A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification

Anastasios N. Angelopoulos and Stephen Bates

December 8, 2022

Abstract

Black-box machine learning models are now routinely used in high-risk settings, like medical diagnostics, which demand uncertainty quantification to avoid consequential model failures. Conformal prediction (a.k.a. conformal inference) is a user-friendly paradigm for creating statistically rigorous uncertainty sets/intervals for the predictions of such models. Critically, the sets are valid in a *distribution-free* sense: they possess explicit, non-asymptotic guarantees even without distributional assumptions or model assumptions. One can use conformal prediction with any pre-trained model, such as a neural network, to produce sets that are guaranteed to contain the ground truth with a user-specified probability, such as 90%. It is easy-to-understand, easy-to-use, and general, applying naturally to problems arising in the fields of computer vision, natural language processing, deep reinforcement learning, and so on.

This hands-on introduction is aimed to provide the reader a working understanding of conformal prediction and related distribution-free uncertainty quantification techniques with one self-contained document. We lead the reader through practical theory for and examples of conformal prediction and describe its extensions to complex machine learning tasks involving structured outputs, distribution shift, time-series, outliers, models that abstain, and more. Throughout, there are many explanatory illustrations, examples, and code samples in Python. With each code sample comes a Jupyter notebook implementing the method on a real-data example; the notebooks can be accessed and easily run by clicking on the following icons:  .

Contents

Figure 1: Prediction set examples on Imagenet. *We show three progressively more difficult examples of the class* `fox squirrel` *and the prediction sets (i.e., $\mathcal{C}(X_{\text{test}})$) generated by conformal prediction.*

1 Conformal Prediction

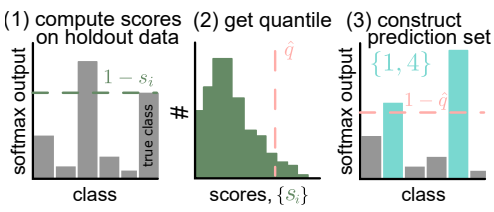
Conformal prediction [1–3] (a.k.a. conformal inference) is a straightforward way to generate prediction sets for any model. We will introduce it with a short, pragmatic image classification example, and follow up in later paragraphs with a general explanation. The high-level outline of conformal prediction is as follows. First, we begin with a fitted predicted model (such as a neural network classifier) which we will call $\hat{f}$. Then, we will create prediction sets (a set of possible labels) for this classifier using a small amount of additional *calibration data*—we will sometimes call this the *calibration step*.

Formally, suppose we have images as input and they each contain one of K classes. We begin with a classifier that outputs estimated probabilities (softmax scores) for each class: $\hat{f}(x) \in [0,1]^K$. Then, we reserve a moderate number (e.g., 500) of fresh i.i.d. pairs of images and classes unseen during training, $(X_1, Y_1), \ldots, (X_n, Y_n)$, for use as calibration data. Using $\hat{f}$ and the calibration data, we seek to construct a *prediction set* of possible labels $\mathcal{C}(X_{\text{test}}) \subset \{1, \ldots, K\}$ that is valid in the following sense:

$$1 - \alpha \leq \mathbb{P}(Y_{\text{test}} \in \mathcal{C}(X_{\text{test}})) \leq 1 - \alpha + \frac{1}{n+1}, \tag{1}$$

where $(X_{\text{test}}, Y_{\text{test}})$ is a fresh test point from the same distribution, and $\alpha \in [0,1]$ is a user-chosen error rate. In words, the probability that the prediction set contains the correct label is almost exactly $1-\alpha$; we call this property *marginal coverage*, since the probability is marginal (averaged) over the randomness in the calibration and test points. See Figure 1 for examples of prediction sets on the Imagenet dataset.

To construct $\mathcal{C}$ from $\hat{f}$ and the calibration data, we will perform a simple calibration step that requires only a few lines of code; see the right panel of Figure 2. We now describe the calibration step in more detail, introducing some terms that will be helpful later on. First, we set the *conformal score* $s_i = 1 - \hat{f}(X_i)_{Y_i}$ to be one minus the softmax output of the true class. The score is high when the softmax output of the true class is low, i.e., when the model is badly wrong. Next comes the critical step: define $\hat{q}$ to be the $\lceil (n+1)(1-\alpha) \rceil / n$ empirical quantile of $s_1, ..., s_n$, where $\lceil \cdot \rceil$ is the ceiling function ($\hat{q}$ is essentially the $1-\alpha$ quantile, but with a small correction). Finally, for a new test data point (where X_{test} is known but Y_{test} is not), create a prediction set $\mathcal{C}(X_{\text{test}}) = \{y : \hat{f}(X_{\text{test}})_y \geq 1 - \hat{q}\}$ that includes all classes with a high enough softmax output (see Figure 2). Remarkably, this algorithm gives prediction sets that are guaranteed to satisfy (1), no matter what (possibly incorrect) model is used or what the (unknown) distribution of the data is.




```python
# 1: get conformal scores. n = calib_Y.shape[0]
cal_smx = model(calib_X).softmax(dim=1).numpy()
cal_scores = 1-cal_smx[np.arange(n),cal_labels]
# 2: get adjusted quantile
q_level = np.ceil((n+1)*(1-alpha))/n
qhat = np.quantile(cal_scores, q_level, method='higher')
val_smx = model(val_X).softmax(dim=1).numpy()
prediction_sets = val_smx >= (1-qhat) # 3: form prediction sets
```

Figure 2: Illustration of conformal prediction *with matching Python code.*

Remarks

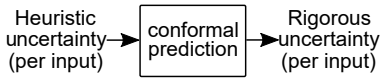
Let us think about the interpretation of $\mathcal{C}$. The function $\mathcal{C}$ is *set-valued*—it takes in an image, and it outputs a set of classes as in Figure 1. The model's softmax outputs help to generate the set. This method constructs a different output set *adaptively to each particular input.* The sets become larger when the model is uncertain or the image is intrinsically hard. This is a property we want, because the size of the set gives you an indicator of the model's certainty. Furthermore, $\mathcal{C}(X_{\text{test}})$ can be interpreted as a set of plausible classes that the image X_{test} could be assigned to. Finally, $\mathcal{C}$ is *valid*, meaning it satisfies (1).[1] These properties of $\mathcal{C}$ translate naturally to other machine learning problems, like regression, as we will see.

With an eye towards generalization, let us review in detail what happened in our classification problem. To begin, we were handed a model that had an inbuilt, but heuristic, notion of uncertainty: softmax outputs. The softmax outputs attempted to measure the conditional probability of each class; in other words, the jth entry of the softmax vector estimated $\mathbb{P}(Y = j \mid X = x)$, the probability of class j conditionally on an input image x. However, we had no guarantee that the softmax outputs were any good; they may have been arbitrarily overfit or otherwise untrustworthy. Therefore, instead of taking the softmax outputs at face value, we used the holdout set to adjust for their deficiencies.

The holdout set contained $n \approx 500$ fresh data points that the model never saw during training, which allowed us to get an honest appraisal of its performance. The adjustment involved computing conformal scores, which grow when the model is uncertain, but are not valid prediction intervals on their own. In our case, the conformal score was one minus the softmax output of the true class, but in general, the score can be any function of x and y. We then took $\hat{q}$ to be roughly the $1-\alpha$ quantile of the scores. In this case, the quantile had a simple interpretation—when setting $\alpha = 0.1$, at least 90% of ground truth softmax outputs are guaranteed to be above the level $1-\hat{q}$ (we prove this rigorously in Appendix D). Taking advantage of this fact, at test-time, we got the softmax outputs of a new image X_{test} and collected all classes with outputs above $1-\hat{q}$ into a prediction set $\mathcal{C}(X_{\text{test}})$. Since the softmax output of the new true class Y_{test} is guaranteed to be above $1-\hat{q}$ with probability at least 90%, we finally got the guarantee in Eq. (1).

1.1 Instructions for Conformal Prediction

As we said during the summary, conformal prediction is not specific to softmax outputs or classification problems. In fact, conformal prediction can be seen as a method for taking **any heuristic notion of uncertainty** from **any model** and converting it to a rigorous one (see the diagram below). Conformal prediction does not care if the underlying prediction problem is discrete/continuous or classification/regression.



We next outline conformal prediction for a general input x and output y (not necessarily discrete).

1. Identify a heuristic notion of uncertainty using the pre-trained model.
2. Define the score function $s(x, y) \in \mathbb{R}$. (Larger scores encode worse agreement between x and y.)
3. Compute $\hat{q}$ as the $\frac{\lceil (n+1)(1-\alpha) \rceil}{n}$ quantile of the calibration scores $s_1 = s(X_1, Y_1), ..., s_n = s(X_n, Y_n)$.
4. Use this quantile to form the prediction sets for new examples:

$$\mathcal{C}(X_{\text{test}}) = \{y : s(X_{\text{test}}, y) \leq \hat{q}\}\,. \tag{2}$$

As before, these sets satisfy the validity property in (1), for any (possibly uninformative) score function and (possibly unknown) distribution of the data. We formally state the coverage guarantee next.

[1] Due to the discreteness of Y, a small modification involving tie-breaking is needed to additionally satisfy the upper bound (see [4] for details; this randomization is usually ignored in practice). We will henceforth ignore such tie-breaking.

Theorem 1 (Conformal coverage guarantee; Vovk, Gammerman, and Saunders [5]). *Suppose* $(X_i, Y_i)_{i=1,\ldots,n}$ *and* $(X_{\text{test}}, Y_{\text{test}})$ *are i.i.d. and define* $\hat{q}$ *as in step 3 above and* $\mathcal{C}(X_{\text{test}})$ *as in step 4 above. Then the following holds:*

$$P\Big(Y_{\text{test}} \in \mathcal{C}(X_{\text{test}})\Big) \geq 1 - \alpha.$$

See Appendix D for a proof and a statement that includes the upper bound in (1). We note that the above is only a special case of conformal prediction, called *split conformal prediction*. This is the most widely-used version of conformal prediction, and it will be our primary focus. To complete the picture, we describe conformal prediction in full generality later in Section 6 and give an overview of the literature in Section 7.

Choice of score function

Upon first glance, this seems too good to be true, and a skeptical reader might ask the following question:

How is it possible to construct a statistically valid prediction set even if the heuristic notion of uncertainty of the underlying model is arbitrarily bad?

Let's give some intuition to supplement the mathematical understanding from the proof in Appendix D. Roughly, if the scores s_i correctly rank the inputs from lowest to highest magnitude of model error, then the resulting sets will be smaller for easy inputs and bigger for hard ones. If the scores are bad, in the sense that they do not approximate this ranking, then the sets will be useless. For example, if the scores are random noise, then the sets will contain a random sample of the label space, where that random sample is large enough to provide valid marginal coverage. This illustrates an important underlying fact about conformal prediction: although the guarantee always holds, **the usefulness of the prediction sets is primarily determined by the score function**. This should be no surprise—the score function incorporates almost all the information we know about our problem and data, including the underlying model itself. For example, the main difference between applying conformal prediction on classification problems versus regression problems is the choice of score. There are also many possible score functions for a single underlying model, which have different properties. Therefore, constructing the right score function is an important engineering choice. We will next show a few examples of good score functions.

2 Examples of Conformal Procedures

In this section we give examples of conformal prediction applied in many settings, with the goal of providing the reader a bank of techniques to practically deploy. Note that we will focus only on one-dimensional Y in this section, and smaller conformal scores will correspond to more model confidence (such scores are called *nonconformity* scores). Richer settings, such as high-dimensional Y, complicated (or multiple) notions of error, or where different mistakes cost different amounts, often require the language of *risk control*, outlined in Section A.

2.1 Classification with Adaptive Prediction Sets

Let's begin our sequence of examples with an improvement to the classification example in Section 1. The previous method produces prediction sets with the smallest average size [6], but it tends to undercover hard subgroups and overcover easy ones. Here we develop a different method called *adaptive prediction sets* (APS) that avoids this problem. We will follow [7] and [4].

As motivation for this new procedure, note that if the softmax outputs $\hat{f}(X_{\text{test}})$ were a perfect model of $Y_{\text{test}}|X_{\text{test}}$, we would greedily include the top-scoring classes until their total mass just exceeded $1 - \alpha$.

```python
# Get scores. calib_X.shape[0] == calib_Y.shape[0] == n
cal_pi = cal_smx.argsort(1)[:,::-1]; cal_srt = np.take_along_axis(cal_smx,cal_pi,axis=1).cumsum(axis=1)
cal_scores = np.take_along_axis(cal_srt,cal_pi.argsort(axis=1),axis=1)[range(n),cal_labels]
# Get the score quantile
qhat = np.quantile(cal_scores, np.ceil((n+1)*(1-alpha))/n, interpolation='higher')
# Deploy (output=list of length n, each element is tensor of classes)
val_pi = val_smx.argsort(1)[:,::-1]; val_srt = np.take_along_axis(val_smx,val_pi,axis=1).cumsum(axis=1)
prediction_sets = np.take_along_axis(val_srt <= qhat,val_pi.argsort(axis=1),axis=1)
```

Figure 3: Python code for adaptive prediction sets.

Formally, we can describe this oracle algorithm as

$$\{\pi_1(x), ..., \pi_k(x)\}, \text{ where } k = \sup\left\{k' : \sum_{j=1}^{k'} \hat{f}(X_{\text{test}})_{\pi_j(x)} < 1-\alpha\right\} + 1,$$

and $\pi(x)$ is the permutation of $\{1, ..., K\}$ that sorts $\hat{f}(X_{\text{test}})$ from most likely to least likely. In practice, however, this procedure fails to provide coverage, since $\hat{f}(X_{\text{test}})$ is not perfect; it only provides us a heuristic notion of uncertainty. Therefore, we will use conformal prediction to turn this into a rigorous notion of uncertainty.

To proceed, we define a score function inspired by the oracle algorithm:

$$s(x, y) = \sum_{j=1}^{k} \hat{f}(x)_{\pi_j(x)}, \text{ where } y = \pi_k(x).$$

In other words, we greedily include classes in our set until we reach the true label, then we stop. Unlike the score from Section 1, this one utilizes the softmax outputs of all classes, not just the true class.

The next step, as in all conformal procedures, is to set $\hat{q} = \text{Quantile}(s_1, ..., s_n\ ;\ \frac{\lceil (n+1)(1-\alpha) \rceil}{n})$. Having done so, we will form the prediction set $\{y : s(x, y) \leq \hat{q}\}$, modified slightly to avoid zero-size sets:

$$\mathcal{C}(x) = \{\pi_1(x), ..., \pi_k(x)\}, \text{ where } k = \sup\left\{k' : \sum_{j=1}^{k'} \hat{f}(x)_{\pi_j(x)} < \hat{q}\right\} + 1. \tag{3}$$

Figure 3 shows Python code to implement this method. As usual, these uncertainty sets (with tie-breaking) satisfy (1). See [4] for details and significant practical improvements, which we implemented here: .

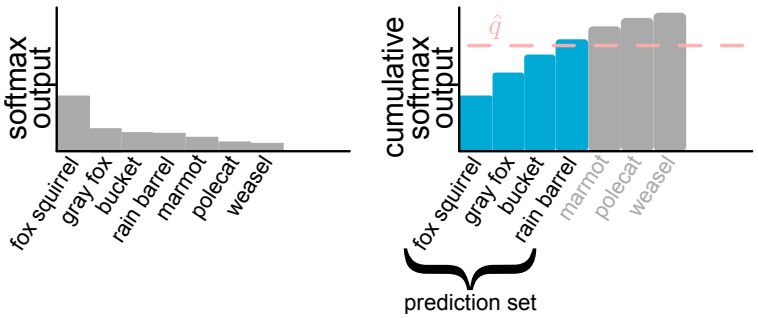


Figure 4: A visualization of the adaptive prediction sets algorithm *in Eq.* (3)*. Classes are included from most to least likely until their cumulative softmax output exceeds the quantile.*

```python
# Get scores
cal_scores = np.maximum(cal_labels-model_upper(cal_X), model_lower(cal_X)-cal_labels)
# Get the score quantile
qhat = np.quantile(cal_scores, np.ceil((n+1)*(1-alpha))/n, interpolation='higher')
# Deploy (output=lower and upper adjusted quantiles)
prediction_sets = [val_lower - qhat, val_upper + qhat]
```

Figure 5: Python code for conformalized quantile regression.

2.2 Conformalized Quantile Regression

We will next show how to incorporate uncertainty into regression problems with a continuous output, following the algorithm in [8]. We use quantile regression [9] as our base model. As a reminder, the quantile regression algorithm attempts to learn the γ quantile of $Y_{\text{test}}|X_{\text{test}} = x$ for each possible value of x. We will call the true quantile $t_\gamma(x)$ and the fitted model $\hat{t}_\gamma(x)$. Since by definition $Y_{\text{test}}|X_{\text{test}} = x$ lands below $t_{0.05}(x)$ with 5% probability and above $t_{0.95}(x)$ with 5% probability, we would expect the interval $\left[\hat{t}_{0.05}(x), \hat{t}_{0.95}(x)\right]$ to have approximately 90% coverage. However, because the fitted quantiles may be inaccurate, we will conformalize them. Python pseudocode for conformalized quantile regression is in Figure 5.

After training an algorithm to output two such quantiles (this can be done with a standard loss function, see below), $t_{\alpha/2}$ and $t_{1-\alpha/2}$, we can define the score to be the difference between y and its nearest quantile,

$$s(x, y) = \max\left\{\hat{t}_{\alpha/2}(x) - y, y - \hat{t}_{1-\alpha/2}(x)\right\}.$$

After computing the scores on our calibration set and setting $\hat{q} = \text{Quantile}(s_1, ..., s_n\ ;\ \frac{\lceil (n+1)(1-\alpha)\rceil}{n})$, we can form valid prediction intervals by taking

$$\mathcal{C}(x) = \left[\hat{t}_{\alpha/2}(x) - \hat{q}, \hat{t}_{1-\alpha/2}(x) + \hat{q}\right]. \tag{4}$$

Intuitively, the set $\mathcal{C}(x)$ just grows or shrinks the distance between the quantiles by $\hat{q}$ to achieve coverage.

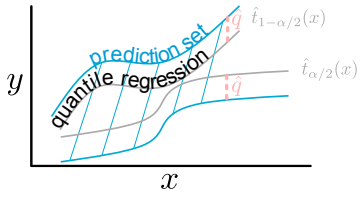
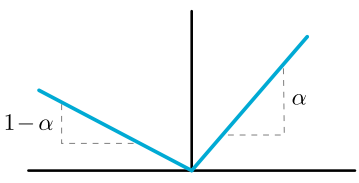


Figure 6: A visualization of the conformalized quantile regrssion algorithm *in Eq.* (4)*. We adjust the quantiles by the constant $\hat{q}$, picked during the calibration step.*

As before, $\mathcal{C}$ satisfies the coverage property in Eq. (1). However, unlike our previous example in Section 1, $\mathcal{C}$ is no longer a set of classes, but instead a *continuous interval* in $\mathbb{R}$. Quantile regression is not the only way to get such continuous-valued intervals. However, it is often the best way, especially if α is known in advance. The reason is that the intervals generated via quantile regression even without conformal prediction, i.e. $[\hat{t}_{\alpha/2}(x), \hat{t}_{1-\alpha/2}(x)]$, have good coverage to begin with. Furthermore, they have asymptotically valid conditional coverage (a concept we will explain in Section 3). These properties propagate through the conformal procedure and lead to prediction sets with good performance.

One attractive feature of quantile regression is that it can easily be added on top of any base model simply by changing the loss function to a *quantile loss* (informally referred to as a *pinball loss*),

$$L_\gamma(\hat{t}_\gamma, y) = (y - \hat{t}_\gamma)\gamma \mathbb{1}\left\{y > \hat{t}_\gamma\right\} + (\hat{t}_\gamma - y)(1-\gamma)\mathbb{1}\left\{y \le \hat{t}_\gamma\right\}.$$



The reader can think of quantile regression as a generalization of L1-norm regression: when $\gamma = 0.5$, the loss function reduces to $L_{0.5} = |\hat{t}_\gamma(x) - y|/2$, which encourages $\hat{t}_{0.5}(x)$ to converge to the conditional median. Changing γ just modifies the L1 norm as in the illustration above to target other quantiles. In practice, one can just use a quantile loss instead of MSE at the end of any algorithm, like a neural network, in order to regress to a quantile.

2.3 Conformalizing Scalar Uncertainty Estimates

2.3.1 The Estimated Standard Deviation

As an alternative to quantile regression, our next example is a different way of constructing prediction sets for continuous y with a less rich but more common notion of heuristic uncertainty: an estimate of the standard deviation $\hat{\sigma}(x)$. For example, one can produce uncertainty scalars by assuming $Y_{\text{test}} \mid X_{\text{test}} = x$ follows some parametric distribution—like a Gaussian distribution—and training a model to output the mean and variance of that distribution. To be precise, in this setting we choose to model $Y_{\text{test}} \mid X_{\text{test}} = x \sim \mathcal{N}(\mu(x), \sigma(x))$, and we have models $\hat{f}(x)$ and $\hat{\sigma}(x)$ trained to maximize the likelihood of the data with respect to $\mathbb{E}\left[Y_{\text{test}} \mid X_{\text{test=x}}\right]$ and $\sqrt{\text{Var}\left[Y_{\text{test}} \mid X_{\text{test}} = x\right]}$ respectively. Then, $\hat{f}(x)$ gets used as the point prediction and $\hat{\sigma}(x)$ gets used as the uncertainty. This strategy is so common that it is commoditized: there are inbuilt PyTorch losses, such as `GaussianNLLLoss`, that enable training a neural network this way. However, we usually know $Y_{\text{test}} \mid X_{\text{test}}$ isn't Gaussian, so even if we had infinite data, $\hat{\sigma}(x)$ would not necessarily be reliable. We can use conformal prediction to turn this heuristic uncertainty notion into rigorous prediction intervals of the form $\hat{f}(x) \pm \hat{q}\hat{\sigma}(x)$.

