b). "Bin-Conditional Conformal Prediction of Fatalities from Armed Conflict." *Political Analysis*, p. 1–13. doi:10.1017/pan.2025.10010.

Sablica L, Hornik K (2020). "mistr: A Computational Framework for Mixture and Composite Distributions." *The R Journal*, **12**(1), 283–299. doi:10.32614/RJ-2020-003. URL https://journal.r-project.org/articles/RJ-2020-003/index.html.

Sablica L, Hornik K (2023). *mistr: Mixture and Composite Distributions*. doi:10.32614/CRAN.package.mistr. R package version 0.0.6, URL https://CRAN.R-project.org/package=mistr.

Shafer G, Vovk V (2008). "A tutorial on conformal prediction." *Journal of Machine Learning Research*, **9**(3).

Tian Q, Nordman DJ, Meeker WQ (2022). "Methods to compute prediction intervals: A review and new results." *Statistical Science*, **37**(4), 580–597.

Tibshirani RJ, Foygel Barber R, Candes E, Ramdas A (2019). "Conformal Prediction Under Covariate Shift." In *Advances in Neural Information Processing Systems*, volume 32.

Vovk V, Gammerman A, Shafer G (2022). *Algorithmic Learning in a Random World*. Springer Nature.

Wang X, Hyndman R (2025). *conformalForecast: Conformal Prediction Methods for Multistep-Ahead Time Series Forecasting*. doi:10.32614/CRAN.package.conformalForecast. R package version 0.1.0, URL https://CRAN.R-project.org/package=conformalForecast.

Williams JP (2023). "Model-free generalized fiducial inference." *arXiv preprint arXiv:2307.12472*.

Williams JP, Liu Y (2024). "Decision theory via model-free generalized fiducial inference." In *Belief Functions: Theory and Applications*, volume 14909, pp. 131–139. Springer.

Zhang H, Zimmerman J, Nettleton D, Nordman DJ (2020). "Random forest prediction intervals." *The American Statistician*.

Appendix: Simulation-based evaluation of prediction interval methods

This appendix presents a simulation study designed to systematically evaluate the empirical coverage properties of the prediction interval methods implemented in the **pintervals** package. While the main text illustrates these methods using a single calibration-test split, the simulation framework assesses their performance across repeated random splits of the data. The focus is on three complementary dimensions of validity: aggregate coverage, subgroup-wise coverage, and coverage within ranges of the outcome variable. In total, we run 1000 simulation iterations for each setting, generating new calibration and test sets in each iteration.

Across all simulation settings, the nominal coverage level is fixed at $1 - \alpha = 0.9$. For each simulation run, prediction intervals are constructed using the same procedures described in the main text, and performance is summarized using (i) empirical coverage, taken as the proportion of test observations whose true outcome lies within the corresponding prediction interval, averaged across simulation runs, and (ii) the mean absolute error (MAE) of coverage relative to the nominal level. The MAE metric is defined as

$$\text{MAE} = \frac{1}{G}\sum_{g=1}^{G} |\hat{c}_g - (1-\alpha)|$$

where $\hat{c}_g$ denotes empirical coverage in group g, and G is the number of groups or bins under consideration. Lower MAE values indicate more uniform and locally valid coverage. The MAE is computed for each simulation iteration and then averaged across all runs to summarize performance.

Aggregate coverage simulations

The first set of simulations evaluates aggregate coverage, averaging across all observations in the test set without conditioning on subgroups or outcome ranges. For each simulation iteration, the data are randomly split into calibration and test sets, and prediction intervals are constructed using standard conformal prediction, parametric prediction intervals, and bootstrap-based intervals. Here, we compare the standard CP method against parametric intervals assuming normal and logistic distributions, as well as bootstrapped intervals.

The results show that all four methods achieve nominal coverage very close to the nominal level of 0.9, with MAE values below 0.01. Both of the parametric methods have slight over-coverage, while the bootstrap method has a minimal level of under-coverage.[7] These findings confirm the robustness of CP methods in achieving valid coverage without relying on distributional assumptions.