2.3.2 Other 1-D Uncertainty Estimates

More generally, we assume there is a function $u(x)$ such that larger values encode more uncertainty. This single number can have many interpretations beyond the standard deviation. For example, one instance of an uncertainty scalar simply involves the user creating a model for the magnitude of the residual. In that setting, the user would first fit a model $\hat{f}$ that predicts y from x. Then, they would fit a second model $\hat{r}$ (possibly the same neural network), that predicts $\left|y - \hat{f}(x)\right|$. If $\hat{r}$ were perfect, we would expect the set $\left[\hat{f}(x) - \hat{r}(x), \hat{f}(x) + \hat{r}(x)\right]$ to have perfect coverage. However, our learned model of the error $\hat{r}$ is often poor in practice.

There are many more such uncertainty scalars than we can discuss in this document in detail, including

1. measuring the variance of $\hat{f}(x)$ across an ensemble of models,
2. measuring the variance of $\hat{f}(x)$ when randomly dropping out a fraction of nodes in a neural net,
3. measuring the variance of $\hat{f}(x)$ to small, random input perturbations,
4. measuring the variance of $\hat{f}(x)$ over different noise samples input to a generative model,
5. measuring the magnitude of change in $\hat{f}(x)$ when applying an adversarial perturbation, etc.

These cases will all be treated the same way. There will be some point prediction $\hat{f}(x)$, and some uncertainty scalar $u(x)$ that is large when the model is uncertain and small otherwise (in the residual setting, $u(x) := \hat{r}(x)$, and in the Gaussian setting, $u(x) := \hat{\sigma}(x)$). We will proceed with this notation for the sake of generality, but the reader should understand that u can be replaced with any function.

Now that we have our heuristic notion of uncertainty in hand, we can define a score function,

$$s(x, y) = \frac{\left|y - \hat{f}(x)\right|}{u(x)}.$$

```
# model(X)[:,0]=E(Y|X), and model(X)[:,1]=stddev(Y|X)
scores = abs(model(calib_X)[:,0]-calib_Y)/model(calib_X)[:,1]
# Get the score quantile
qhat = torch.quantile(scores,np.ceil((n+1)*(1-alpha))/n)
# Deploy (represent sets as tuple of lower and upper endpoints)
muhat, stdhat = (model(test_X)[:,0], model(test_X)[:,1])
prediction_sets = (muhat-stdhat*qhat, muhat+stdhat*qhat)
```

Figure 7: Python code for conformalized uncertainty scalars.

This score function has a natural interpretation: it is a multiplicative correction factor of the uncertainty scalar (i.e., $s(x,y)u(x) = \left|y - \hat{f}(x)\right|$). As before, taking $\hat{q}$ to be the $\frac{\lceil (1-\alpha)(n+1) \rceil}{n}$ quantile of the calibration scores guarantees us that for a new example,

$$\mathbb{P}\left[s(X_{\text{test}}, Y_{\text{test}}) \leq \hat{q}\right] \geq 1 - \alpha \implies \mathbb{P}\left[\left|Y_{\text{test}} - \hat{f}(X_{\text{test}})\right| \leq u(X_{\text{test}})\hat{q}\right] \geq 1 - \alpha.$$

Naturally, we can then form prediction sets using the rule

$$\mathcal{C}(x) = \left[\hat{f}(x) - u(x)\hat{q}, \hat{f}(x) + u(x)\hat{q}\right]. \tag{5}$$

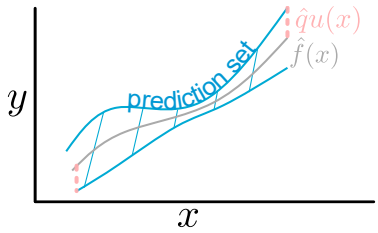


Figure 8: A visualization of the uncertainty scalars algorithm *in Eq.* (5)*. We produce the set by adding and subtracting $\hat{q}u(x)$. The constant $\hat{q}$ is picked during the calibration step.*

Let's reflect a bit on the nature of these prediction sets. The prediction sets are valid, as we desired. Due to our construction, they are also symmetric about the prediction, $\hat{f}(x)$, although symmetry could be relaxed with minor modifications. However, uncertainty scalars do not necessarily scale properly with α. In other words, there is no reason to believe that a quantity like $\hat{\sigma}$ would be directly related to quantiles of the label distribution. We tend to prefer quantile regression when possible, since it directly estimates this quantity and thus should be a better heuristic (and in practice it usually is; see [10] for some evaluations). Nonetheless, uncertainty scalars remain in use because they are easy to deploy and have been commoditized in popular machine learning libraries. See Figure 7 for a Python implementation of this method.

2.4 Conformalizing Bayes

Our final example of conformal prediction will use a Bayesian model. Bayesian predictors, like Bayesian neural networks, are commonly studied in the field of uncertainty quantification, but rely on many unverifiable and/or incorrect assumptions to provide coverage. Nonetheless, we should incorporate any prior information we have into our prediction sets. We will now show how to create valid prediction sets that are also Bayes optimal among all prediction sets that achieve $1 - \alpha$ coverage. These prediction sets use the posterior predictive density as a conformal score. The Bayes optimality of this procedure was first proven in [11], and was previously studied in [12, 13]. Because our algorithm reduces to picking the labels with high posterior predictive density, the Python code will look exactly the same as in Figure 2. The only difference is interpretation, since the softmax now represents an approximation of a continuous distribution rather than a categorical one.

Let us first describe what a Bayesian would do, given a Bayesian model $\hat{f}(y \mid x)$, which estimates the value of the posterior distribution of Y_{test} at label y with input $X_{\text{test}} = x$. If one believed all the necessary assumptions—mainly, a correctly specified model and asymptotically large n—the following would be the optimal prediction set:

$$S(x) = \left\{ y : \hat{f}(y \mid x) > t \right\}, \text{ where } t \text{ is chosen so } \int_{y \in S(x)} \hat{f}(y \mid x) dy = 1 - \alpha.$$

However, because we cannot make assumptions on the model and data, we can only consider $\hat{f}(y \mid x)$ to be a heuristic notion of uncertainty.

Following our now-familiar checklist, we can define a conformal score,

$$s(x, y) = -\hat{f}(y \mid x),$$

which is high when the model is uncertain and otherwise low. After computing $\hat{q}$ over the calibration data, we can then construct prediction sets:

$$\mathcal{C}(x) = \left\{ y : \hat{f}(y \mid x) > -\hat{q} \right\}. \tag{6}$$

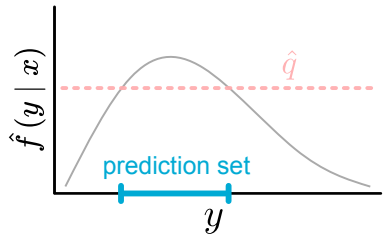


Figure 9: A visualization of the conformalized Bayes algorithm *in Eq.* (6)*. The prediction set is a superlevel set of the posterior predictive density.*

This set is valid because we chose the threshold $\hat{q}$ via conformal prediction. Furthermore, when certain technical assumptions are satisfied, it has the best Bayes risk among all prediction sets with $1 - \alpha$ coverage. To be more precise, under the assumptions in [11], $\mathcal{C}(X_{\text{test}})$ has the smallest average size of any conformal procedure with $1 - \alpha$ coverage, where the average is taken over the data *and* the parameters. This result should not be a surprise to those familiar with decision theory, as the argument we are making feels similar to that of the Neyman-Pearson lemma. This concludes the final example.

Discussion

As our examples have shown, conformal prediction is a simple and pragmatic technique with many use cases. It is also easy to implement and computationally trivial. Additionally, the above four examples serve as roadmaps to the user for designing score functions with various notions of optimality, including average size, adaptivity, and Bayes risk. Still more is yet to come—conformal prediction can be applied more broadly than it may first seem at this point. We will outline extensions of conformal prediction to other prediction tasks such as outlier detection, image segmentation, serial time-series prediction, and so on in Section 4. Before addressing these extensions, we will take a deep dive into diagnostics for conformal prediction in the standard setting, including the important topic of conditional coverage.

3 Evaluating Conformal Prediction

We have spent the last two sections learning how to form valid prediction sets satisfying rigorous statistical guarantees. Now we will discuss how to evaluate them. Our evaluations will fall into one of two categories.

1. **Evaluating adaptivity.** It is extremely important to keep in mind that the conformal prediction procedure with the smallest average set size is not necessarily the best. A good conformal prediction procedure will give small sets on easy inputs and large sets on hard inputs in a way that faithfully reflects the model's uncertainty. This *adaptivity* is not implied by conformal prediction's coverage guarantee, but it is non-negotiable in practical deployments of conformal prediction. We will formalize adaptivity, explore its consequences, and suggest practical algorithms for evaluating it.

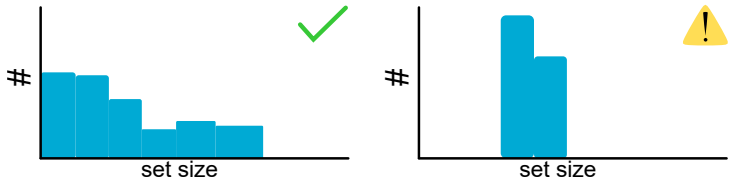
2. **Correctness checks.** Correctness checks help you test whether you've implemented conformal prediction correctly. We will empirically check that the coverage satisfies Theorem 1. Rigorously evaluating whether this property holds requires a careful accounting of the finite-sample variability present with real datasets. We develop explicit formulae for the size of the benign fluctuations—if one observes deviations from $1 - \alpha$ in coverage that are larger than these formulae dictate, then there is a problem with the implementation.

Many of the evaluations we suggest are computationally intensive, and require running the entire conformal procedure on different splits of data at least 100 times. Naïve implementations of these evaluations can be slow when the score takes a long time to compute. With some simple computational tricks and strategic caching, we can speed this process up by orders of magnitude. Therefore to aid the reader, we intersperse the mathematical descriptions with code to efficiently implement these computations.

3.1 Evaluating Adaptivity

Although any conformal procedure yields prediction intervals that satisfy (1), there are many such procedures, and they differ in other important ways. In particular, a key design consideration for conformal prediction is *adaptivity*: we want the procedure to return larger sets for harder inputs and smaller sets for easier inputs. While most reasonable conformal procedures will satisfy this to some extent, we now discuss precise metrics for adaptivity that allow the user to check a conformal procedure and to compare multiple alternative conformal procedures.

Set size. The first step is to plot histograms of set sizes. This histogram helps us in two ways. Firstly, a large average set size indicates the conformal procedure is not very precise, indicating a possible problem with the score or underlying model. Secondly, the spread of the set sizes shows whether the prediction sets properly adapt to the difficulty of examples. A wider spread is generally desirable, since it means that the procedure is effectively distinguishing between easy and hard inputs.



It can be tempting to stop evaluations after plotting the coverage and set size, but certain important questions remain unanswered. A good spread of set sizes is generally better, but it does not necessarily indicate that the sets adapt properly to the difficulty of X. Above seeing that the set sizes have dynamic range, we will need to verify that large sets occur for hard examples. We next formalize this notion and give metrics for evaluating it.

Conditional coverage. Adaptivity is typically formalized by asking for the *conditional coverage* [14] property:

$$\mathbb{P}\left[Y_{\text{test}} \in \mathcal{C}(X_{\text{test}}) \mid X_{\text{test}}\right] \geq 1 - \alpha. \tag{7}$$

That is, for every value of the input X_{test}, we seek to return prediction sets with $1-\alpha$ coverage. This is a stronger property than the *marginal coverage* property in (1) that conformal prediction is guaranteed to achieve—indeed, in the most general case, conditional coverage is impossible to achieve [14]. In other words, conformal procedures are not guaranteed to satisfy (7), so we must check how close our procedure comes to approximating it.

The difference between marginal and conditional coverage is subtle but of great practical importance, so we will spend some time think about the differences here. Imagine there are two groups of people, group A and group B, with frequencies 90% and 10%. The prediction sets always cover Y among people in group A and never cover Y when the person comes from group B. Then the prediction sets have 90% coverage, but not conditional coverage. Conditional coverage would imply that the prediction sets cover Y at least 90% of the time in both groups. This is necessary, but not sufficient; conditional coverage is a very strong property that states the probability of the prediction set needs to be $\geq 90\%$ *for a particular person.* In other words, for any subset of the population, the coverage should be $\geq 90\%$. See Figure 10 for a visualization of the difference between conditional and marginal coverage.

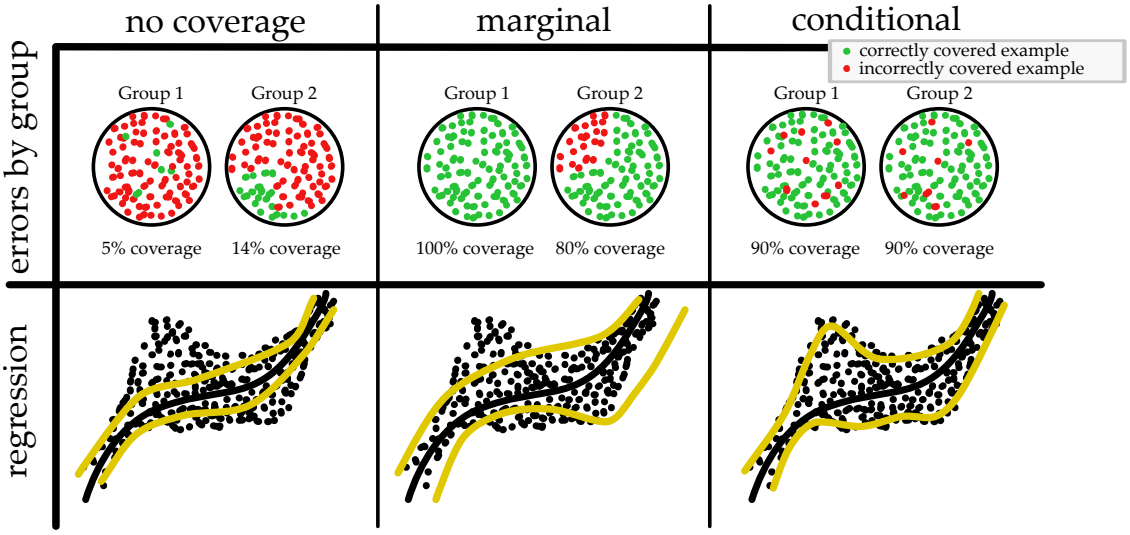


Figure 10: Prediction sets with various notions of coverage: *no coverage, marginal coverage, or conditional coverage (at a level of 90%). In the marginal case, all the errors happen in the same groups and regions in X-space. Conditional coverage disallows this behavior, and errors are evenly distributed.*

Feature-stratified coverage metric. As a first metric for conditional coverage, we will formalize the example we gave earlier, where coverage is unequal over some groups. The reader can think of these groups as discrete categories, like race, or as a discretization of continuous features, like age ranges. Formally, suppose we have features $X_{i,1}^{(\text{val})}$ that take values in $\{1,\ldots,G\}$ for some G. (Here, $i=1,\ldots,n_{\text{val}}$ indexes the example in the validation set, and the first coordinate of each feature is the group.) Let $\mathcal{I}_g \subset \{1,\ldots,n_{\text{val}}\}$ be the set of observations such that $X_{i,1}^{(\text{val})} = g$ for $g=1,\ldots,G$. Since conditional coverage implies that the procedure has the same coverage for all values of X_{test}, we use the following measure:

$$\textbf{FSC metric}: \quad \min_{g\in\{1,\ldots,G\}} \frac{1}{|\mathcal{I}_g|} \sum_{i\in\mathcal{I}_g} \mathbb{1}\left\{Y_i^{(\text{val})} \in \mathcal{C}\left(X_i^{(\text{val})}\right)\right\}$$

In words, this is the observed coverage among all instances where the discrete feature takes the value g. If conditional coverage were achieved, this would be $1-\alpha$, and values farther below $1-\alpha$ indicate a greater violation of conditional coverage. Note that this metric can also be used with a continuous feature by binning the features into a finite number of categories.

Size-stratified coverage metric. We next consider a more general-purpose metric for how close a conformal procedure comes to satisfying (7), introduced in [4]. First, we discretize the possible cardinalities of $\mathcal{C}(x)$, into G bins, $B_1, \ldots, B_G$. For example, in classification we might divide the observations into three groups, depending on whether $\mathcal{C}(x)$ has one element, two elements, or more than two elements. Let $\mathcal{I}_g \subset \{1, \ldots, n_{\text{val}}\}$ be the set of observations falling in bin g for $g = 1, \ldots, G$. Then we consider the following

$$\textbf{SSC metric}: \qquad \min_{g \in \{1,\ldots,G\}} \frac{1}{|\mathcal{I}_g|} \sum_{i \in \mathcal{I}_g} \mathbb{1}\left\{Y_i^{(\text{val})} \in \mathcal{C}\left(X_i^{(\text{val})}\right)\right\}$$

In words, this is the observed coverage for all units for which the set size $|\mathcal{C}(x)|$ falls into bin g. As before, if conditional coverage were achieved, this would be $1-\alpha$, and values farther below $1-\alpha$ indicate a greater violation of conditional coverage. Note that this is the same expression as for the FSC metric, except that the definition of $\mathcal{I}_g$ has changed. Unlike the FSC metric, the user does not have to define an important set of discrete features a-priori—it is a general metric that can apply to any example.

See [15] and [16] for additional metrics of conditional coverage.

3.2 The Effect of the Size of the Calibration Set

We first pause to discuss how the size of the calibration set affects conformal prediction. We consider this question for two reasons. First, the user must choose this for a practical deployment. Roughly speaking, our conclusion will that be choosing a calibration set of size $n = 1000$ is sufficient for most purposes. Second, the size of the calibration set is one source of finite-sample variability that we will need to analyze to correctly check the coverage. We will build on the results here in the next section, where we give a complete description of how to check coverage in practice.

How does the size of the calibration set, n, affect conformal prediction? The coverage guarantee in (1) holds for any n, so we can see that our prediction sets have coverage at least $1-\alpha$ even with a very small calibration set. Intuitively, however, it may seem that larger n is better, and leads to more stable procedures. This intuition is correct, and it explains why using a larger calibration set is beneficial in practice. The details are subtle, so we carefully work through them here.

The key idea is that *the coverage of conformal prediction conditionally on the calibration set is a random quantity*. That is, if we run the conformal prediction algorithm twice, each time sampling a new calibration dataset, then check the coverage on an infinite number of validation points, those two numbers will not be equal. The coverage property in (1) says that coverage will be at least $1-\alpha$ on average over the randomness in the calibration set, but with any one fixed calibration set, the coverage on an infinite validation set will be some number that is not exactly $1-\alpha$. Nonetheless, we can choose n large enough to control these fluctuations in coverage by analyzing its distribution.

In particular, the distribution of coverage has an analytic form, first introduced by Vladimir Vovk in [14], namely,

$$\mathbb{P}\left(Y_{\text{test}} \in \mathcal{C}\left(X_{\text{test}}\right) \,\middle|\, \{(X_i, Y_i)\}_{i=1}^n\right) \sim \text{Beta}\left(n+1-l, l\right),$$

where

$$l = \lfloor (n+1)\alpha \rfloor.$$

Notice that the conditional expectation above is the coverage with an infinite validation data set, holding the calibration data fixed. A simple proof of this fact is available in [14]. We plot the distribution of coverage for several values of n in Figure 11.

Inspecting Figure 11, we see that choosing $n = 1000$ calibration points leads to coverage that is typically between .88 and .92, hence our rough guideline of choosing about 1000 calibration points. More formally, we can compute exactly the number of calibration points n needed to achieve a coverage of $1-\alpha \pm \epsilon$ with probability $1-\delta$. Again, the average coverage is always at least $1-\alpha$; the parameter δ controls the tail probabilities of the coverage conditionally on the calibration data. For any δ, the required calibration set size n can be explicitly computed from a simple expression, and we report on several values in Table 1 for the reader's reference. Code allowing the user to produce results for any choice of n and α accompanies the table.

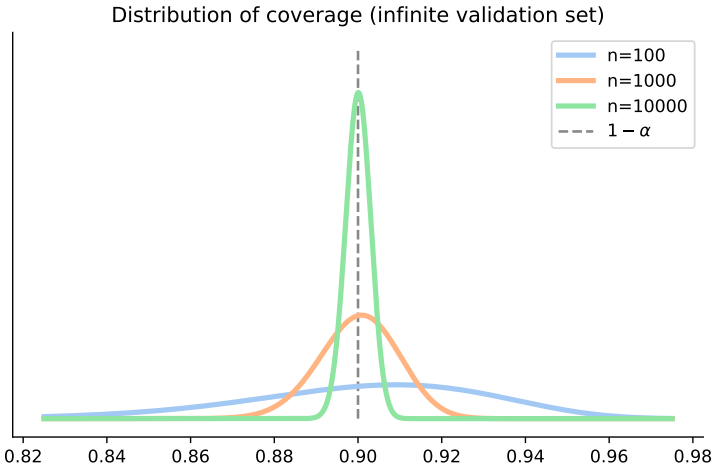


Figure 11: The distribution of coverage *with an infinite validation set is plotted for different values of* n *with* $\alpha = 0.1$*. The distribution converges to* $1 - \alpha$ *with rate* $\mathcal{O}\left(n^{-1/2}\right)$*.*

ϵ	0.1	0.05	0.01	0.005	0.001
$n(\epsilon)$	22	102	2491	9812	244390

Table 1: **Calibration set size** $n(\epsilon)$ **required** for coverage slack ϵ with $\delta = 0.1$ and $\alpha = 0.1$.

3.3 Checking for Correct Coverage

As an obvious diagnostic, the user will want to assess whether the conformal procedure has the correct coverage. This can be accomplished by running the procedure over R trials with new calibration and validation sets, and then calculating the empirical coverage for each,

$$C_j = \frac{1}{n_{\text{val}}} \sum_{i=1}^{n_{\text{val}}} \mathbb{1}\left\{Y_{i,j}^{(\text{val})} \in \mathcal{C}_j\left(X_{i,j}^{(\text{val})}\right)\right\}, \text{ for } j = 1, ..., R,$$

where n_{val} is the size of the validation set, $(X_{i,j}^{(\text{val})}, Y_{i,j}^{(\text{val})})$ is the ith validation example in trial j, and $\mathcal{C}_j$ is calibrated using the calibration data from the jth trial. A histogram of the C_j should be centered at roughly $1 - \alpha$, as in Figure 11. Likewise, the mean value,

$$\overline{C} = \frac{1}{R} \sum_{j=1}^{R} C_j,$$

should be approximately $1 - \alpha$.

With real datasets, we only have $n + n_{\text{val}}$ data points total to evaluate our conformal algorithm and therefore cannot draw new data for each of the R rounds. So, we compute the coverage values by randomly splitting the $n + n_{\text{val}}$ data points R times into calibration and validation datasets, then running conformal. Notice that rather than splitting the data points themselves many times, we can instead first cache all conformal scores and then compute the coverage values over many random splits, as in the code sample in Figure 12.

If properly implemented, conformal prediction is guaranteed to satisfy the inequality in (1). However, if the reader sees minor fluctuations in the observed coverage, they may not need to worry: the finiteness of n, n_{val}, and R can lead to benign fluctuations in coverage which add some width to the Beta distribution in Figure 11. Appendix C gives exact theory for analyzing the mean and standard deviation of $\overline{C}$. From this, we will be able to tell if any deviation from $1 - \alpha$ indicates a problem with the implementation, or

```
try: # try loading the scores first
  scores = np.load('scores.npy')
except:
  # X and Y have n + n_val rows each
  scores = get_scores(X,Y)
  np.save(scores, 'scores.npy')
# calculate the coverage R times and store in list
coverages = np.zeros((R,))
for r in range(R):
  np.random.shuffle(scores) # shuffle
  calib_scores, val_scores = (scores[:n],scores[n:]) # split
  qhat = np.quantile(calib_scores, np.ceil((n+1)*(1-alpha)/n), method='higher') # calibrate
  coverages[r] = (val_scores <= qhat).astype(float).mean() # see caption
average_coverage = coverages.mean() # should be close to 1-alpha
plt.hist(coverages) # should be roughly centered at 1-alpha
```

Figure 12: Python code for computing coverage *with efficient score caching. Notice that from the expression for conformal sets in* (2)*, a validation point is covered if and only if* $s(X, Y) \leq \hat{q}$*, which is how the third to last line is succinctly computing the coverage.*

if it is benign. Code for checking the coverage at all different values of n, n_{val}, and R is available in the accompanying Jupyter notebook of Figure 12.

4 Extensions of Conformal Prediction

At this point, we have seen the core of the matter: how to construct prediction sets with coverage in any standard supervised prediction problem. We now broaden our horizons towards prediction tasks with different structure, such as side information, covariate shift, and so on. These more exotic problems arise quite frequently in the real world, so we present practical conformal algorithms to address them.

4.1 Group-Balanced Conformal Prediction

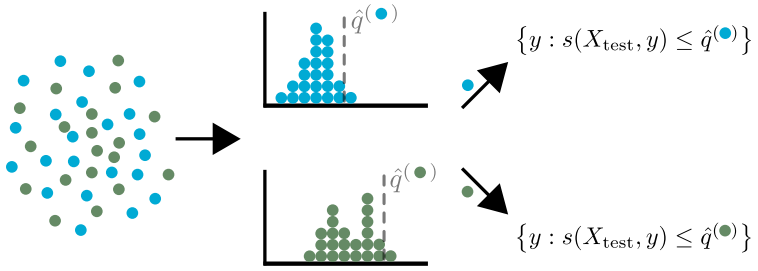
In certain settings, we might want prediction intervals that have equal error rates across certain subsets of the data. For example, we may require our medical classifier to have coverage that is correct for all racial and ethnic groups. To formalize this, we suppose that the first feature of our inputs, $X_{i,1}$, $i = 1, ..., n$ takes values in some discrete set $\{1, ..., G\}$ corresponding to categorical groups. We then ask for *group-balanced* coverage:

$$\mathbb{P}\left(Y_{\text{test}} \in \mathcal{C}(X_{\text{test}}) \mid X_{\text{test},1} = g\right) \geq 1 - \alpha, \tag{8}$$

for all groups $g \in \{1, \ldots, G\}$. In words, this means we have a $1 - \alpha$ coverage rate for all groups. Notice that the group output could be a post-processing of the original features in the data. For example, we might bin the values of X_{test} into a discrete set.

Recall that a standard application of conformal prediction will not necessarily yield coverage within each group simultaneously—that is, (8) may not be satisfied. We saw an example in Figure 10; the marginal guarantee from normal conformal prediction can still be satisfied even if all errors happen in one group.

In order to achieve group-balanced coverage, we will simply run conformal prediction seperately for each group, as visualized below.



Making this formal, given a conformal score function s, we stratify the scores on the calibration set by group,

$$s_i^{(g)} = s(X_j, Y_j), \text{ where } X_{j,1} \text{ is the } i\text{th occurrence of group } g.$$

Then, within each group, we calculate the conformal quantile

$$\hat{q}^{(g)} = \text{Quantile}\left(s_1, ..., s_{n^{(g)}}; \frac{\lceil (n^{(g)}+1)(1-\alpha) \rceil}{n^{(g)}}\right), \text{ where } n^{(g)} \text{ is the number of examples of group } g.$$

Finally, we form prediction sets by first picking the relevant quantile,

$$\mathcal{C}(x) = \left\{y : s(x, y) \leq \hat{q}^{(x_1)}\right\}.$$

That is, for a point x that we see falls in group x_1, we use the threshold $\hat{q}^{(x_1)}$ to form the prediction set, and so on. This choice of $\mathcal{C}$ satisfies (8), as was first documented by Vovk in [14].

Proposition 1 (Error control guarantee for group-balanced conformal prediction)**.** *Suppose* $(X_1, Y_1), \ldots, (X_n, Y_n), (X_{test}, Y_{test})$ *are an i.i.d. sample from some distribution. Then the set* $\mathcal{C}$ *defined above satisfies the error control property in* (8)*.*

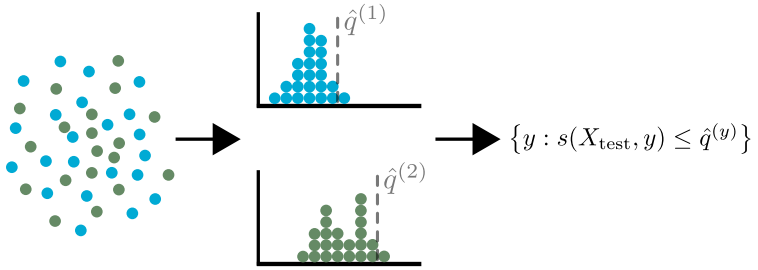
4.2 Class-Conditional Conformal Prediction

In classification problems, we might similarly ask for coverage on *every* ground truth class. For example, if we had a medical classifier assigning inputs to class normal or class cancer, we might ask that the prediction sets are 95% accurate both when the ground truth is class cancer and also when the ground truth is class normal. Formally, we return to the classification setting, where $\mathcal{Y} = \{1, ..., K\}$. We seek to achieve *class-balanced* coverage,

$$\mathbb{P}\left(Y_{\text{test}} \in \mathcal{C}(X_{\text{test}}) \mid Y_{\text{test}} = y\right) \geq 1 - \alpha, \tag{9}$$

for all classes $y \in \{1, \ldots, K\}$.

To achieve class-balanced coverage, we will calibrate within each class separately. The algorithm will be similar to the group-balanced coverage of Section 4.1, but we must modify it because we do not know the correct class at test time. (In contrast, in Section 4.1, we observed the group information $X_{\text{test},1}$ as an input feature.) See the visualization below.



Turning to the algorithm, given a conformal score function s, stratify the scores on the calibration set by class,

$$s_i^{(k)} = s(X_j, Y_j), \text{ where } Y_j \text{ is the } i\text{th occurrence of class } k.$$

Then, within each class, we calculate the conformal quantile,

$$\hat{q}^{(k)} = \text{Quantile}\left(s_1, ..., s_{n^{(k)}}; \frac{\lceil (n^{(k)}+1)(1-\alpha) \rceil}{n^{(k)}}\right), \text{ where } n^{(k)} \text{ is the number of examples of class } k.$$

Finally, we iterate through our classes and include them in the prediction set based on their quantiles:

$$\mathcal{C}(x) = \left\{ y : s(x, y) \leq \hat{q}^{(y)} \right\}.$$

Notice that in the preceding display, we take a provisional value of the response, y, and then use the conformal threshold $\hat{q}^{(y)}$ to determine if it is included in the prediction set. This choice of $\mathcal{C}$ satisfies (9), as proven by Vovk in [14]; another version can be found in [6].

Proposition 2 (Error control guarantee for class-balanced conformal prediction)**.** *Suppose* $(X_1, Y_1), \ldots, (X_n, Y_n), (X_{test}, Y_{test})$ *are an i.i.d. sample from some distribution. Then the set* $\mathcal{C}$ *defined above satisfies the error control property in* (9)*.*

4.3 Conformal Risk Control

So far, we have used conformal prediction to construct prediction sets that bound the *miscoverage*,

$$\mathbb{P}\Big(Y_{\text{test}} \notin \mathcal{C}(X_{\text{test}})\Big) \leq \alpha. \tag{10}$$

However, for many machine learning problems, the natural notion of error is not miscoverage. Here we show that conformal prediction can also provide guarantees of the form

$$\mathbb{E}\Big[\ell\big(\mathcal{C}(X_{\text{test}}), Y_{\text{test}}\big)\Big] \leq \alpha, \tag{11}$$

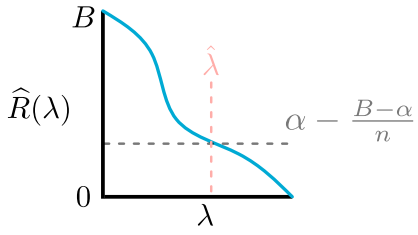
for any bounded *loss function* ℓ that shrinks as $\mathcal{C}$ grows. This is called a *conformal risk control* guarantee. Note that (11) recovers (10) when using the miscoverage loss, $\ell\big(C(X_{\text{test}}), Y_{\text{test}}\big) = \mathbb{1}\left\{Y_{\text{test}} \notin C(X_{\text{test}})\right\}$. However, this algorithm also extends conformal prediction to situations where other loss functions, such as the false negative rate (FNR), are more appropriate.

As an example, consider multilabel classification. Here, the response $Y_i \subseteq \{1, ..., K\}$ a subset of K classes. Given a trained model $f : \mathcal{X} \to [0,1]^K$, we wish to output sets that include a large fraction of the true classes in Y_i. To that end, we post-process the model's raw outputs into the set of classes with sufficiently high scores, $\mathcal{C}_\lambda(x) = \{k : f(X)_k \geq 1 - \lambda\}$. Note that as the threshold λ grows, we include more classes in $\mathcal{C}_\lambda(x)$—it becomes more conservative in that we are less likely to omit true classes. Conformal risk control can be used to find a threshold value $\hat{\lambda}$ that controls the fraction of missed classes. That is, $\hat{\lambda}$ can be chosen so that the expected value of $\ell\big(\mathcal{C}_{\hat{\lambda}}(X_{\text{test}}), Y_{\text{test}}\big) = 1 - |Y_{\text{test}} \cap \mathcal{C}_\lambda(X_{\text{test}})|/|Y_{\text{test}}|$ is guaranteed to fall below a user-specified error rate α. For example, setting $\alpha = 0.1$ ensures that $\mathcal{C}_{\hat{\lambda}}(X_{\text{test}})$ contains 90% of the true classes in Y_{test} on average. We will work through a multilabel classification example in detail in Section 5.1.

Formally, we will consider post-processing the predictions of the model f to create a prediction set $\mathcal{C}_\lambda(\cdot)$. The prediction set has a parameter λ that encodes its level of conservativeness: larger λ values yield more conservative outputs (e.g., larger prediction sets). To measure the quality of the output of $\mathcal{C}_\lambda$, we consider a loss function $\ell(\mathcal{C}_\lambda(x), y) \in (-\infty, B]$ for some $B < \infty$. We require the loss function to be non-increasing as a function of λ. The following algorithm picks $\hat{\lambda}$ so that risk control as in (11) holds:

$$\hat{\lambda} = \inf\left\{\lambda : \widehat{R}(\lambda) \leq \alpha - \frac{B - \alpha}{n}\right\}, \tag{12}$$

where $\widehat{R}(\lambda) = \big(\ell\big(\mathcal{C}_\lambda(X_1), Y_1\big) + \ldots + \ell\big(\mathcal{C}_\lambda(X_n), Y_n\big)\big)/n$ is the empirical risk on the calibration data. Note that this algorithm simply corresponds to tuning based on the empirical risk at a slightly more conservative level than α. For example, if $B = 1$, $\alpha = 0.1$, and we have $n = 1000$ calibration points, then we select $\hat{\lambda}$ to be the value where empirical risk hits level $\hat{\lambda} = 0.0991$ instead of 0.1.



Then the prediction set $\mathcal{C}_{\hat{\lambda}}(X_{\text{test}})$ satisfies (11).

Theorem 2 (Conformal Risk Control [17]). *Suppose $(X_1, Y_1), \ldots, (X_n, Y_n), (X_{test}, Y_{test})$ are an i.i.d. sample from some distribution. Further, suppose ℓ is a monotone function of λ, i.e., one satisfying*

$$\ell\big(\mathcal{C}_{\lambda_1}(x), y\big) \geq \ell\big(\mathcal{C}_{\lambda_2}(x), y\big) \tag{13}$$

for all (x, y) and $\lambda_1 \leq \lambda_2$. Then

$$\mathbb{E}\left[\ell\big(\mathcal{C}_{\hat{\lambda}}(X_{\text{test}}), Y_{\text{test}}\big)\right] \leq \alpha,$$

where $\hat{\lambda}$ is picked as in (12).

Theory and worked examples of conformal risk control are presented in [17]. In Sections 5.1 and 5.2 we show a worked example of conformal risk control applied to tumor segmentation. Furthermore, Appendix A describes a more powerful technique called Learn then Test [18] capable of controlling general risks that do not satisfy (13).

4.4 Outlier Detection

Conformal prediction can also be adapted to handle unsupervised outlier detection. Here, we have access to a clean dataset $X_1, \ldots, X_n$ and wish to detect when test points do not come from the same distribution. As before, we begin with a heuristic model that tries to identify outliers; a larger score means that the model judges the point more likely to be an outlier. We will then use a variant of conformal prediction to calibrate it to have statistical guarantees. In particular, we will guarantee that it does not return too many false positives.

Formally, we will construct a function that labels test points as outliers or inliers, $\mathcal{C} : \mathcal{X} \to \{\text{outlier}, \text{inlier}\}$, such that

$$\mathbb{P}\left(\mathcal{C}(X_{\text{test}}) = \text{outlier}\right) \leq \alpha, \tag{14}$$

where the probability is over X_{test}, a fresh sample from the clean-data distribution. The algorithm for achieving (14) is similar to the usual conformal algorithm. We start with a conformal score $s : \mathcal{X} \to \mathbb{R}$ (note that since we are in the unsupervised setting, the score only depends on the features). Next, we compute the conformal score on the clean data: $s_i = s(X_i)$ for $i = 1, \ldots, n$. Then, we compute the conformal threshold in the usual way:

$$\hat{q} = \text{quantile}\left(s_1, \ldots, s_n; \frac{\lceil (n+1)(1-\alpha) \rceil}{n}\right).$$

Lastly, when we encounter a test point, we declare it to be an outlier if the score exceeds $\hat{q}$:

$$\mathcal{C}(x) = \begin{cases} \text{inlier} & \text{if } s(x) \leq \hat{q} \\ \text{outlier} & \text{if } s(x) > \hat{q} \end{cases}.$$

This construction guarantees error control, as we record next.

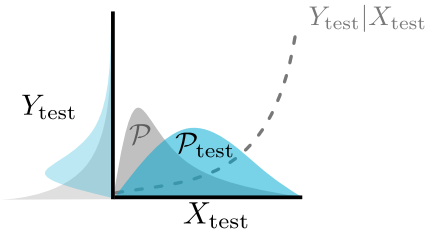
Proposition 3 (Error control guarantee for outlier detection)**.** *Suppose $X_1, \ldots, X_n, X_{test}$ are an i.i.d. sample from some distribution. Then the set $\mathcal{C}$ defined above satisfies the error control property in* (14)*.*

As with standard conformal prediction, the score function is very important for the method to perform well—that is, to be effective at flagging outliers. Here, we wish to choose the score function to effectively distinguish the type of outliers that we expect to see in the test data from the clean data. The general problem of training models to distinguish outliers is sometimes called *anomaly detection*, *novelty detection*, or *one-class classification*, and there are good out-of-the box methods for doing this; see [19] for an overview of outlier detection. Conformal outlier detection can also be seen as a hypothesis testing problem; points that are rejected as outliers have a p-value less than *alpha* for the null hypothesis of exchangeability with the calibration data. This interpretation is closely related to the classical permutation test [20, 21]. See [22–24] for more on this interpretation and other statistical properties of conformal outlier detection.

4.5 Conformal Prediction Under Covariate Shift

All previous conformal methods rely on Theorem 1, which assumes that the incoming test points come from the same distribution as the calibration points. However, past data is not necessarily representative of future data in practice.

One type of distribution shift that conformal prediction can handle is *covariate shift*. Covariate shift refers to the situation where the distribution of X_{test} changes from $\mathcal{P}$ to $\mathcal{P}_{\text{test}}$, but the relationship between X_{test} and Y_{test}, i.e. the distribution of $Y_{\text{test}}|X_{\text{test}}$, stays fixed.



Imagine our calibration features $\{X_i\}_{i=1}^n$ are drawn independently from $\mathcal{P}$ but our test feature X_{test} is drawn from $\mathcal{P}_{\text{test}}$. Then, there has been a covariate shift, and the data are no longer i.i.d. This problem is common in the real world. For example,

- You are trying to predict diseases from MRI scans. You conformalized on a balanced dataset of 50% infants and 50% adults, but in reality, the frequency is 5% infants and 95% adults. Deploying the model in the real world would invalidate coverage; the infants are over-represented in our sample, so diseases present during infancy will be over-predicted. This was a covariate shift in age.
- You are trying to do instance segmentation, i.e., to segment each object in an image from the background. You collected your calibration images in the morning but seek to deploy your system in the afternoon. The amount of sunlight has changed, and more people are eating lunch. This was a covariate shift in the time of day.

To address the covariate shift from $\mathcal{P}$ to $\mathcal{P}_{\text{test}}$, one can form valid prediction sets with *weighted conformal prediction*, first developed in [25].

In weighted conformal prediction, we account for covariate shift by upweighting conformal scores from calibration points that would be more likely under the new distribution. We will be using the *likelihood ratio*

$$w(x) = \frac{\mathrm{d}\mathcal{P}_{\text{test}}(x)}{\mathrm{d}\mathcal{P}(x)};$$

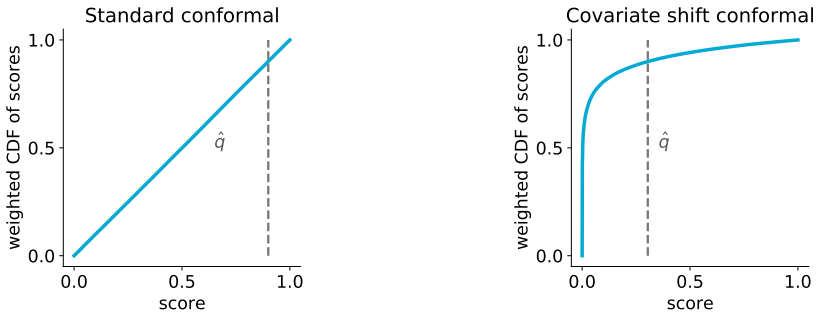
usually this is just the ratio of the new PDF to the old PDF at the point x. Now we define our weights,

$$p_i^w(x) = \frac{w(X_i)}{\sum_{j=1}^{n} w(X_j) + w(x)} \quad \text{and} \quad p_{\text{test}}^w(x) = \frac{w(x)}{\sum_{j=1}^{n} w(X_j) + w(x)}.$$

Intuitively, the weight $p_i^w(x)$ is large when X_i is likely under the new distribution, and $p_{\text{test}}^w(x)$ is large when the input x is likely under the new distribution. We can then express our conformal quantile as the $1-\alpha$ quantile of a reweighted distribution,

$$\hat{q}(x) = \inf\left\{ s_j : \sum_{i=1}^{j} p_i^w(x) \mathbb{1}\left\{s_i \le s_j\right\} \ge 1-\alpha \right\},$$

where above for notational convenience we assume that the scores are ordered from smallest to largest a-priori. The choice of quantile is the key step in this algorithm, so we pause to parse it. First of all, notice that the quantile is now a function of an input x, although the dependence is only minor. Choosing $p_i^w(x) = p_{\text{test}}^w(x) = \frac{1}{n+1}$ gives the familiar case of conformal prediction—all points are equally weighted, so we end up choosing the $\lceil (n+1)(1-\alpha) \rceil$th-smallest score as our quantile. When there is covariate shift, we instead re-weight the calibration points with non-equal weights to match the test distribution. If the covariate shift makes easier values of x more likely, it makes our quantile smaller. This happens because the covariate shift puts more weight on small scores—see the diagram below. Of course, the opposite holds the covariate shift upweights difficult values of x: so the covariate-shift-adjusted quantile grows.