Method	Mean Coverage	MAE Coverage
Logistic	0.903	0.008
SCP	0.899	0.009
Bootstrap	0.898	0.009
Normal	0.907	0.009

[7] These patterns were confirmed by rerunning the analysis using different random seeds, with the bootstrap method achieving coverage rates of 89.7–89.9%, and the logistic method achieving coverage rates of 90.3–90.5%, and the normal method slightly higher.

Subgroup-wise coverage simulations

The second set of simulations evaluates coverage within predefined subgroups of the data. These tests are motivated by the concern that methods with correct aggregate coverage may still exhibit systematic under- or over-coverage within substantively meaningful groups.

In each simulation run, prediction intervals are constructed using standard conformal prediction (SCP), Mondrian conformal prediction (MCP), clustered conformal prediction (CCP), and distance-weighted conformal prediction (DWCP), as well as bootstrapped and parametric (logistic and normal) intervals.

Group membership is defined using geographic classifications, corresponding to U.S. Census regions and divisions, as described in the main text. Coverage is computed separately within each group, while MAE of coverage is calculated across groups to quantify uniformity.

The results show that while all methods achieve empirical coverage close to the nominal level in aggregate, the group-aware CP methods (MCP, CCP, DWCP) have group-wise coverage much closer to the nominal level compared to standard CP, bootstrapped, and parametric intervals. Specifically, MCP, CCP, and DWCP. In addition, these methods exhibit substantially lower MAE of coverage across regions compared to standard CP, bootstrapped, and parametric intervals. This indicates that the group-aware methods provide more uniform and reliable coverage within subpopulations, addressing potential heterogeneity in the data.

	Empirical Coverage by Region				Overall Metrics	
Method	Midwest	Northeast	South	West	Mean Coverage	MAE Coverage
MCP	0.898	0.890	0.898	0.894	0.897	0.022
CCP	0.904	0.884	0.894	0.880	0.895	0.024
DWCP	0.942	0.901	0.877	0.883	0.901	0.028
Bootstrap	0.948	0.927	0.873	0.844	0.898	0.040
Normal	0.956	0.937	0.882	0.851	0.907	0.040
SCP	0.949	0.929	0.873	0.847	0.899	0.040
Logistic	0.952	0.934	0.877	0.849	0.903	0.040

Coverage within ranges of the target simulations

The third set of simulations focuses on coverage within different ranges of the outcome variable. This scenario is particularly relevant when prediction errors vary systematically across the outcome space, such as in the tails of the distribution.

In each simulation iteration, the calibration data are partitioned into bins based on the observed outcome, using thresholds chosen to approximately balance the number of observations across four bins. Prediction intervals are then constructed using bin-conditional conformal prediction (BCCP), with both contiguized and non-contiguized variants, alongside standard conformal, parametric, and bootstrap-based intervals.

Coverage is computed separately within each outcome bin. As in the grouped simulations, performance is summarized using mean aggregate coverage and the MAE of bin-wise coverage relative to the nominal level.

The results indicate that both contiguized and non-contiguized BCCP achieve empirical coverage closer to the nominal level within each outcome bin compared to standard CP, bootstrapped, and parametric prediction intervals. The MAE of coverage across bins is lowest for the contiguized BCCP method, followed closely by the non-contiguized version. These findings highlight the effectiveness of BCCP in addressing heterogeneity in prediction errors across the outcome space, leading to more reliable uncertainty quantification. As expected, the aggregate coverage for the contiguized BCCP is slightly higher than nominal due to the effect of contiguization.

	Empirical Coverage by Outcome Bin				Overall Metrics	
Method	1	2	3	4	Mean Coverage	MAE Coverage
BCCP(d)	0.895	0.898	0.899	0.894	0.897	0.019
BCCP(c)	0.895	0.909	0.912	0.894	0.906	0.020
Normal	0.804	0.960	0.937	0.779	0.907	0.079
Logistic	0.796	0.957	0.934	0.773	0.903	0.081
Bootstrap	0.796	0.954	0.929	0.760	0.898	0.082
SCP	0.788	0.955	0.932	0.768	0.899	0.083

Affiliation:

David Randahl
Dep. of War Studies
Swedish Defense University
Drottning Kristinas Väg 47
114 28 Stockholm, Sweden
E-mail: david.randahl@fhs.se