With this quantile function in hand, we form our prediction set in the standard way,

$$\mathcal{C}(x) = \{y : s(x,y) \le \hat{q}(x)\}.$$

By accounting for the covariate shift in our choice of $\hat{q}$, we were able to make our calibration data look exchangeable with the test point, achieving the following guarantee.

Theorem 3 (Conformal prediction under covariate shift [25])**.** *Suppose* $(X_1, Y_1), ..., (X_n, Y_n)$ *are drawn i.i.d. from* $\mathcal{P} \times \mathcal{P}_{Y|X}$ *and that* $(X_{\text{test}}, Y_{\text{test}})$ *is drawn independently from* $\mathcal{P}_{\text{test}} \times \mathcal{P}_{Y|X}$*. Then the choice of* $\mathcal{C}$ *above satisfies*

$$\mathbb{P}\left(Y_{\text{test}} \in \mathcal{C}(X_{\text{test}})\right) \ge 1-\alpha.$$

Conformal prediction under various distribution shifts is an active and important area of research with many open challenges. This algorithm addresses a somewhat restricted case—that of a known covariate shift—but is nonetheless quite practical.

4.6 Conformal Prediction Under Distribution Drift

Another common form of distribution shift is *distribution drift*: slowly varying changes in the data distribution. For example, when collecting time-series data, the data distribution may change—furthermore, it may

change in a way that is unknown or difficult to estimate. Here, one can imagine using weights that give more weight to recent conformal scores. The following theory provides some justification for such *weighted conformal* procedures; in particular, they always satisfy marginal coverage, and are exact when the magnitude of the distribution shift is known.

More formally, suppose the calibration data $\{(X_i, Y_i)\}_{i=1}^n$ are drawn independently from different distributions $\{\mathcal{P}_i\}_{i=1}^n$ and the test point $(X_{\text{test}}, Y_{\text{test}})$ is drawn from $\mathcal{P}_{\text{test}}$. Given some weight schedule $w_1, ..., w_n$, $w_i \in [0, 1]$, we will consider the calculation of weighted quantiles using the calibration data:

$$\hat{q} = \inf\left\{q : \sum_{i=1}^{n} \tilde{w}_i \mathbb{1}\left\{s_i \leq q\right\} \geq 1 - \alpha\right\},$$

where the $\tilde{w}_i$ are normalized weights,

$$\tilde{w}_i = \frac{w_i}{w_1 + \ldots + w_n + 1}.$$

Then we can construct prediction sets in the usual way,

$$\mathcal{C}(x) = \left\{y : s(x, y) \leq \hat{q}\right\}.$$

We now state a theorem showing that when the distribution is shifting, it is a good idea to apply a discount factor to old samples. In particular, let $\epsilon_i = \mathrm{d_{TV}}\big((X_i, Y_i), (X_{\text{test}}, Y_{\text{test}})\big)$ be the TV distance between the ith data point and the test data point. The TV distance is a measure of how much the distribution has shifted—a large ϵ_i (close to 1) means the ith data point is not representative of the new test point. The result states that if w discounts those points with large shifts, the coverage remains close to $1 - \alpha$.

Theorem 4 (Conformal prediction under distribution drift [26])**.** *Suppose* $\epsilon_i = \mathrm{d_{TV}}\big((X_i, Y_i), (X_{\text{test}}, Y_{\text{test}})\big)$. *Then the choice of* $\mathcal{C}$ *above satisfies*

$$\mathbb{P}\left(Y_{\text{test}} \in \mathcal{C}(X_{\text{test}})\right) \geq 1 - \alpha - 2\sum_{i=1}^{n} \tilde{w}_i \epsilon_i.$$

When either factor in the product $\tilde{w}_i \epsilon_i$ is small, that means that the ith data point doesn't result in loss of coverage. In other words, if there isn't much distribution shift, we can place a high weight on that data point without much penalty, and vice versa. Setting $\epsilon_i = 0$ above, we can also see that when there is no distribution shift, there is no loss in coverage regardless of what choice of weights is used—this fact had been observed previously in [25, 27].

The ϵ_i are never known exactly in advance—we only have some heuristic sense of their size. In practice, for time-series problems, it often suffices to pick either a rolling window of size K or a smooth decay using some domain knowledge about the speed of the drift:

$$w_i^{\text{fixed}} = \mathbb{1}\left\{i \geq n - K\right\} \qquad \text{or} \qquad w_i^{\text{decay}} = 0.99^{n-i+1}.$$

We give a worked example of this procedure for a distribution shifting over time in Section 5.3.

As a final point on this algorithm, we note that there is some cost to using this or any other weighted conformal procedure. In particular, the weights determine the *effective sample size* of the distribution:

$$n^{\text{eff}}(w_1, \ldots, w_n) = \frac{w_1 + \ldots + w_n}{w_1^2 + \ldots + w_n^2}.$$

This is quite important in practice, since the variance of the weighted conformal procedure can explode when n^{eff} is small; as in Section 3, the variance of coverage scales as $1/\sqrt{n^{\text{eff}}}$, which can be large if too many of the w_i are small. To see more of the theory of weighted conformal prediction under distribution drift, see [26].

5 Worked Examples

We now show several worked examples of the techniques described in Section 4. For each example, we provide Jupyter notebooks that allow the results to be conveniently replicated and extended.

5.1 Multilabel Classification

Figure 13: Examples of false negative rate control in multilabel classification *on the MS COCO dataset with* $\alpha = 0.1$. *False negatives are red, false positives are blue, and true positives are black.*

In the multilabel classification setting, we receive an image and predict which of K objects are in an image. We have a pretrained model $\hat{f}$ that outputs estimated probabilities for each of the K classes. We wish to report on the possible classes contained in the image, returning most of the true labels. To this end, we will threshold the model's outputs to get the subset of K classes that the model thinks is most likely, $\mathcal{C}_\lambda(x) = \{y : \hat{f}(x) \geq \lambda\}$, which we call the prediction. We will use conformal risk control (Section 4.3) to pick the threshold value λ certifying a low *false negative rate* (FNR), i.e., to guarantee the average fraction of ground truth classes that the model missed is less than α.

More formally, our calibration set $\{(X_i, Y_i)\}_{i=1}^n$ contains exchangeable images X_i and sets of classes $Y_i \subseteq \{1, ..., K\}$. With the notation of Section 4.3, we set our loss function to be $\ell_{\text{FNR}}(\mathcal{C}_\lambda(x), y) = 1 - |\mathcal{C}_\lambda(x) \cap y| / |y|$. Then, picking $\hat{\lambda}$ as in 12 yields a bound on the false negative rate,

$$\mathbb{E}\left[\ell_{\text{FNR}}\left(\mathcal{C}_{\hat{\lambda}}(X_{\text{test}}), Y_{\text{test}}\right)\right] \leq \alpha.$$

Figure 13 gives results and code for FNR control on the Microsoft Common Objects in Context dataset [28].

5.2 Tumor Segmentation

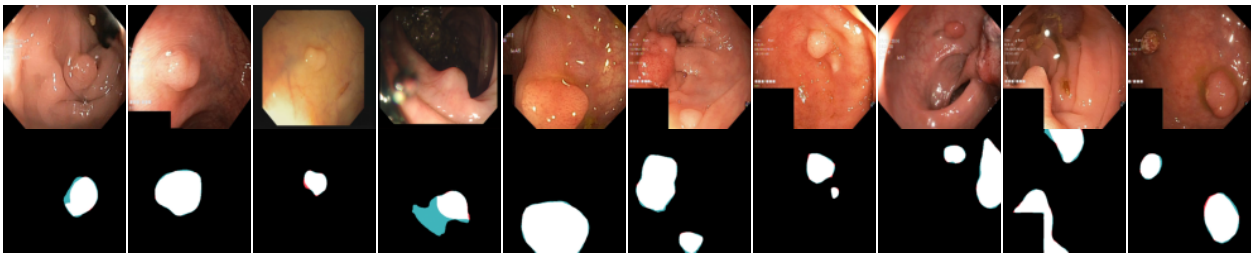


Figure 14: Examples of false negative rate control in tumor segmentation *with* $\alpha = 0.1$. *False negatives are red, false positives are blue, and true positives are black.*

In the tumor segmentation setting, we receive an $M \times N \times 3$ image of a tumor and predict an $M \times N$ binary mask, where '1' indicates a tumor pixel. We start with a pretrained segmentation model $\hat{f}$ that outputs an $M \times N$ grid of the estimated probabilities that each pixel is a tumor pixel. We will threshold the model's outputs to get our predicted binary mask, $\mathcal{C}_\lambda(x) = \{(i, j) : \hat{f}(x)_{(i,j)} \geq \lambda\}$, which we call the

prediction. We will use conformal risk control (Section 4.3) to pick the threshold value λ certifying a low FNR, i.e., guaranteeing the average fraction of tumor pixels missed is less than α.

More formally, our calibration set $\{(X_i, Y_i)\}_{i=1}^n$ contains exchangeable images X_i and sets of tumor pixels $Y_i \subseteq \{1, \ldots, M\} \times \{1, \ldots, N\}$. As in the previous example, we let the loss be the false negative proportion, ℓ_{FNR}. Then, picking $\hat{\lambda}$ as in 12 yields the bound on the FNR in 5.1. Figure 14 gives results and code on a dataset of gut polyps.

5.3 Weather Prediction with Time-Series Distribution Shift

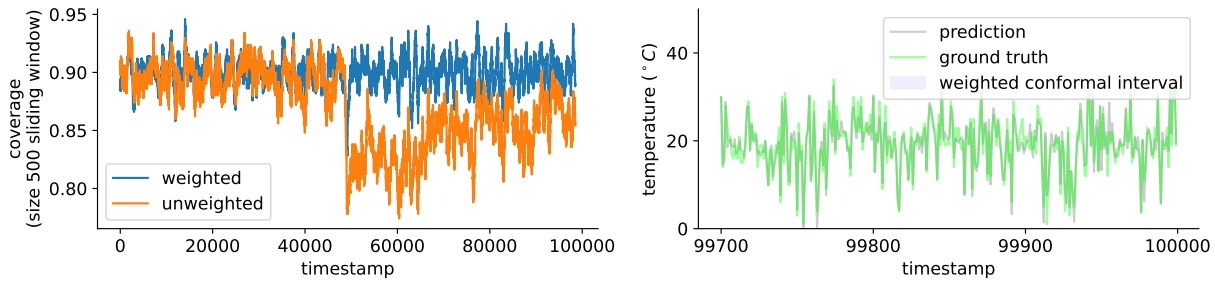


Figure 15: Conformal prediction for time-series temperature estimation *with $\alpha = 0.1$. On the left is a plot of coverage over time; 'weighted' denotes the procedure in Section 5.3 while 'unweighted' denotes the procedure that simply computes the conformal quantile on all conformal scores seen so far. Note that we compute coverage using a sliding window of 500 points, which explains some of the variability in the coverage. Running the notebook with a trailing average of 5000 points reveals that the unweighted version systematically undercovers before the change-point as well. On the right is a plot showing the intervals resulting from the weighted procedure.*

In this example we seek to predict the temperature of different locations on Earth given covariates such as the latitude, longitude, altitude, atmospheric pressure, and so on. We will make these predictions serially in time. Dependencies between adjacent data points induced by local and global weather changes violate the standard exchangeability assumption, so we will need to apply the method from Section 4.6.

In this setting, we have a time series $\left\{(X_t, Y_t)\right\}_{t=1}^T$, where the X_t are tabular covariates and the $Y_t \in \mathbb{R}$ are temperatures in degrees Celsius. Note that these data points are not exchangeable or i.i.d.; adjacent data points will be correlated. We start with a pretrained model $\hat{f}$ taking features and predicting temperature and an uncertainty model $\hat{u}$ takes features and outputs a scalar notion of uncertainty. Following Section 2.3, we compute the conformal scores

$$s_t = \frac{\left|Y_t - \hat{f}(X_t)\right|}{\hat{u}(X_t)}.$$

Since we observe the data points sequentially, we also observe the scores sequentially, and we will need to pick a different conformal quantile for each incoming data point. More formally, consider the task of predicting the temperature at time $t \leq T$. We use the weighted conformal technique in Section 5.3 with the fixed K-sized window $w_{t'} = \mathbb{1}\left\{t' \geq t - K\right\}$ for all $t' < t$. This yields the quantiles

$$\hat{q}_t = \inf\left\{q : \frac{1}{\min(K, t'-1)+1} \sum_{t'=1}^{t-1} s_{t'} \mathbb{1}\left\{t' \geq t - K\right\} \geq 1 - \alpha\right\}.$$

With these adjusted quantiles in hand, we form prediction sets at each time step in the usual way,

$$\mathcal{C}(X_t) = \left[\hat{f}(X_t) - \hat{q}_t \hat{u}(X_t)\ ,\ \hat{f}(X_t) + \hat{q}_t \hat{u}(X_t)\right].$$

We run this procedure on the Yandex Weather Prediction dataset. This dataset is part of the Shifts Project [29], which also provides an ensemble of 10 pretrained CatBoost [30] models for making the temperature predictions. We take the average prediction of these models as our base model $\hat{f}$. Each of the models has its own internal variance; we take the average of these variances as our uncertainty scalar $\hat{u}$. The dataset includes an in-distribution split of fresh data from the same time frame that the base model was trained and an out-of-distribution split consisting of time windows the model has never seen. We concatenate these datasets in time, leading to a large change point in the score distribution. Results in Figure 15 show that the weighted method works better than a naive unweighted conformal baseline, achieving the desired coverage in steady-state and recovering quickly from the change point. There is no hope of measuring the TV distance between adjacent data points in order to apply Theorem 4, so we cannot get a formal coverage bound. Nonetheless, the procedure is useful with this simple fixed window of weights, which we chose with only a heuristic understanding of the distribution drift speed. It is worth noting that conformal prediction for time-series applications is a particularly active area of research currently, and the method we have presented is not clearly the best. See [31–33] and [34] for two differing perspectives.

5.4 Toxic Online Comment Identification via Outlier Detection

True Negative:	ХОТЕЛОСЬ БЫ УЗНАТЬ ,КАК В ЭТОМ ГОДУ БУДУ ТПРОИЗВОДИТЬ СПУСК ВОДЫ И БУДУТ ЛИ ЗАТОПЛЕНИЯ I would like to know how water will be drained this year and whether there will be flooding.
False Negative:	votre petit discours provocateur à souhait n a pas plus d effet sur moi qu un pet de lapin sur une toile cirée your provocative little speech has no more effect on me than a rabbit fart on an oilcloth
False Positive:	Bah, sono completamente d accordo con te, e questa dovrebbe essere un enciclopedia libera???? veramente ridicolo! Bah, I completely agree with you, and this is supposed to be a free encyclopedia ????
True Positive:	CALLATE BOT DE M****A SIN VIDA SOCIAL SHUT UP, BOT OF S**T WITHOUT A SOCIAL LIFE

Figure 16: Examples of toxic online comment identification with type-1 error control *at level* $\alpha = 0.1$ *on the Jigsaw Multilingual Toxic Comment Classification dataset.*

We provide a type-1 error guarantee on a model that flags toxic online comments, such as threats, obscenity, insults, and identity-based hate. Suppose we are given n non-toxic text samples $X_1, ..., X_n$ and asked whether a new text sample X_{test} is toxic. We also have a pre-trained toxicity prediction model $\hat{f}(x) \in [0, 1]$, where values closer to 1 indicate a higher level of toxicity. The goal is to flag as many toxic comments as possible while not flagging more than α proportion of non-toxic comments.

The outlier detection procedure in Section 4.4 applies immediately. First, we run the model on each calibration point, yielding conformal scores $s_i = \hat{f}(X_i)$. Taking the toxicity threshold $\hat{q}$ to be the $\lceil (n+1)(1-\alpha) \rceil$-smallest of the s_i, we construct the function

$$\mathcal{C}(x) = \begin{cases} \text{inlier} & \hat{f}(x) \leq \hat{q} \\ \text{outlier} & \hat{f}(x) > \hat{q}. \end{cases}$$

This gives the guarantee in Proposition 3—no more than α fraction of future nontoxic text will be classified as toxic.

Figure 16 shows results of this procedure using the Unitary Detoxify BERT-based model [35, 36] on the Jigsaw Multilingual Toxic Comment Classification dataset from the WILDS benchmark [37]. It is composed of comments from the talk channels of Wikipedia pages. With a type-1 error of $\alpha = 10\%$, the system correctly flags 70% of all toxic comments.

5.5 Selective Classification

In many situations, we only want to show a model's predictions when it is confident. For example, we may only want to make medical diagnoses when the model will be 95% accurate, and otherwise to say "I don't

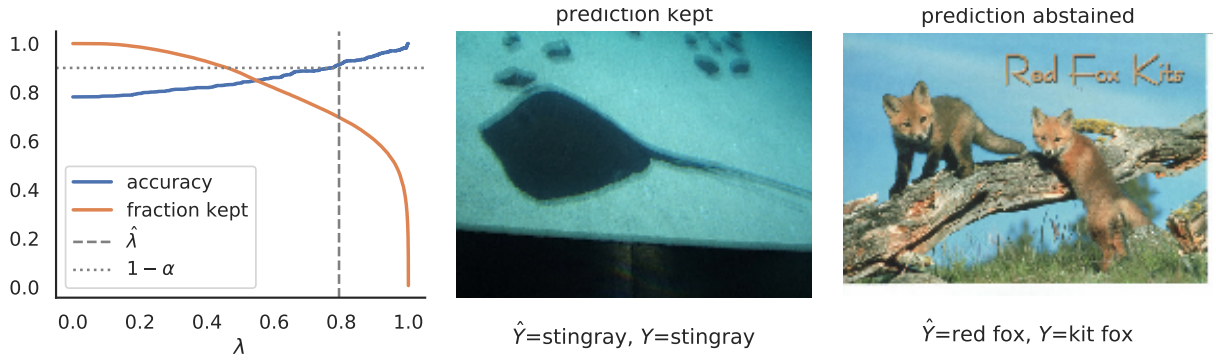


Figure 17: Results using selective classification on Imagenet *with* $\alpha = 0.1$.

know." We next demonstrate a system that strategically abstains in order to achieve a higher accuracy than the base model in the problem of image classification.

More formally, given image-class pairs $\{(X_i, Y_i)\}_{i=1}^n$ and an image classifier $\hat{f}$, we seek to ensure

$$\mathbb{P}\left(Y_{\text{test}} = \widehat{Y}(X_{\text{test}}) \,\middle|\, \widehat{P}(X_{\text{test}}) \geq \hat{\lambda}\right) \geq 1 - \alpha, \tag{15}$$

where $\widehat{Y}(x) = \arg\max_y \hat{f}(x)_y$, $\widehat{P}(X_{\text{test}}) = \max_y \hat{f}(x)_y$, and $\hat{\lambda}$ is a threshold chosen using the calibration data. This is called a *selective accuracy* guarantee, because the accuracy is only computed over a subset of high-confidence predictions. This quantity cannot be controlled with techniques we've seen so far, since we are not guaranteed that model accuracy is monotone in the cutoff λ. Nonetheless, it can be handled with Learn then Test—a framework for controlling arbitrary risks (see Appendix A). We show only the special case of controlling selective classification accuracy here.

We pick the threshold using based on the empirical estimate of selective accuracy on the calibration set,

$$\widehat{R}(\lambda) = \frac{1}{n(\lambda)} \sum_{i=1}^{n} \mathbb{1}\left\{Y_i \neq \widehat{Y}(X_i) \text{ and } \widehat{P}(X_i) \geq \lambda\right\}, \text{ where } n(\lambda) = \sum_{i=1}^{n} \mathbb{1}\left\{\widehat{P}(X_i) \geq \lambda\right\}.$$

Since this function is not monotone in λ, we will choose $\hat{\lambda}$ differently than in Section 4.3. In particular, we will scan across values of λ looking at a conservative upper bound for the true risk (i.e., the top end of a confidence interval for the selective misclassification rate). Realizing that $\widehat{R}(\lambda)$ is a Binomial random variable with $n(\lambda)$ trials, we upper-bound the misclassification error as

$$\widehat{R}^+(\lambda) = \sup\left\{r \;:\; \text{BinomCDF}(\widehat{R}(\lambda);\, n(\lambda), r) \geq \delta\right\}$$

for some user-specified failure rate $\delta \in [0, 1]$. Then, scan the upper bound until the last time the bound exceeds α,

$$\hat{\lambda} = \inf\left\{\lambda : \widehat{R}^+(\lambda') \leq \alpha \text{ for all } \lambda' \geq \lambda\right\}.$$

Deploying the threshold $\hat{\lambda}$ will satisfy (15) with high probability.

Proposition 4. *Assume the* $\{(X_i, Y_i)\}_{i=1}^n$ *and* $(X_{\text{test}}, Y_{\text{test}})$ *are i.i.d. and* $\hat{\lambda}$ *is chosen as above. Then* (15) *is satisfied with probability* $1 - \delta$.

See results on Imagenet at level $\alpha = 0.1$ in Figure 17. For a deeper dive into this procedure and techniques for controlling other non-monotone risks, see Appendix A.

6 Full conformal prediction

Up to this point, we have only considered *split conformal prediction*, otherwise known as inductive conformal prediction. This version of conformal prediction is computationally attractive, since it only requires fitting the model one time, but it sacrifices statistical efficiency because it requires splitting the data into training and calibration datasets. Next, we consider *full conformal prediction*, or transductive conformal prediction, which avoids data splitting at the cost of many more model fits. Historically, full conformal prediction was developed first, and then split conformal prediction was later recognized as an important special case. Next, we describe full conformal prediction. This discussion is motivated from three points of view. First, full conformal prediction is an elegant, historically important idea in our field. Second, the exposition will reveal a complimentary interpretation of conformal prediction as a hypothesis test. Lastly, full conformal prediction is a useful algorithm when statistical efficiency is of paramount importance.

6.1 Full Conformal Prediction

This topic requires expanded notation. Let $(X_1, Y_1), \ldots, (X_{n+1}, Y_{n+1})$ be $n+1$ exchangeable data points. As before, the user sees $(X_1, Y_1), \ldots, (X_n, Y_n)$ and X_{n+1}, and wishes to make a prediction set that contains Y_{n+1}. But unlike split conformal prediction, we allow the model to train on all the data points, so there is no separate calibration dataset.

The core idea of full conformal prediction is as follows. We know that the true label, Y_{n+1}, lives somewhere in $\mathcal{Y}$ — so if we loop over all possible $y \in \mathcal{Y}$, then we will eventually hit the data point (X_{n+1}, Y_{n+1}), which is exchangeable with the first n data points. Full conformal prediction is so-named because it directly computes this loop. For each $y \in \mathcal{Y}$, we fit a new model $\hat{f}^y$ to the augmented dataset $(X_1, Y_1), \ldots, (X_{n+1}, y)$. Importantly, the model fitting for $\hat{f}$ must be invariant to permutations of the data. Then, we compute a score function $s_i^y = s(X_i, Y_i, \hat{f}^y)$ for i = 1,...,n and $s_{n+1}^y = s(X_{n+1}, y, \hat{f}^y)$. This score function is exactly the same as those from Section 2, except that the model $\hat{f}^y$ is now given as an argument because it is no longer fixed. Then, we calculate the conformal quantile,

$$\hat{q}^y = \text{Quantile}\left(s_1^y, \ldots, s_n^y; \frac{\lceil (n+1)(1-\alpha) \rceil}{n}\right).$$

Then, we collect all values of y that are sufficiently consistent with the previous data $(X_1, Y_1), \ldots, (X_n, Y_n)$ are collected into a confidence set for the unknown value of Y_{n+1}:

$$\mathcal{C}(X_{\text{test}}) = \{y : s_{n+1}^y \leq \hat{q}^y\}. \tag{16}$$

This prediction set has the same validity guarantee as before:

Theorem 5 (Full conformal coverage guarantee [1])**.** *Suppose $(X_1, Y_1), ..., (X_{n+1}, Y_{n+1})$ are drawn i.i.d. from $\mathcal{P}$, and that $\hat{f}$ is a symmetric algorithm. Then the choice of $\mathcal{C}$ above satisfies*

$$\mathbb{P}\left(Y_{n+1} \in \mathcal{C}(X_{n+1})\right) \geq 1 - \alpha.$$

More generally, the above holds for exchangeable random variables $(X_1, Y_1), ..., (X_{n+1}, Y_{n+1})$; the proof of Theorem 5 critically relies on the fact that the score $s_{n+1}^{Y_{n+1}}$ is exchangeable with $s_1^{Y_{n+1}}, \ldots, s_n^{Y_{n+1}}$. We defer the proof to [1], and note that upper bound in (1) also holds when the score function is continuous.

What about computation? In principle, to compute (16), we must iterate over all $y \in \mathcal{Y}$, which leads to a substantial computational burden. (When $\mathcal{Y}$ is continuous, we would typically first discretize the space and then check each element in a finite set.) For example, if $|Y| = K$, then computing (16) requires $(n+1) \cdot K$ model fits. For some specific score functions, the set in (16) can actually be computed exactly even for continuous Y, and we refer the reader to [1] and [38] for a summary of such cases and [39, 40] for recent developments. Still, full conformal prediction is generally computationally costly.

Lastly, we give a statistical interpretation for the prediction set in (16). The condition

$$s_{n+1}^y \leq \hat{q}^y$$

is equivalent to the acceptance condition of a certain permutation test. To see this, consider a level α permutation test for the exchangeability of $s_1^y, \ldots, s_n^y$ and the test score s_{n+1}^y, rejecting when the score function is large. The values of y such that the test does not reject are exactly those in (16). In words, the confidence set is all values of y such that the hypothetical data point is consistent with the other data, as judged by this permutation test. We again refer the reader to [1] for more on this viewpoint on conformal prediction.

6.2 Cross-Conformal Prediction, CV+, and Jackknife+

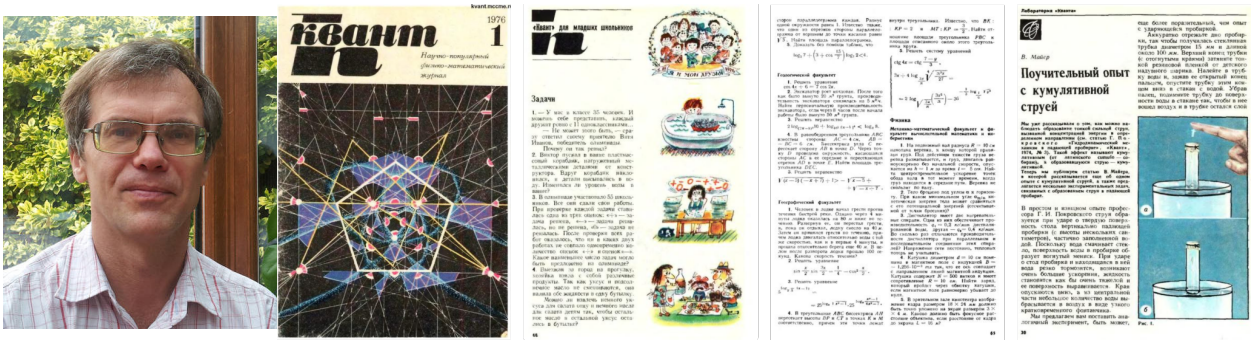
Split conformal prediction requires only one model fitting step, but sacrifices statistical efficiency. On the other hand, full conformal prediction requires a very large number of model fitting steps, but has high statistical efficiency. These are not the only two achievable points on the spectrum—there are techniques that fall in between, trading off statistical efficiency and computational efficiency differently. In particular, cross-conformal prediction [41] and CV+/Jackknife+ [42] both use a small number of model fits, but still use all data for both model fitting and calibration. We refer the reader to those works for a precise description of the algorithms and corresponding statistical guarantees.

7 Historical Notes on Conformal Prediction

We hope the reader has enjoyed reading the technical content in our gentle introduction. As a dénouement, we now pay homage to the history of conformal prediction. Specifically, we will trace the history of techniques related to conformal prediction that are distribution-free, i.e., (1) agnostic to the model, (2) agnostic to the data distribution, and (3) valid in finite samples. There are other lines of work in statistics with equal claim to the term "distribution-free" especially when it is interpreted asymptotically, such as permutation tests [43], quantile regression [9], rank tests [44–46], and even the bootstrap [47, 48]—the following is not a history of those topics. Rather, we focus on the progenitors and progeny of conformal prediction.

Origins

The story of conformal prediction begins sixty-three kilometers north of the seventh-largest city in Ukraine, in the mining town of Chervonohrad in the Oblast of Lviv, where Vladimir Vovk spent his childhood. Vladimir's parents were both medical professionals, of Ukrainian descent, although the Lviv region changed hands many times over the years. During his early education, Vovk recalls having very few exams, with grades mostly based on oral answers. He did well in school and eventually took first place in the Mathematics Olympiad in Ukraine; he also got a Gold Medal, meaning he was one of the top graduating secondary school students. Perhaps because he was precocious, his math teacher would occupy him in class by giving him copies of a magazine formerly edited by Isaak Kikoin and Andrey Kolmogorov, Kvant, where he learned about physics, mathematics, and engineering—see Figure 18. Vladimir originally attended the Moscow Second Medical Institute (now called the Russian National Research Medical University) studying Biological Cybernetics, but eventually became disillusioned with the program, which had too much of a medical emphasis and imposed requirements to take classes like anatomy and physiology (there were "too many bones with strange Latin names"). Therefore, he sat the entrance exams a second time and restarted school at the Mekh-Mat (faculty of mechanics and mathematics) in Moscow State University. In his third year there, he became the student of Andrey Kolmogorov. This was when the seeds of conformal prediction were first laid. Today, Vladimir Vovk is widely recognized for being the co-inventor of conformal prediction, along with collaborators Alexander Gammerman, Vladimir Vapnik, and others, whose contributions we will soon discuss. First, we will relay some of the historical roots of conformal prediction, along with some oral history related by Vovk that may be forgotten if never written.



Vladimir Vovk

Figure 18: Pages from the 1976 edition of Kvant magazine.

Kolmogorov and Vovk met approximately once a week during his three remaining years as an undergraduate at MSU. At that time, Kolmogorov took an interest in Vovk, and encouraged him to work on difficult mathematical problems. Ultimately, Vovk settled on studying a topic of interest to Kolmogorov: algorithmically random sequences, then known as *collectives*, and which were modified into *Bernoulli sequences* by Kolmogorov.

Work on collectives began at the turn of the 20th century, with Gustav Fechner's *Kollectivmasslehre* [49], and was developed significantly by von Mises [50], Abraham Wald [51], Alonzo Church [52], and so on. A long debate ensued among these statisticians as to whether von Mises' axioms formed a valid foundation for probability, with Jean Ville being a notable opponent [53]. Although the theory of von Mises' collectives is somewhat defunct, the mathematical ideas generated during this time continue to have a broad impact on statistics, as we will see. More careful historical reviews of the original debate on collectives exist elsewhere [52, 54–56]. We focus on its connection to the development of conformal prediction.

Kolmogorov's interest in *Bernoulli sequences* continued into the 1970s and 1980s, when Vovk was his student. Vovk recalls that, on the way to the train station, Kolmogorov told him (not in these exact words),

> "Look around you; you do not only see infinite sequences. There are finite sequences."

Feeling that the finite case was practically important, Kolmogorov extended the idea of collectives via Bernoulli sequences.

Definition 1 (Bernoulli sequence, informal)**.** *A deterministic binary sequence of length n with k 1s is Bernoulli if it is a "random" element of the set of all $\binom{n}{k}$ sequences of the same length and with the same number of 1s. "Random" is defined as having a Kolmogorov complexity close to the maximum,* $\log \binom{n}{k}$.

As is typical in the study of random sequences, the underlying object itself is not a sequence of random variables. Rather, Kolmogorov quantified the "typicality" of a sequence via Kolmogorov complexity: he asked how long a program we would need to write in order to distinguish it from other sequences in the same space [57–59]. Vovk's first work on random sequences modified Kolmogorov's [60] definition to better reflect the randomness in an event like a coin toss. Vovk discusses the history of Bernoulli sequences, including the important work done by Martin-Löf and Levin, in the Appendix of [61]. Learning the theory of Bernoulli sequences brought Vovk closer to understanding finite-sample exchangeability and its role in prediction problems.

We will make a last note about the contributions of the early probabilists before moving to the modern day. The concept of a nonconformity score came from the idea of (local) *randomness deficiency*. Consider the sequence

$$0000000000000000000000000000000000000000000000000000000000000000000000000001.$$

With a computer, we could write a very short program to identify the '1' in the sequence, since it is atypical — it has a *large* randomness deficiency. But to identify any particular '0' in the sequence, we must specify its

location, because it is so typical — it has a *small* randomness deficiency. A heuristic understanding suffices here, and we defer the formal definition of randomness deficiency to [62], avoiding the notation of Turing machines and Kolmogorov complexity. When randomness deficiency is large, a point is atypical, just like the scores we discussed in Section 2. These ideas, along with the existing statistical literature on tolerance intervals [63–66] and works related to de Finetti's theorems on exchangeability [67–72] formed the seedcorn for conformal prediction: the rough notion of collectives eventually became exchangeability, and the idea of randomness deficiency eventually became nonconformity. Furthermore, the early literature on tolerance intervals was quite close mathematically to conformal prediction—indeed, the fact that order statistics of a uniform distribution are Beta distributed was known at the time, and this was used to form prediction regions in high probability, much like [14]; more on this connection is available in Edgar Dobriban's lecture notes [73].

Enter Conformal Prediction

The framework we now call conformal prediction was hatched by Vladimir Vovk, Alexander Gammerman, Craig Saunders, and Vladimir Vapnik in the years 1996-1999, first using e-values [74] and then with p-values [5, 75]. For decades, Vovk and collaborators developed the theory and applications of conformal prediction. Key moments include:

- the 2002 proof that in online conformal prediction, the probability of error is independent across time-steps [76];
- the 2002 development, along with Harris Papadopoulos and Kostas Proedrou, of split-conformal predictors [2];
- Glenn Shafer coins the term "conformal predictor" on December 1, 2003 while writing *Algorithmic Learning in a Random World* with Vovk [1].
- the 2003 development of Venn Predictors [77] (Vovk says this idea came to him on a bus in Germany during the Dagstuhl seminar "Kolmogorov Complexity & Applications");
- the 2012 founding of the Symposium on Conformal and Probabilistic Prediction and its Applications (COPA), hosted in Greece by Harris Papadopoulos and colleagues;
- the 2012 creation of cross-conformal predictors [41] and Venn-Abers predictors [78];
- The 2017 invention of conformal predictive distributions [79].

Algorithmic Learning in a Random World [1], by Vovk, Gammerman, and Glenn Shafer, contains further perspective on the history described above in the bibliography of Chapter 2 and the main text of Chapter 10. Also, the book's website links to several dozen technical reports on conformal prediction and related topics. We now help the reader understand some of these key developments.

Conformal prediction was recently popularized in the United States by the pioneering work of Jing Lei, Larry Wasserman, and colleagues [3, 80–83]. Vovk himself remembers Wasserman's involvement as a landmark moment in the history of the field. In particular, their general framework for distribution-free predictive inference in regression [83] has been a seminal work. They have also, in the special cases of kernel density estimation and kernel regression, created efficient approximations to full conformal prediction [3, 84]. Jing Lei also created a fast and exact conformalization of the Lasso and elastic net procedures [85]. Another equally important contribution of theirs was to introduce conformal prediction to thousands of researchers, including the authors of this paper, and also Rina Barber, Emmanuel Candès, Aaditya Ramdas, Ryan Tibshirani who themselves have made recent fundamental contributions. Some of these we have already touched upon in Section 2, such as adaptive prediction sets, conformalized quantile regression, covariate-shift conformal, and the idea of conformal prediction as indexing nested sets [86].

This group also did fundamental work circumscribing the conditions under which distribution-free conditional guarantees can exist [87], building on previous works by Vovk, Lei, and Wasserman that showed for an

arbitrary continuous distribution, conditional coverage is impossible [3, 14, 83]. More fine-grained analysis of this fact has also recently been done in [88], showing that vanishing-width intervals are achievable if and only if the effective support size of the distribution of X_{test} is smaller than the square of the sample size.

Current Trends

We now discuss recent work in conformal prediction and distribution-free uncertainty quantification more generally, providing pointers to topics we did not discuss in earlier sections. Many of the papers we cite here would be great starting points for novel research on distribution-free methods.

Many recent papers have focused on designing conformal procedures to have good practical performance according to specific desiderata like small set sizes [6], coverage that is approximately balanced across regions of feature space [4, 7, 15, 27, 87, 89], and errors balanced across classes [6, 23, 90, 91]. This usually involves adjusting the conformal score; we gave many examples of such adjustments in Section 2. Good conformal scores can also be trained with data to optimize more complicated desiderata [92].

Many statistical extensions to conformal prediction have also emerged. Such extensions include the ideas of risk control [4, 18] and covariate shift [25] that we previously discussed. One important and continual area of work is distribution shift, where our test point has a different distribution from our calibration data. For example, [93] builds a conformal procedure robust to shifts of known f-divergence in the score function, and adaptive conformal prediction [31] forms prediction sets in a data stream where the distribution varies over time in an unknown fashion by constantly re-estimating the conformal quantile. A weighted version of conformal prediction pioneered by [26] provides tools for addressing non-exchangeable data, most notably slowly changing time-series. This same work develops techniques for applying full conformal prediction to asymmetric algorithms. Beyond distribution shift, recent statistical extensions also address topics such as creating reliable conformal prediction intervals for counterfactuals and individual treatment effects [94–96], covariate-dependent lower bounds on survival times [97], prediction sets that preserve the privacy of the calibration data [98], handling dependent data [99–101], and achieving 'multivalid' coverage that is conditionally valid with respect to several possibly overlapping groups [102, 103].

Furthermore, prediction sets are not the only important form of distribution-free uncertainty quantification. One alternative form is a *conformal predictive distribution*, which outputs a probability distribution over the response space $\mathcal{Y}$ in a regression problem [79]. Recent work also addresses the issue of calibrating a scalar notion of uncertainty to have probabilistic meaning via histogram binning [104, 105]—this is like a rigorous version of Platt scaling or isotonic regression. The tools from conformal prediction can also be used to identify times when the distribution of data has changed by examining the score function's behavior on new data points. For example, [24] performs outlier detection using conformal prediction, [61, 106] detect change points in time-series data, [107] tests for covariate shift between two datasets, and [108] tracks the risk of a predictor on a data-stream to identify when harmful changes in its distribution (one that increases the risk) occur.

Developing better estimators of uncertainty improves the practical effectiveness of conformal prediction. The literature on this topic is too wide to even begin discussing; instead, we point to quantile regression as an example of a fruitful line of work that mingled especially nicely with conformal prediction in Section 2.2. Quantile regression was first proposed in [9] and extended to the locally polynomial case in [109]. Under sufficient regularity, quantile regression converges uniformly to the true quantile function [109–113]. Practical and accessible references for quantile regression have been written by Koenker and collaborators [114, 115]. Active work continues today to analyze the statistical properties of quantile regression and its variants under different conditions, for example in additive models [116] or to improve conditional coverage when the size of the intervals may correlate with miscoverage events [16]. The Handbook of Quantile Regression [115] includes more detail on such topics, and a memoir of quantile regression for the interested reader. Since quantile regression provides intervals with near-conditional coverage asymptotically, the conformalized version inherits this good behavior as well.

Along with such statistical advances has come a recent wave of practical applications of conformal prediction. Conformal prediction in large-scale deep learning was studied in [4], focusing on image classification.

One compelling use-case of conformal prediction is speeding up and decreasing the computational cost of the test-time evaluation of complex models [117, 118]. The same researchers pooled information across multiple tasks in a meta-learning setup to form tight prediction sets for few-shot prediction [119]. There is also an earlier line of work, appearing slightly after that of Lei and Wasserman, applying conformal prediction to decision trees [120–122]. Closer to end-users, we are aware of several real applications of conformal prediction. The Washington Post estimated the number of outstanding Democratic and Republican votes in the 2020 United States presidential election using conformal prediction [123]. Early clinical experiments in hospitals underscore the utility of conformal prediction in that setting as well, although real deployments are still to come [124, 125]. Fairness and reliability of algorithmic risk forecasts in the criminal justice system improves (on controlled datasets) when applying conformal prediction [125–127]. Conformal prediction was recently applied to create safe robotic planning algorithms that avoid bumping into objects [128, 129]. Recently a `scikit-learn` compatible open-source library, `MAPIE`, has been developed for constructing conformal prediction intervals. There remains a mountain of future work in these applications of conformal prediction and many others.

Today, the field of distribution-free uncertainty quantification remains small, but grows rapidly year-on-year. The promulgation of machine learning deployments has caused a reckoning that point predictions are not enough and shown that we still need rigorous statistical inference for reliable decision-making. Many researchers around the world have keyed into this fact and have created new algorithms and software using distribution-free ideas like conformal prediction. These developments are numerous and high-quality, so most reviews are out-of-date. To keep track of what gets released, the reader may want to see the `Awesome Conformal Prediction` repository [130], which provides a frequently-updated list of resources in this area.

We will end our Gentle Introduction with a personal note to the reader—you can be part of this story too. The infant field of distribution-free uncertainty quantification has ample room for significant technical contributions. Furthermore, the concepts are practical and approachable; they can easily be understood and implemented in code. Thus, we encourage the reader to try their hand at distribution-free uncertainty quantification; there is a lot more to be done!

References

[1] V. Vovk, A. Gammerman, and G. Shafer, *Algorithmic Learning in a Random World*. Springer, 2005.

[2] H. Papadopoulos, K. Proedrou, V. Vovk, and A. Gammerman, "Inductive confidence machines for regression," in *Machine Learning: European Conference on Machine Learning*, 2002, pp. 345–356.

[3] J. Lei and L. Wasserman, "Distribution-free prediction bands for non-parametric regression," *Journal of the Royal Statistical Society: Series B: Statistical Methodology*, pp. 71–96, 2014.

[4] A. N. Angelopoulos, S. Bates, J. Malik, and M. I. Jordan, "Uncertainty sets for image classifiers using conformal prediction," in *International Conference on Learning Representations*, 2021.

[5] V. Vovk, A. Gammerman, and C. Saunders, "Machine-learning applications of algorithmic randomness," in *International Conference on Machine Learning*, 1999, pp. 444–453.

[6] M. Sadinle, J. Lei, and L. Wasserman, "Least ambiguous set-valued classifiers with bounded error levels," *Journal of the American Statistical Association*, vol. 114, pp. 223–234, 2019.

[7] Y. Romano, M. Sesia, and E. J. Candès, "Classification with valid and adaptive coverage," *arXiv:2006.02544*, 2020.

[8] Y. Romano, E. Patterson, and E. Candès, "Conformalized quantile regression," in *Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 3543–3553.

[9] R. Koenker and G. Bassett Jr, "Regression quantiles," *Econometrica: Journal of the Econometric Society*, vol. 46, no. 1, pp. 33–50, 1978.

[10] A. N. Angelopoulos, A. P. Kohli, S. Bates, M. I. Jordan, J. Malik, T. Alshaabi, S. Upadhyayula, and Y. Romano, "Image-to-image regression with distribution-free uncertainty quantification and applications in imaging," *arXiv preprint arXiv:2202.05265*, 2022.

[11] P. Hoff, "Bayes-optimal prediction with frequentist coverage control," *arXiv:2105.14045*, 2021.

[12] L. Wasserman, "Frasian inference," *Statistical Science*, vol. 26, no. 3, pp. 322–325, 2011.

[13] T. Melluish, C. Saunders, I. Nouretdinov, and V. Vovk, "Comparing the bayes and typicalness frameworks," in *European Conference on Machine Learning*, Springer, 2001, pp. 360–371.

[14] V. Vovk, "Conditional validity of inductive conformal predictors," in *Proceedings of the Asian Conference on Machine Learning*, vol. 25, 2012, pp. 475–490.

[15] M. Cauchois, S. Gupta, and J. Duchi, "Knowing what you know: Valid and validated confidence sets in multiclass and multilabel prediction," *arXiv:2004.10181*, 2020.

[16] S. Feldman, S. Bates, and Y. Romano, "Improving conditional coverage via orthogonal quantile regression," in *Advances in Neural Information Processing Systems*, 2021.

[17] A. N. Angelopoulos, S. Bates, A. Fisch, L. Lei, and T. Schuster, "Conformal risk control," *arXiv preprint arXiv:2208.02814*, 2022.

[18] A. N. Angelopoulos, S. Bates, E. J. Candès, M. I. Jordan, and L. Lei, "Learn then test: Calibrating predictive algorithms to achieve risk control," *arXiv:2110.01052*, 2021.

[19] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Processing*, vol. 99, pp. 215–249, 2014.

[20] R. A. Fisher, "Design of experiments," *British Medical Journal*, vol. 1, no. 3923, p. 554, 1936.

[21] E. J. Pitman, "Significance tests which may be applied to samples from any populations," *Supplement to the Journal of the Royal Statistical Society*, vol. 4, no. 1, pp. 119–130, 1937.

[22] V. Vovk, I. Nouretdinov, and A. Gammerman, "Testing exchangeability on-line," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 768–775.

[23] L. Guan and R. Tibshirani, "Prediction and outlier detection in classification problems," *arXiv:1905.04396*, 2019.

[24] S. Bates, E. Candès, L. Lei, Y. Romano, and M. Sesia, "Testing for outliers with conformal p-values," *arXiv:2104.08279*, 2021.

[25] R. J. Tibshirani, R. Foygel Barber, E. Candes, and A. Ramdas, "Conformal prediction under covariate shift," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 2530–2540.

[26] R. F. Barber, E. J. Candes, A. Ramdas, and R. J. Tibshirani, "Conformal prediction beyond exchangeability," *arXiv:2202.13415*, 2022.

[27] L. Guan, "Conformal prediction with localization," *arXiv:1908.08558*, 2020.

[28] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, Springer, 2014, pp. 740–755.

[29] A. Malinin, N. Band, G. Chesnokov, Y. Gal, M. J. Gales, A. Noskov, A. Ploskonosov, L. Prokhorenkova, I. Provilkov, V. Raina, *et al.*, "Shifts: A dataset of real distributional shift across multiple large-scale tasks," *arXiv preprint arXiv:2107.07455*, 2021.

[30] A. V. Dorogush, V. Ershov, and A. Gulin, "Catboost: Gradient boosting with categorical features support," *arXiv preprint arXiv:1810.11363*, 2018.

[31] I. Gibbs and E. Candès, "Adaptive conformal inference under distribution shift," *arXiv:2106.00170*, 2021.

[32] M. Zaffran, O. Féron, Y. Goude, J. Josse, and A. Dieuleveut, "Adaptive conformal predictions for time series," in *International Conference on Machine Learning*, PMLR, 2022, pp. 25 834–25 866.

[33] I. Gibbs and E. Candès, "Conformal inference for online prediction with arbitrary distribution shifts," *arXiv preprint arXiv:2208.08401*, 2022.

[34] C. Xu and Y. Xie, "Conformal prediction interval for dynamic time-series," in *International Conference on Machine Learning*, PMLR, 2021, pp. 11 559–11 569.

[35] L. Hanu and Unitary team, *Detoxify*, Github. https://github.com/unitaryai/detoxify, 2020.

[36] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[37] P. W. Koh, S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, I. Gao, *et al.*, "Wilds: A benchmark of in-the-wild distribution shifts," in *International Conference on Machine Learning*, PMLR, 2021, pp. 5637–5664.

[38] G. Shafer and V. Vovk, "A tutorial on conformal prediction," *Journal of Machine Learning Research*, vol. 9, no. Mar, pp. 371–421, 2008.

[39] E. Ndiaye and I. Takeuchi, "Computing full conformal prediction set with approximate homotopy," in *Advances in Neural Information Processing Systems*, 2019.

[40] E. Ndiaye and I. Takeuchi, "Root-finding approaches for computing conformal prediction set," *Machine Learning*, 2022.

[41] V. Vovk, "Cross-conformal predictors," *Annals of Mathematics and Artificial Intelligence*, vol. 74, no. 1-2, pp. 9–28, 2015.

[42] R. F. Barber, E. J. Candes, A. Ramdas, and R. J. Tibshirani, "Predictive inference with the jackknife+," *The Annals of Statistics*, vol. 49, no. 1, pp. 486–507, 2021.

[43] E. Chung and J. P. Romano, "Exact and asymptotically robust permutation tests," *The Annals of Statistics*, vol. 41, no. 2, pp. 484–507, 2013.

[44] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The Annals of Mathematical Statistics*, pp. 50–60, 1947.

[45] E. L. Lehmann, "The power of rank tests," *The Annals of Mathematical Statistics*, pp. 23–43, 1953.

[46] Z. Sidak, P. K. Sen, and J. Hajek, *Theory of rank tests*. Elsevier, 1999.

[47] B. Efron and R. J. Tibshirani, *An introduction to the bootstrap*. CRC press, 1994.

[48] S. Chatterjee and P. Qiu, "Distribution-free cumulative sum control charts using bootstrap-based control limits," *The Annals of Applied Statistics*, vol. 3, no. 1, pp. 349–369, 2009.

[49] G. T. Fechner, *Kollektivmasslehre*. Engelmann, 1897.

[50] R. von Mises, "Grundlagen der wahrscheinlichkeitsrechnung," *Mathematische Zeitschrift*, vol. 5, no. 1, pp. 52–99, 1919.

[51] A. Wald, "Die widerspruchfreiheit des kollectivbegriffes der wahrscheinlichkeitsrechnung," *Ergebnisse Eines Mathematischen Kolloquiums*, vol. 8, no. 38-72, p. 37, 1937.

[52] A. Church, "On the concept of a random sequence," *Bulletin of the American Mathematical Society*, vol. 46, no. 2, pp. 130–135, 1940.

[53] J. Ville, "Etude critique de la notion de collectif," *Bull. Amer. Math. Soc*, vol. 45, no. 11, p. 824, 1939.

[54] G. Shafer and V. Vovk, "The sources of Kolmogorov's Grundbegriffe," *Statistical Science*, vol. 21, no. 1, pp. 70–98, 2006.

[55] V. Vovk, "Kolmogorov's complexity conception of probability," *Synthese Library*, pp. 51–70, 2001.

[56] C. P. Porter, "Kolmogorov on the role of randomness in probability theory," *Mathematical Structures in Computer Science*, vol. 24, no. 3, 2014.

[57] A. N. Kolmogorov, "Three approaches to the quantitative definition of information," *Problems of Information Transmission*, vol. 1, no. 1, pp. 1–7, 1965.

[58] A. Kolmogorov, "Logical basis for information theory and probability theory," *IEEE Transactions on Information Theory*, vol. 14, no. 5, pp. 662–664, 1968.

[59] A. N. Kolmogorov, "Combinatorial foundations of information theory and the calculus of probabilities," *Russian Mathematical Surveys*, vol. 38, no. 4, pp. 29–40, 1983.

[60] V. G. Vovk, "On the concept of the Bernoulli property," *Russian Mathematical Surveys*, vol. 41, no. 1, p. 247, 1986.

[61] V. Vovk, "Testing randomness online," *Statistical Science*, vol. 36, no. 4, pp. 595–611, 2021.

[62] F. Mota, S. Aaronson, L. Antunes, and A. Souto, "Sophistication as randomness deficiency," in *International Workshop on Descriptional Complexity of Formal Systems*, Springer, 2013, pp. 172–181.

[63] S. S. Wilks, "Determination of sample sizes for setting tolerance limits," *Annals of Mathematical Statistics*, vol. 12, no. 1, pp. 91–96, 1941.

[64] ——, "Statistical prediction with special reference to the problem of tolerance limits," *Annals of Mathematical Statistics*, vol. 13, no. 4, pp. 400–409, 1942.

[65] A. Wald, "An extension of Wilks' method for setting tolerance limits," *Annals of Mathematical Statistics*, vol. 14, no. 1, pp. 45–55, 1943.

[66] J. W. Tukey, "Non-parametric estimation II. Statistically equivalent blocks and tolerance regions–the continuous case," *Annals of Mathematical Statistics*, vol. 18, no. 4, pp. 529–539, 1947.

[67] P. Diaconis and D. Freedman, "Finite exchangeable sequences," *The Annals of Probability*, pp. 745–764, 1980.

[68] D. J. Aldous, "Exchangeability and related topics," in *École d'Été de Probabilités de Saint-Flour XIII—1983*, 1985, pp. 1–198.

[69] B. De Finetti, "Funzione caratteristica di un fenomeno aleatorio," in *Atti del Congresso Internazionale dei Matematici: Bologna del 3 al 10 de Settembre di 1928*, 1929, pp. 179–190.

[70] D. A. Freedman, "Bernard Friedman's urn," *The Annals of Mathematical Statistics*, pp. 956–970, 1965.

[71] E. Hewitt and L. J. Savage, "Symmetric measures on Cartesian products," *Transactions of the American Mathematical Society*, vol. 80, no. 2, pp. 470–501, 1955.

[72] J. F. Kingman, "Uses of exchangeability," *The Annals of Probability*, vol. 6, no. 2, pp. 183–197, 1978.

[73] E. Dobriban, *Topics in Modern Statistical Learning (STAT 991, UPenn, 2022 Spring)*, Dec. 2022.

[74] A. Gammerman, V. Vovk, and V. Vapnik, "Learning by transduction," *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, vol. 14, pp. 148–155, 1998.

[75] C. Saunders, A. Gammerman, and V. Vovk, "Transduction with confidence and credibility," 1999.

[76] V. Vovk, "On-line confidence machines are well-calibrated," in *The 43rd Annual IEEE Symposium on Foundations of Computer Science*, IEEE, 2002, pp. 187–196.

[77] V. Vovk, G. Shafer, and I. Nouretdinov, "Self-calibrating probability forecasting.," in *Neural Information Processing Systems*, 2003, pp. 1133–1140.

[78] V. Vovk and I. Petej, "Venn-Abers predictors," *arXiv:1211.0025*, 2012.

[79] V. Vovk, J. Shen, V. Manokhin, and M.-g. Xie, "Nonparametric predictive distributions based on conformal prediction," *Machine Learning*, pp. 1–30, 2017.

[80] J. Lei, J. Robins, and L. Wasserman, "Efficient nonparametric conformal prediction regions," *arXiv:1111.1418*, 2011.

[81] ——, "Distribution-free prediction sets," *Journal of the American Statistical Association*, vol. 108, no. 501, pp. 278–287, 2013.

[82] B. Póczos, A. Singh, A. Rinaldo, and L. Wasserman, "Distribution-free distribution regression," in *Artificial Intelligence and Statistics*, PMLR, 2013, pp. 507–515.

[83] 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*, vol. 113, no. 523, pp. 1094–1111, 2018.

[84] J. Lei, A. Rinaldo, and L. Wasserman, "A conformal prediction approach to explore functional data," *Annals of Mathematics and Artificial Intelligence*, vol. 74, pp. 29–43, 2015.

[85] J. Lei, "Fast exact conformalization of the lasso using piecewise linear homotopy," *Biometrika*, vol. 106, no. 4, pp. 749–764, 2019.

[86] C. Gupta, A. K. Kuchibhotla, and A. Ramdas, "Nested conformal prediction and quantile out-of-bag ensemble methods," *Pattern Recognition*, p. 108 496, 2021.

[87] R. Foygel Barber, E. J. Candes, A. Ramdas, and R. J. Tibshirani, "The limits of distribution-free conditional predictive inference," *Information and Inference: A Journal of the IMA*, vol. 10, no. 2, pp. 455–482, 2021.

[88] Y. Lee and R. F. Barber, "Distribution-free inference for regression: Discrete, continuous, and in between," *arXiv:2105.14075*, 2021.

[89] R. Izbicki, G. Shimizu, and R. Stern, "Flexible distribution-free conditional predictive bands using density estimators," in *Proceedings of Machine Learning Research*, vol. 108, PMLR, 2020, pp. 3068–3077.

[90] J. Lei, "Classification with confidence," *Biometrika*, vol. 101, no. 4, pp. 755–769, Oct. 2014.

[91] Y. Hechtlinger, B. Poczos, and L. Wasserman, "Cautious deep learning," *arXiv:1805.09460*, 2018.

[92] D. Stutz, K. D. Dvijotham, A. T. Cemgil, and A. Doucet, "Learning optimal conformal classifiers," in *International Conference on Learning Representations*, 2022.

[93] M. Cauchois, S. Gupta, A. Ali, and J. C. Duchi, "Robust validation: Confident predictions even when distributions shift," *arXiv:2008.04267*, 2020.

[94] L. Lei and E. J. Candès, "Conformal inference of counterfactuals and individual treatment effects," *arXiv:2006.06138*, 2020.

[95] M. Yin, C. Shi, Y. Wang, and D. M. Blei, "Conformal sensitivity analysis for individual treatment effects," *arXiv:2112.03493*, 2021.

[96] V. Chernozhukov, K. Wüthrich, and Y. Zhu, "An exact and robust conformal inference method for counterfactual and synthetic controls," *Journal of the American Statistical Association*, pp. 1–16, 2021.

[97] E. J. Candès, L. Lei, and Z. Ren, "Conformalized survival analysis," *arXiv:2103.09763*, 2021.

[98] A. N. Angelopoulos, S. Bates, T. Zrnic, and M. I. Jordan, "Private prediction sets," *arXiv:2102.06202*, 2021.

[99] V. Chernozhukov, K. Wüthrich, and Z. Yinchu, "Exact and robust conformal inference methods for predictive machine learning with dependent data," in *Conference On Learning Theory*, PMLR, 2018, pp. 732–749.

[100] R. Dunn, L. Wasserman, and A. Ramdas, "Distribution-free prediction sets with random effects," *arXiv:1809.07441*, 2018.

[101] R. I. Oliveira, P. Orenstein, T. Ramos, and J. V. Romano, "Split conformal prediction for dependent data," *arXiv:2203.15885*, 2022.

[102] O. Bastani, V. Gupta, C. Jung, G. Noarov, R. Ramalingam, and A. Roth, "Practical adversarial multivalid conformal prediction," in *Advances in Neural Information Processing Systems*, A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, Eds., 2022.

[103] C. Jung, G. Noarov, R. Ramalingam, and A. Roth, "Batch multivalid conformal prediction," *arXiv preprint arXiv:2209.15145*, 2022.

[104] C. Gupta and A. Ramdas, "Distribution-free calibration guarantees for histogram binning without sample splitting," in *International Conference on Machine Learning*, vol. 139, 2021, pp. 3942–3952.

[105] S. Park, S. Li, O. Bastani, and I. Lee, "PAC confidence predictions for deep neural network classifiers," in *International Conference on Learning Representations*, 2021.

[106] D. Volkhonskiy, E. Burnaev, I. Nouretdinov, A. Gammerman, and V. Vovk, "Inductive conformal martingales for change-point detection," in *Conformal and Probabilistic Prediction and Applications*, PMLR, 2017, pp. 132–153.

[107] X. Hu and J. Lei, "A distribution-free test of covariate shift using conformal prediction," *arXiv:2010.07147*, 2020.

[108] A. Podkopaev and A. Ramdas, "Tracking the risk of a deployed model and detecting harmful distribution shifts," *arXiv:2110.06177*, 2021.

[109] P. Chaudhuri, "Global nonparametric estimation of conditional quantile functions and their derivatives," *Journal of Multivariate Analysis*, vol. 39, no. 2, pp. 246–269, 1991.

[110] I. Steinwart and A. Christmann, "Estimating conditional quantiles with the help of the pinball loss," *Bernoulli*, vol. 17, no. 1, pp. 211–225, 2011.

[111] I. Takeuchi, Q. V. Le, T. D. Sears, and A. J. Smola, "Nonparametric quantile estimation," *Journal of Machine Learning Research*, vol. 7, pp. 1231–1264, 2006.

[112] K. Q. Zhou, S. L. Portnoy, *et al.*, "Direct use of regression quantiles to construct confidence sets in linear models," *The Annals of Statistics*, vol. 24, no. 1, pp. 287–306, 1996.

[113] K. Q. Zhou and S. L. Portnoy, "Statistical inference on heteroscedastic models based on regression quantiles," *Journal of Nonparametric Statistics*, vol. 9, no. 3, pp. 239–260, 1998.

[114] R. Koenker, *Quantile Regression*. Cambridge University Press, 2005.

[115] R. Koenker, V. Chernozhukov, X. He, and L. Peng, "Handbook of quantile regression," 2018.

[116] R. Koenker, "Additive models for quantile regression: Model selection and confidence bandaids," *Brazilian Journal of Probability and Statistics*, vol. 25, no. 3, pp. 239–262, 2011.

[117] A. Fisch, T. Schuster, T. S. Jaakkola, and R. Barzilay, "Efficient conformal prediction via cascaded inference with expanded admission," in *International Conference on Learning Representations*, 2021.

[118] T. Schuster, A. Fisch, T. Jaakkola, and R. Barzilay, "Consistent accelerated inference via confident adaptive transformers," *Empirical Methods in Natural Language Processing*, 2021.

[119] A. Fisch, T. Schuster, T. Jaakkola, and D. Barzilay, "Few-shot conformal prediction with auxiliary tasks," in *International Conference on Machine Learning*, vol. 139, 2021, pp. 3329–3339.

[120] U. Johansson, H. Boström, T. Löfström, and H. Linusson, "Regression conformal prediction with random forests," *Machine learning*, vol. 97, no. 1, pp. 155–176, 2014.

[121] H. Linusson, U. Norinder, H. Boström, U. Johansson, and T. Löfström, "On the calibration of aggregated conformal predictors," in *Conformal and probabilistic prediction and applications*, PMLR, 2017, pp. 154–173.

[122] H. Boström, H. Linusson, T. Löfström, and U. Johansson, "Accelerating difficulty estimation for conformal regression forests," *Annals of Mathematics and Artificial Intelligence*, vol. 81, no. 1, pp. 125–144, 2017.

[123] J. Cherian and L. Bronner, "How the Washington Post estimates outstanding votes for the 2020 presidential election," *Washington Post*, 2021, https://s3.us-east-1.amazonaws.com/elex-models-prod/2020-general/write-up/election_model_writeup.pdf.

[124] C. Lu and J. Kalpathy-Cramer, "Distribution-free federated learning with conformal predictions," *arXiv:2110.07661*, 2021.

[125] C. Lu, A. Lemay, K. Chang, K. Hoebel, and J. Kalpathy-Cramer, "Fair conformal predictors for applications in medical imaging," *arXiv:2109.04392*, 2021.

[126] Y. Romano, R. F. Barber, C. Sabatti, and E. Candès, "With malice toward none: Assessing uncertainty via equalized coverage," *Harvard Data Science Review*, vol. 2, no. 2, Apr. 30, 2020.

[127] A. K. Kuchibhotla and R. A. Berk, "Nested conformal prediction sets for classification with applications to probation data," *arXiv:2104.09358*, 2021.

[128] L. Lindemann, M. Cleaveland, G. Shim, and G. J. Pappas, "Safe planning in dynamic environments using conformal prediction," *arXiv preprint arXiv:2210.10254*, 2022.

[129] A. Dixit, L. Lindemann, M. Cleaveland, S. Wei, G. J. Pappas, and J. W. Burdick, "Adaptive conformal prediction for motion planning among dynamic agents," *arXiv preprint arXiv:2212.00278*, 2022.

[130] V. Manokhin, *Awesome Conformal Prediction*, version v1.0.0, Apr. 2022.

[131] S. Bates, A. Angelopoulos, L. Lei, J. Malik, and M. Jordan, "Distribution-free, risk-controlling prediction sets," *Journal of the Association for Computing Machinery*, vol. 68, no. 6, Sep. 2021.

[132] F. Bretz, W. Maurer, W. Brannath, and M. Posch, "A graphical approach to sequentially rejective multiple test procedures," *Statistics in Medicine*, vol. 28, no. 4, pp. 586–604, 2009.

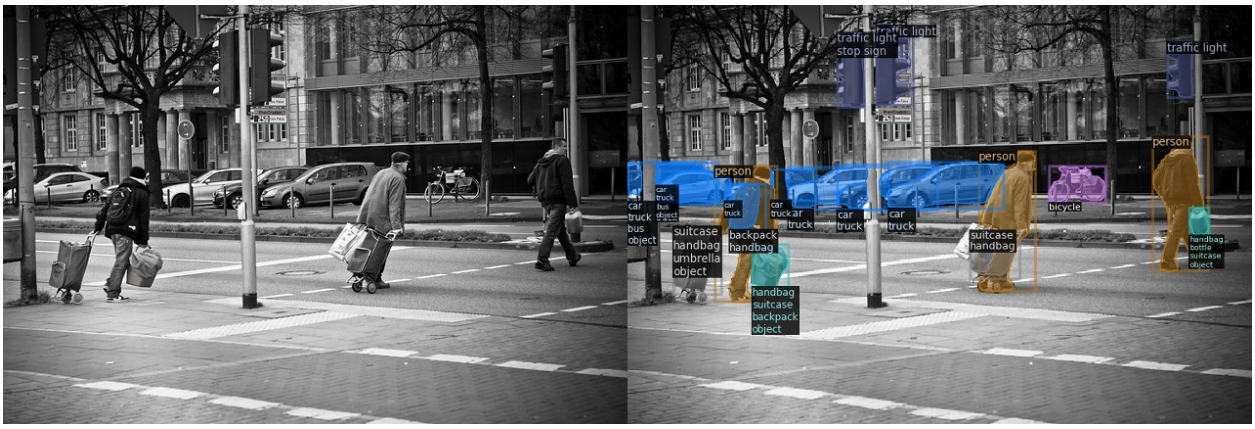
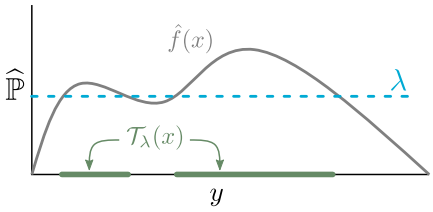


Figure 19: Object detection with simultaneous distribution-free guarantees *on the expected intersection-over-union, recall, and coverage rate.*

A Distribution-Free Control of General Risks

For many prediction tasks, the relevant notion of reliability is not coverage. Indeed, many applications have problem-specific performance metrics—from false-discovery rate to fairness—that directly encode the soundness of a prediction. In Section 4.3, we saw how to control the expectation of monotone loss functions using conformal risk control. Here, we generalize further to control *any* risk and multiple risks in a distribution-free way without retraining the model. As an example, in instance segmentation, we are given an image and asked to identify all objects in the image, segment them, and classify them. All three of these sub-tasks have their own risks: recall, *intersection-over-union* (IOU), and coverage respectively. These risks can be automatically controlled using distribution-free statistics, as we preview in Figure 19.

We first re-introduce the theory of risk control below, then give a list of illustrative examples. As in conformal risk control, we start with a pretrained model $\hat{f}$. The model also has a *parameter* λ, which we are free to choose. We use $\hat{f}(x)$ and λ to form our prediction, $\mathcal{T}_\lambda(x)$, which may be a set or some other object. For example, when performing regression, λ could threshold the estimated probability density, as below.



We then define a notion of risk $R(\lambda)$. The risk function measures the quality of $\mathcal{T}_\lambda$ according to the user. The goal of risk control is to use our calibration set to pick a parameter $\hat{\lambda}$ so that the risk is small with high probability. In formal terms, for a user-defined *risk tolerance* α and *error rate* δ, we seek to ensure

$$\mathbb{P}\left(R(\hat{\lambda}) < \alpha\right) \geq 1 - \delta, \tag{17}$$

where the probability is taken over the calibration data used to pick $\hat{\lambda}$. Note that this guarantee is high-probability, unlike that in Section 4.3, which is in expectation. We will soon introduce a distribution-free technique called *Learn then Test* (LTT) for finding $\hat{\lambda}$ that satisfy (17). Below we include two example applications of risk control which would be impossible with conformal prediction and conformal risk control.

- *Multi-label Classification with FDR Control:* In this setting, X_{test} is an image and Y_{test} is a subset of K classes contained in the image. Our model $\hat{f}$ gives us the probability each of the K classes is contained in the image. We will include a class in our estimate of y if $\hat{f}_k > \lambda$ — i.e., the parameter

λ thresholds the estimated probabilities. We seek to find the $\hat{\lambda}$s that guarantees our predicted set of labels is sufficiently reliable as measured by the *false-discovery rate* (FDR) risk $R(\hat{\lambda})$.

- *Simultaneous Guarantees on OOD Detection and Coverage:* For each input X_{test} with true class Y_{test}, we want to decide if it is out-of-distribution. If so, we will flag it as such. Otherwise, we want to output a prediction set that contains the true class with 90% probability. In this case, we have two models: $\text{OOD}(x)$, which tells us how OOD the input is, and $\hat{f}(x)$, which gives the estimated probability that the input comes from each of K classes. In this case, λ has two coordinates, and we also have two risks. The first coordinate λ_1 tells us where to threshold $\text{OOD}(x)$ such that the fraction of false alarms R_1 is controlled. The second coordinate λ_2 tells us how many classes to include in the prediction set to control the miscoverage R_2 among points identified as in-distribution. We will find $\hat{\lambda}$s that control both $R_1(\hat{\lambda})$ and $R_2(\hat{\lambda})$ jointly.

We will describe each of these examples in detail in Section B. Many more worked examples, including the object detection example in Figure 19, are available in the cited literature on risk control [18, 131]. First, however, we will introduce the general method of risk control via Learn then Test.

A.1 Instructions for Learn then Test

First, we will describe the formal setting of risk control. We introduce notation and the risk-control property in Definition 2. Then, we describe the calibration algorithm.

Formal notation for error control

Let $(X_i, Y_i)_{i=1,\ldots,n}$ be an independent and identically distributed (i.i.d.) set of variables, where the features X_i take values in $\mathcal{X}$ and the responses Y_i take values in $\mathcal{Y}$. The researcher starts with a pre-trained predictive model $\hat{f}$. We show how to subsequently create predictors from $\hat{f}$ that control a risk, regardless of the quality of the initial model fit or the distribution of the data.

Next, let $\mathcal{T}_\lambda : \mathcal{X} \to \mathcal{Y}'$ be a function with parameter λ that maps a feature to a prediction ($\mathcal{Y}'$ can be any space, including the space of responses $\mathcal{Y}$ or prediction sets $2^{\mathcal{Y}}$). This function $\mathcal{T}_\lambda$ would typically be constructed from the predictive model, $\hat{f}$, as in our earlier regression example. We further assume λ takes values in a (possibly multidimensional) discrete set Λ. If Λ is not naturally discrete, we usually discretize it finely. For example, Λ could be the set $\{0, 0.001, 0.002, ..., 0.999, 1\}$.

We then allow the user to choose a *risk* for the predictor $\mathcal{T}_\lambda$. This risk can be any function of $\mathcal{T}_\lambda$, but often we take the risk function to be the expected value of a *loss function*,

$$R(\mathcal{T}_\lambda) = \mathbb{E}\left[\underbrace{L\big(\mathcal{T}_\lambda(X_{\text{test}}), Y_{\text{test}}\big)}_{\text{Loss function}}\right]. \tag{18}$$

The loss function is a deterministic function that is high when $\mathcal{T}_\lambda(X_{\text{test}})$ does badly at predicting Y_{test}. The risk then averages this loss over the distribution of $(X_{\text{test}}, Y_{\text{test}})$. For example, taking

$$R_{\text{miscoverage}}\big(\mathcal{T}_\lambda\big) = \mathbb{E}\big[\mathbb{1}\left\{Y_{\text{test}} \notin \mathcal{T}_\lambda(X_{\text{test}})\right\}\big] = \mathbb{P}\left(Y_{\text{test}} \notin \mathcal{T}_\lambda(X_{\text{test}})\right)$$

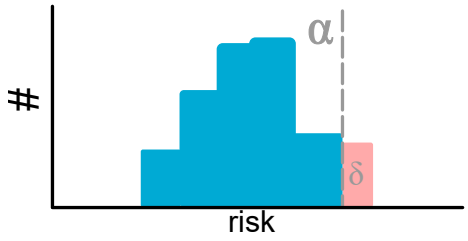
gives us the familiar case of controlling miscoverage.

To aid the reader, we point out some facts about (18) that may not be obvious. The input $\mathcal{T}_\lambda$ to the risk is a function; this makes the risk a *functional* (a function of a function). When we plug $\mathcal{T}_\lambda$ into the risk, we take an expectation of the loss over the randomness in a single test point. At the end of the process, for a deterministic λ, we get a deterministic scalar $R(\mathcal{T}_\lambda)$. Henceforth, for ease of notation, we abbreviate this number as $R(\lambda) := R(\mathcal{T}_\lambda)$.

Our goal is control the risk in the following sense:

Definition 2 (Risk control)**.** *Let $\hat{\lambda}$ be a random variable taking values in Λ (i.e., the output of an algorithm run on the calibration data). We say that $\mathcal{T}_{\hat{\lambda}}$ is a* (α, δ)-risk-controlling prediction *(RCP) if, with probability at least $1 - \delta$, we have $R(\hat{\lambda}) \leq \alpha$.*

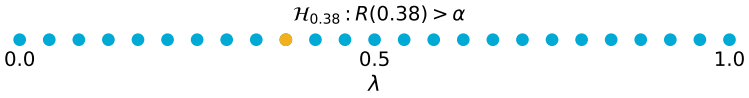
In Definition 2, we plug in a *random parameter* $\hat{\lambda}$ which is chosen based on our calibration data; therefore, $R(\hat{\lambda})$ is random even though the risk is a deterministic function. The high-probability portion of Definition 2 therefore says that $\hat{\lambda}$ can only violate risk control if we choose a bad calibration set; this happens with probability at most δ. The distribution of the risk over many resamplings of the calibration data should therefore look as below.



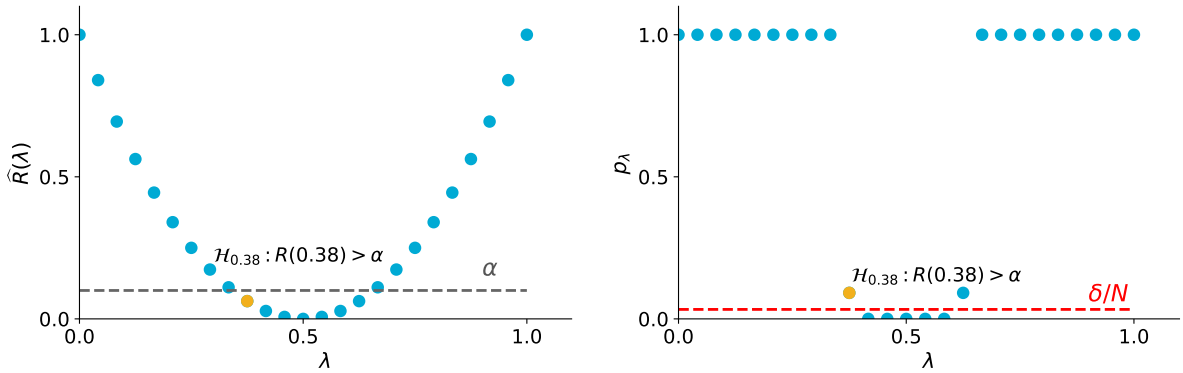
The Learn then Test procedure

Recalling Definition 2, our goal is to find a set function whose risk is less than some user-specified threshold α. To do this, we search across the collection of functions $\{\mathcal{T}_\lambda\}_{\lambda\in\Lambda}$ and estimate their risk on the calibration data $(X_1, Y_1), \ldots, (X_n, Y_n)$. The output of the procedure will be a set of λ values which are all guaranteed to control the risk, $\widehat{\Lambda} \subseteq \Lambda$. The Learn then Test procedure is outlined below.

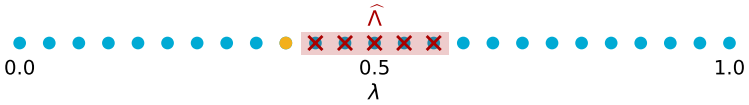
1. For each $\lambda \in \Lambda$, associate the null hypothesis $\mathcal{H}_\lambda : R(\lambda) > \alpha$. Notice that *rejecting* the $\mathcal{H}_\lambda$ means you selected λ as a point where the risk is controlled. Here we denote each null with a blue dot; the yellow dot is highlighted, so we can keep track of it as we explain the procedure.



2. For each null hypothesis, compute a p-value using a concentration inequality. For example, Hoeffding's inequality yields $p_\lambda = e^{-2n(\alpha - \widehat{R}(\lambda))_+^2}$, where $\widehat{R}(\lambda) = \frac{1}{n}\sum_{i=1}^{n} L(\mathcal{T}_\lambda(X_i), Y_i)$. We remind the reader what a p-value is, why it is relevant to risk control, and point to references with stronger p-values in A.1.1.



3. Return $\widehat{\Lambda} = \mathcal{A}\big(\{p_\lambda\}_{\lambda\in\Lambda}\big)$, where $\mathcal{A}$ is an algorithm that controls the familywise-error rate (FWER). For example, the Bonferroni correction yields $\widehat{\Lambda} = \big\{\lambda : p_\lambda < \frac{\delta}{|\Lambda|}\big\}$. We define the FWER and preview ways to design good FWER-controlling procedures in Section A.1.2. The nulls with red crosses through them below have been rejected by the procedure; i.e., they all control the risk with high probability.



By following the above procedure, we get the statistical guarantee in Theorem A.1.

Theorem A.1. *The $\widehat{\Lambda}$ returned by the Learn then Test procedure satisfies*

$$\mathbb{P}\left(\sup_{\hat{\lambda}\in\widehat{\Lambda}}\{R(\hat{\lambda})\} \leq \alpha\right) \geq 1 - \delta.$$

Thus, selecting any $\hat{\lambda} \in \widehat{\Lambda}$, $\mathcal{T}_{\hat{\lambda}}$ is an (α, δ)-RCP.

The LTT procedure decomposes risk control into two subproblems: computing p-values and combining them with multiple testing. We will now take a closer look at each of these subproblems.

```
# Implementation of LTT. Assume access to X, Y where n=X.shape[0]=Y.shape[0]
lambdas = torch.linspace(0,1,N) # Commonly choose N=1000
losses = torch.zeros((n,N)) # Compute the loss function next
for (i,j) in [(i,j) for i in range(n) for j in range(N)]:
  prediction_set = T(X[i],lambdas[j]) # T ( ) is problem depemdent
  losses[i,j] = get_loss(prediction_set,Y[i]) # Loss is problem dependent
risk = losses.mean(dim=0)
pvals = torch.exp(-2*n*(torch.relu(alpha-risk)**2)) # Or any p-value
lambda_hat = lambdas[pvals<delta/lambdas.shape[0]] # Or any FWER-controlling algorithm
```

Figure 20: PyTorch code for running Learn then Test.

A.1.1 Crash Course on Generating p-values

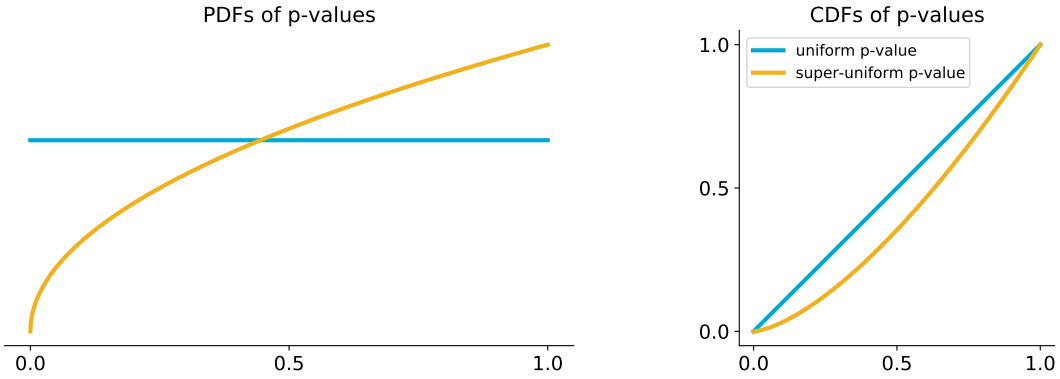
What is a p-value, and why is it related to risk control? In Step 1 of the LTT procedure, we associated a null hypothesis $\mathcal{H}_\lambda$ to every $\lambda \in \Lambda$. When the null hypothesis at λ holds, the risk is *not* controlled for that value of the parameter. In this reframing, our task is to automatically identify points λ where the null hypothesis does not hold—i.e., to *reject the null hypotheses* for some subset of λ such that $R(\lambda) \leq \alpha$. The process of accepting or rejecting a null hypothesis is called *hypothesis testing*.

$$\begin{aligned} \text{Rejecting the null hypothesis } \mathcal{H}_\lambda &\rightarrow \text{ the risk } \textit{is} \text{ controlled at } \lambda. \\ \text{Accepting the null hypothesis } \mathcal{H}_\lambda &\rightarrow \text{ the risk } \textit{is not} \text{ controlled at } \lambda. \end{aligned}$$

In order to reject a null hypothesis, we need to have empirical evidence that at λ, the risk is controlled. We use our calibration data to summarize this information in the form of a *p-value* p_λ. A p-value must satisfy the following condition, which we sometimes refer to as *validity* or *super-uniformity*,

$$\forall t \in [0,1], \;\; \mathbb{P}_{\mathcal{H}_\lambda}\left(p_\lambda \leq t\right) \leq t,$$

where $\mathbb{P}_{\mathcal{H}_\lambda}$ refers to the probability under the null hypothesis. Parsing the super-uniformity condition carefully tells us that when p_λ is low, there is evidence against the null hypothesis $\mathcal{H}_\lambda$. In other words, for a particular λ, we can reject $\mathcal{H}_\lambda$ if $p_\lambda < 5\%$ and expect to be wrong no more than 5% of the time. This process is called *testing the hypothesis at level* δ, where in the previous sentence, $\delta = 5\%$.



One of the key ingredients in Learn then Test is a p-value with distribution-free validity: it is valid under without assumptions on the data distribution. For example, when working with risk functions that take values in $[0,1]$—like coverage, IOU, FDR, and so on—the easiest choice of p-value is based on Hoeffding's inequality:

$$p_\lambda^{\text{Hoeffding}} = e^{-2n\left(\alpha - \widehat{R}(\lambda)\right)_+^2}.$$

More powerful p-values based on tighter concentration bounds are included in [18]. In particular, many of the practical examples in that reference use a stronger p-value called the *Hoeffding-Bentkus* (HB) p-value,

$$p_\lambda^{\text{HB}} = \min\left(\exp\{-nh_1(\widehat{R}(\lambda) \wedge \alpha, \alpha)\}, e\mathbb{P}\big(\text{Bin}(n,\alpha) \leq \left\lceil n\widehat{R}(\lambda)\right\rceil\big)\right),$$
$$\text{where } h_1(a,b) = a\log\left(\frac{a}{b}\right) + (1-a)\log\left(\frac{1-a}{1-b}\right).$$

Note that any valid p-value will work—it is fine for the reader to keep $p_\lambda^{\text{Hoeffding}}$ in mind for the rest of this manuscript, with the understanding that more powerful choices are available.

A.1.2 Crash Course on Familywise-Error Rate Algorithms

If we only had one hypothesis H_λ, we could simply test it at level δ. However, we have one hypothesis for each $\lambda \in \Lambda$, where $|\Lambda|$ is often very large (in the millions or more). This causes a problem: the more hypotheses we test, the higher chance we incorrectly reject at least one hypothesis. We can formally reason about this with the *familywise-error rate* (FWER).

Definition 3 (familywise-error rate)**.** *The familywise-error rate of a procedure returning $\hat{\Lambda}$ is the probability of making at least one false rejection, i.e.,*

$$\text{FWER}\left(\widehat{\Lambda}\right) = \mathbb{P}\left(\exists\hat{\lambda} \in \widehat{\Lambda} : R(\hat{\lambda}) > \alpha\right).$$

As a simple example to show how naively thresholding the p-values at level δ fails to control FWER, consider the case where all the hypotheses are null, and we have uniform p-values independently tested at level δ. The FWER then approaches 1; see below.

$$\text{If we take } \widehat{\Lambda} = \{\lambda : p_\lambda < \delta\}, \text{ then } \text{FWER}(\widehat{\Lambda}) = 1 - (1-\delta)^{|\Lambda|}.$$

This simple toy analysis exposes a deeper problem: without an intelligent strategy for combining the information from many p-values together, we can end up making false rejections with high probability. Our challenge is to intelligently combine the p-values to avoid this issue of multiplicity (without assuming the p-values are independent).

This fundamental statistical challenge has led to a decades-long and continually rich area of research called *multiple hypothesis testing*. In particular, a genre of algorithms called *FWER-controlling algorithms* seek to select the largest set of $\widehat{\Lambda}$ that guarantees $\text{FWER}(\widehat{\Lambda}) \leq \delta$. The simplest FWER-controlling algorithm is the *Bonferroni correction*,

$$\widehat{\Lambda}_{\text{Bonferroni}} = \left\{\lambda \in \Lambda : p_\lambda \leq \frac{\delta}{|\Lambda|}\right\}.$$

Under the hood, the Bonferroni correction simply tests each hypothesis at level $\delta/|\Lambda|$, so the probability there exists a failed test is no more than δ by a union bound. It should not be surprising that there exist improvements on Bonferroni correction.

First, we will discuss one important improvement in the case of a monotone loss function: *fixed-sequence testing*. As the name suggests, in fixed-sequence testing, we construct a sequence of hypotheses $\{\mathcal{H}_{\lambda_j}\}_{j=1}^N$ where $\text{N} = |\Lambda|$, before looking at our calibration data. Usually, we just sort our hypotheses from most- to least-promising based on information we knew a-priori. For example, if large values of λ are more likely to control the risk, $\{\lambda_j\}_{j=1}^N$ just sorts Λ from greatest to least. Then, we test the hypotheses sequentially in some fixed order at level δ, including them in $\widehat{\Lambda}$ as we go, and stopping when we make our first acceptance:

$$\widehat{\Lambda}_{\text{FST}} = \{\lambda_j, j \leq T\}, \text{ where } T = \max\left\{t \in \{1,...,N\} : p_{\lambda_{t'}} \leq \delta, \text{ for all } t' \leq t\right\}.$$

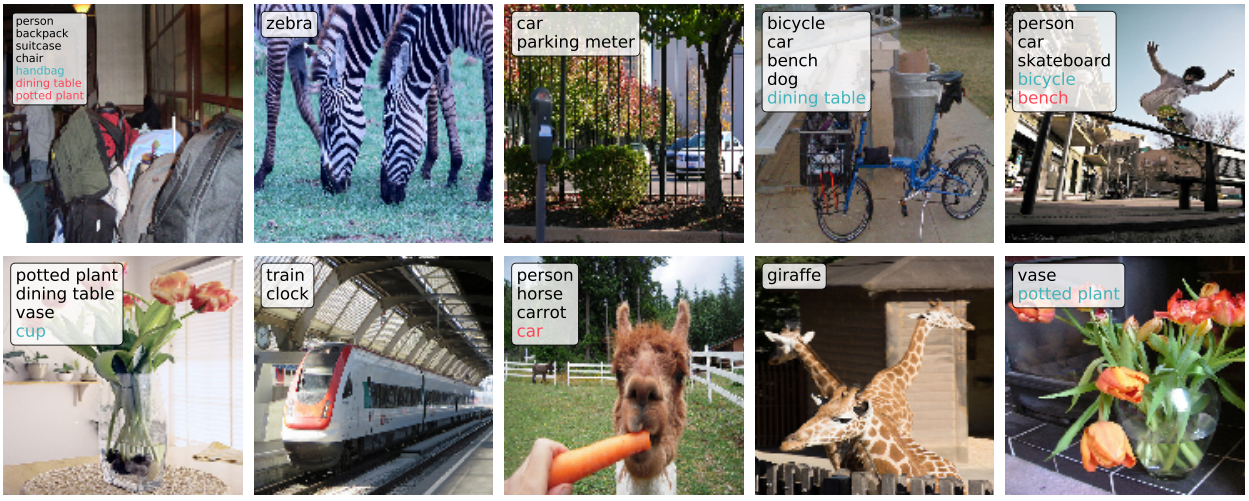


Figure 22: Examples of multi-label classification with FDR control *on the MS-COCO dataset. Black classes are true positives, blue classes are spurious, and red classes are missed. The FDR is controlled at level $\alpha = 0.1$, $\delta = 0.1$.*

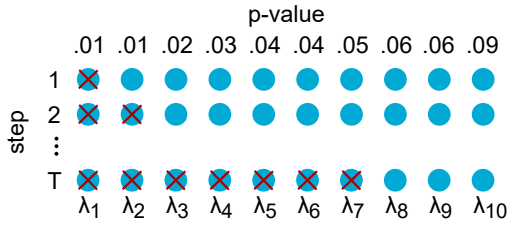


Figure 21: An example of fixed-sequence testing *with $\delta = 0.05$. Each blue circle represents a null, and each row a step of the procedure. The nulls with a red cross have been rejected at that step.*

This sequential procedure, despite testing all hypotheses it encounters at level δ, still controls the FWER. For monotone and near-monotone risks, such as the false-discovery rate, it works quite well.

It is also possible to extend the basic idea of fixed-sequence testing to non-monotone functions, creating powerful and flexible FWER-controlling procedures using an idea called sequential graphical testing [132]. Good graphical FWER-controlling procedures can be designed to have high power for particular problems, or alternatively, automatically discovered using data. This topic is given a detailed treatment in [18], and we omit it here for simplicity.

We have described a general-purpose pipeline for distribution-free risk control. It is described in PyTorch code in Figure 20. Once the user sets up the problem (i.e., picks Λ, $\mathcal{T}_\lambda$, and R), the LTT pipeline we described above automatically produces $\widehat{\Lambda}$. We now go through three worked examples which teach the reader how to choose Λ, $\mathcal{T}$ and R in practical circumstances.

B Examples of Distribution-Free Risk Control

In this section, we will walk through several examples of distribution-free risk control applied to practical machine learning problems. The goal is again to arm the reader with an arsenal of pragmatic prototypes of distribution-free risk control that work on real problems.

B.1 Multi-label Classification with FDR Control

We begin our sequence of examples with a familiar and fundamental setup: multi-label classification. Here, the features X_{test} can be anything (e.g. an image), and the label $Y_{\text{test}} \subseteq \{1, ..., K\}$ must be a set of classes (e.g. those contained in the image X_{test}). We have a pre-trained machine learning model $\hat{f}(x)$, which gives us an estimated probability $\hat{f}(x)_k$ that class k is in the corresponding set-valued label. We will use these probabilities to include the estimated most likely classes in our prediction set,

$$\mathcal{T}_\lambda(x) = \left\{k : \hat{f}(x)_k > \lambda\right\}, \quad \lambda \in \Lambda$$

where $\Lambda = \{0, 0.001, ..., 1\}$ (a discretization of $[0, 1]$). However, one question remains: *how do we choose* λ?

LTT will allow us to identify values of λ that satisfy a precise probabilistic guarantee—in this case, a bound on the *false-discovery rate* (FDR),

$$R_{\text{FDR}}(\lambda) = \mathbb{E}\left[\underbrace{1 - \frac{|Y_{\text{test}} \cap \mathcal{T}_\lambda(X_{\text{test}})|}{|\mathcal{T}_\lambda(X_{\text{test}})|}}_{L_{\text{FDP}}(\mathcal{T}_\lambda(X_{\text{test}}), Y_{\text{test}})}\right].$$

As annotated in the underbrace, the FDR is the expectation of a loss function, the *false-discovery proportion* (FDP). The FDP is low when our prediction set $\mathcal{T}_\lambda(X_{\text{test}})$ contains mostly elements from Y_{test}. In this sense, the FDR measures the quality of our prediction set: if we have a low FDR, it means most of the elements in our prediction set are good. By setting $\alpha = 0.1$ and $\delta = 0.1$, we desire that

$$\mathbb{P}\left[R_{\text{FDR}}(\hat{\lambda}) > 0.1\right] < 0.1,$$

where the probability is over the randomness in the calibration set used to pick $\hat{\lambda}$.

```
# model is a multi-class neural network, X.shape[0]=Y.shape[0]=n
lambdas = torch.linspace(0,1,N) # N can be taken to infinity without penalty
losses = torch.zeros((n,N)) # loss for example i with parameter lambdas[j]
for i in range(n): # In reality we parallelize these loops massively
  sigmoids = model(X[i].unsqueeze(0)).sigmoid().squeeze() # Care with dims
  for j in range(N):
    T = sigmoids > lambdas[j] # This is the prediction set
    set_size = T.float().sum()
    if set_size != 0:
      losses[i,j] = 1 - (T[Y] == True).float().sum()/set_size
risk = losses.mean(dim=0)
pvals = torch.exp(-2*n*(torch.relu(alpha-risk)**2)) # Or the HB p-value
# Fixed-sequence test starting at lambdas[-1] and ending at lambdas[0]
below_delta = (pvals <= delta).float()
valid = torch.tensor([(below_delta[j:].mean() == 1) for j in range(N)])
lambda_hat = lambdas[valid]
```

Figure 23: PyTorch code for performing FDR control with LTT.

Now that we have set up our problem, we can just run the LTT procedure via the code in Figure 23. We use fixed-sequence testing because the FDR is a nearly monotone risk. In practice, we also wish to use the HB p-value, which is stronger than the simple Hoeffding p-value in Figure 23. The result of this procedure on the MS-COCO image dataset is in Figure 22.

B.2 Simultaneous Guarantees on OOD Detection and Coverage

In our next example, we perform classification with two goals:

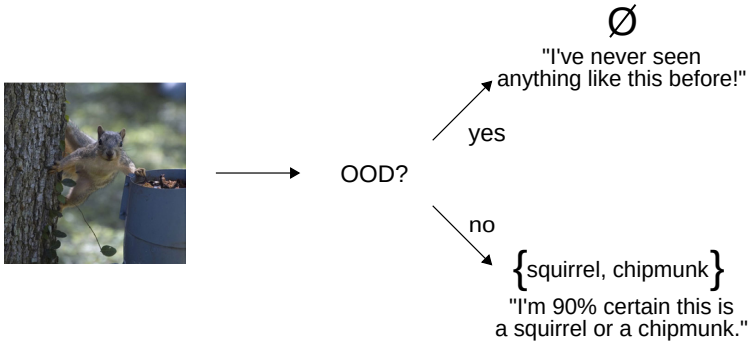
1. Flag *out-of-distribution* (OOD) inputs without too many false flags.
2. If an input is deemed *in-distribution* (In-D), output a prediction set that contains the true class with high probability.

Part of the purpose of this example is to teach the reader how to deal with multiple risk functions (one of which is a conditional risk) and a multi-dimensional parameter λ.

Our setup requires two different models. The first, $\text{OOD}(x)$, outputs a scalar that should be larger when the input is OOD. The second, $\hat{f}(x)_y$, estimates the probability that input x is of class y; for example, $\hat{f}(x)$ could represent the softmax outputs of a neural net. Similarly, the construction of $\mathcal{T}_\lambda(x)$ has two substeps, each of which uses a different model. In our first substep, when $\text{OOD}(x)$ becomes sufficiently large, exceeding λ_1, we flag the example as OOD by outputting $\emptyset$. Otherwise, we essentially use the APS method from Section 2.1 to form prediction sets. We precisely describe this procedure below:

$$\mathcal{T}_\lambda(x) = \begin{cases} \emptyset & \text{OOD}(x) > \lambda_1 \\ \{\pi_1(x), ..., \pi_K(x)\} & \text{else,} \end{cases}$$

where $K = \inf\{k : \sum_{j=1}^{k} \hat{f}(x)_{\pi_j(x)} > \lambda_2\}$ and $\pi(x)$ sorts $\hat{f}(x)$ from greatest to least. We usually take $\Lambda = \{0, 1/N, 2/N, ..., 1\}^2$, i.e., we discretize the box $[0,1] \times [0,1]$ into N^2 smaller boxes, with $N \approx 1000$. The intuition of $\mathcal{T}_\lambda(x)$ is very simple. If the example is sufficiently atypical, we give up. Otherwise, we create a prediction set using a procedure similar to (but not identical to) conformal prediction.



Along the same lines, we control two risk functions simultaneously,

$$R_1(\lambda) = \mathbb{P}\left(\mathcal{T}_\lambda(X_{\text{test}}) = \emptyset\right) \text{ and } R_2(\lambda) = \mathbb{P}\left(Y_{\text{test}} \notin \mathcal{T}_\lambda(X_{\text{test}}) \,\middle|\, \mathcal{T}_\lambda(X_{\text{test}}) \neq \emptyset\right).$$

The first risk function R_1 is the probability of a false flag, and the second risk function R_2 is the coverage conditionally on being deemed in-distribution. The user must define risk-tolerances for each, so α is a two-vector, where α_1 determines the desired fraction of false flags and α_2 determines the desired miscoverage rate. Setting $\alpha = (0.05, 0.1)$ will guarantee that we falsely throw out no more than 5% of in-distribution data points, and also that among the data points we claim are in-distribution, we will output a prediction set containing the correct class with 90% probability. In order to control both risks, we now need to associate a composite null hypothesis to each $\lambda \in \Lambda$. Namely, we choose

$$\mathcal{H}_\lambda : \mathcal{H}_\lambda^{(1)} \text{ or } \mathcal{H}_\lambda^{(2)},$$

where $\mathcal{H}_\lambda$ is the union of two intermediate null hypotheses,

$$\mathcal{H}_\lambda^{(1)} : R_1(\lambda) > \alpha_1 \text{ and } \mathcal{H}_\lambda^{(2)} : R_2(\lambda) > \alpha_2.$$

We summarize our setup in the below table.

```
# ood is an OOD detector, model is classifier with softmax output
lambda1s = torch.linspace(0,1,N) # Usually N ~= 1000
lambda2s = torch.linspace(0,1,N)
losses = torch.zeros((2,n,N,N)) # 2 losses, n data points, N x N lambdas
# The following loop can be massively parallelized (and GPU accelerated)
for (i,j,k) in [(i,j,k) for i in range(n) for j in range(N) for k in range(N)]:
  softmaxes = model(X[i].unsqueeze(0)).softmax(1).squeeze() # Care with dims
  cumsum = softmaxes.sort(descending=True)[0].cumsum(0)[Y[i]]
  if odd(X) > lambda1s[j]:
    losses[0,i,j,k] = 1
    continue
  losses[1,i,j,k] = int(cumsum > lambda2s[k])
risks = losses.mean(dim=1) # 2 x N x N
risks[1] = risks[1] - alpha2*risks[0]
pval1s = torch.exp(-2*n*(torch.relu(alpha1-risks[0])**2)) # Or HB p-value
pval2s = torch.exp(-2*n*(torch.relu(alpha2-risks[1])**2)) # Ditto
pvals = torch.maximum(pval1s,pval2s)
# Bonferroni can be replaced by sequential graphical test as in LTT paper
valid = torch.where(pvals <= delta/(N*N))
lambda_hat = [lambda1s[valid[0]], lambda2s[valid[1]]]
```

Figure 24: PyTorch code for simultaneously controlling the type-1 error of OOD detection and prediction set coverage.

Goal	Null hypothesis	Parameter
Do not incorrectly label too many images as OOD.	$H_\lambda^{(1)} : R_1(\lambda) > \alpha_1$	λ_1
Return a set of labels guaranteed to contain the true one.	$H_\lambda^{(2)} : R_2(\lambda) > \alpha_2$	λ_2

Having completed our setup, we can now apply LTT. The presence of multiple risks creates some wrinkles, which we will now iron out with the reader. The null hypothesis $\mathcal{H}_\lambda$ has a different structure than the ones we saw before, but we can use the same tools to test it. To start, we produce p-values for the intermediate nulls,

$$p_\lambda^{(1)} = e^{-2n\left(\alpha_1 - \widehat{R}_1(\lambda)\right)_+^2} \text{ and } p_\lambda^{(2)} = e^{-2n\left(\alpha_2 - \widehat{R}_2(\lambda)\right)_+^2},$$

where

$$\widehat{R}_1(\lambda) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\left\{\mathcal{T}_\lambda(X_i) = \emptyset\right\} \text{ and } \widehat{R}_2(\lambda) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\left\{Y_i \notin \mathcal{T}_\lambda(X_i), \mathcal{T}_\lambda(X_i) \neq \emptyset\right\} - \alpha_2 \mathbb{1}\left\{\mathcal{T}_\lambda(X_i) = \emptyset\right\}.^2$$

Since the maximum of two p-values is also a p-value (you can check this manually by verifying its super-uniformity), we can form the p-value for our union null as

$$p_\lambda = \max\left(p_\lambda^{(1)}, p_\lambda^{(2)}\right).$$

In practice, as before, we use the p-values from the HB inequality as opposed to those from Hoeffding. Then, instead of Bonferroni correction, we combine them with a less conservative form of sequential graphical testing; see [18] for these more mathematical details. For the purposes of this development, it suffices to return the Bonferroni region,

$$\widehat{\Lambda} = \left\{\lambda : p_\lambda \leq \frac{\delta}{|\Lambda|}\right\}.$$

Then, every element of $\widehat{\Lambda}$ controls both risks simultaneously. See Figure 24 for a PyTorch implementation of this procedure.

[2]The second empirical risk, $\widehat{R}_2$, looks different from a standard empirical risk because of the conditioning. In other words, not all of our calibration data points have nonempty prediction sets; see Section 4 of [18] to learn more about this point.

C Concentration Properties of the Empirical Coverage

We adopt the same notation as Section 3.

The variation in $\overline{C}$ has three components. First, n is finite. We analyzed how this leads to fluctuations in the coverage in Section 3.2. The second source of fluctuations is the finiteness of n_{val}, the size of the validation set. A small number of validation points can result in a high-variance estimate of the coverage. This makes the histogram of the C_j wider than the beta distribution above. However, as we will now show, C_j has an analytical distribution that allows us to exactly understand the histogram's expected properties.

We now examine the distribution of C_j. Because C_j is an average of indicator functions, it looks like it is a binomially distributed random variable. This is true conditionally on the calibration data, but not marginally. This is because the mean of the binomial is beta distributed; as we showed in the above analysis, $\mathbb{E}\left[C_j \middle| \{(X_{i,j}, Y_{i,j})\}_{i=1}^{n}\right] \sim \text{Beta}(n+1-l, l)$, where $(X_{i,j}, Y_{i,j})$ is the ith calibration point in the jth trial. Conveniently, binomial random variables with beta-distributed mean,

$$C_j \sim \frac{1}{n_{\text{val}}}\text{Binom}(n_{\text{val}}, \mu) \text{ where } \mu \sim \text{Beta}(n+1-l, l),$$

are called *beta-binomial* random variables. We refer to this distribution as $\text{BetaBinom}(n_{\text{val}}, n+1-l, l)$; its properties, such as moments and probability mass function, can be found in standard references.

Knowing the analytic form of the C_j allows us to directly plot its distribution. After a sufficient number of trials R, the histogram of C_j should converge almost exactly to its analytical PMF (which is only a function of α, n, and n_{val}). The plot in Figure 25 shows how the histograms should look with different values of n_{val} and large R. Code for producing these plots is also available in the aforementioned Jupyter notebook.

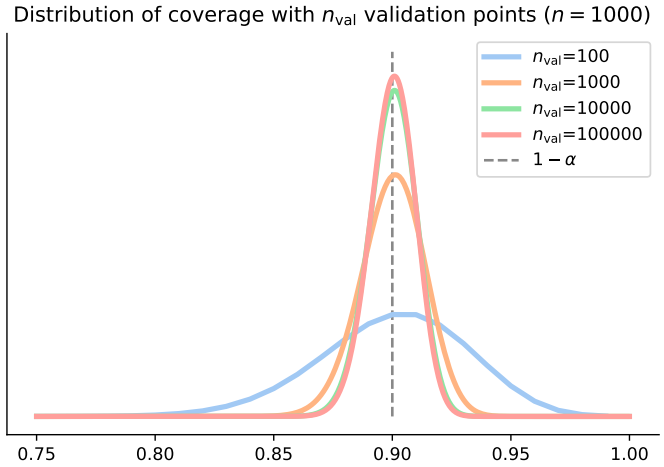


Figure 25: The distribution of empirical coverage *converges to the Beta distribution in Figure 11 as n_{val} grows. However, for small values of n_{val}, the histogram can have an inflated variance.*

The final source of fluctuations is due to the finite number of experiments, R. We have now shown that the C_j are independent beta-binomial random variables. Unfortunately, the distribution of $\overline{C}$—the mean of R independent beta-binomial random variables—does not have a closed form. However, we can simulate the distribution easily, and we visualize it for several realistic choices of R, n_{val}, and n in Figure 26.

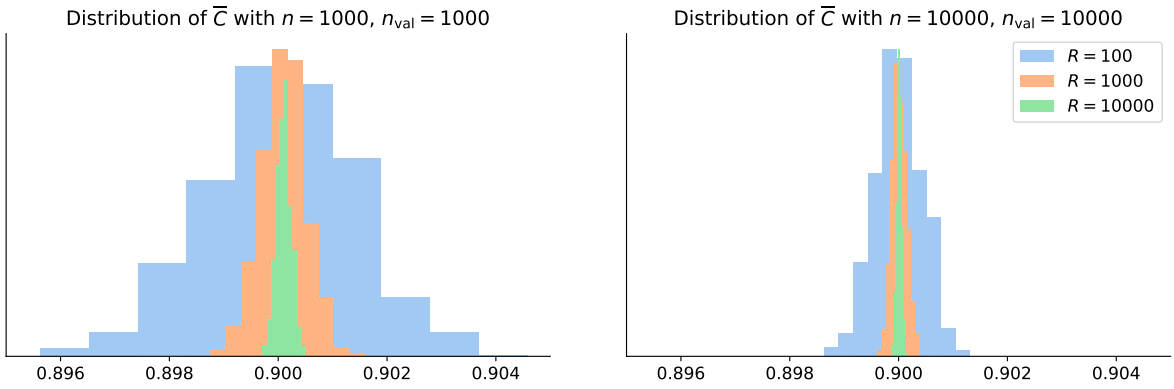


Figure 26: The distribution of average empirical coverage *over R trials with n calibration points and n_{val} validation points.*

Furthermore, we can analytically reason about the tail properties of $\overline{C}$. Since $\overline{C}$ is the average of R i.i.d. beta-binomial random variables, its mean and standard deviation are

$$\mathbb{E}\left(\overline{C}\right) = 1 - \frac{l}{n+1} \quad \text{and} \quad \sqrt{\text{Var}\left(\overline{C}\right)} = \sqrt{\frac{l(n+1-l)(n+n_{\text{val}}+1)}{n_{\text{val}}R(n+1)^2(n+2)}} = \mathcal{O}\left(\frac{1}{\sqrt{R\min(n, n_{\text{val}})}}\right).$$

The best way for a practitioner to carefully debug their procedure is to compute $\overline{C}$ empirically, and then cross-reference with Figure 26. We give code to simulate histograms with any n, R, and n_{val} in the linked notebook of Figure 26. If the simulated average empirical coverage does not align well with the coverage observed on the real data, there is likely a problem in the conformal implementation.

D Theorem and Proof: Coverage Property of Conformal Prediction

This is a standard proof of validity for split-conformal prediction first appearing in [2], but we reproduce it here for completeness. Let us begin with the lower bound.

Theorem D.1 (Conformal calibration coverage guarantee)**.** *Suppose $(X_i, Y_i)_{i=1,\ldots,n}$ and $(X_{\text{test}}, Y_{\text{test}})$ are i.i.d. Then define $\hat{q}$ as*

$$\hat{q} = \inf\left\{q : \frac{|\{i : s(X_i, Y_i) \leq q\}|}{n} \geq \frac{\lceil (n+1)(1-\alpha) \rceil}{n}\right\}.$$

and the resulting prediction sets as

$$\mathcal{C}(X) = \{y : s(X, y) \leq \hat{q}\}.$$

Then,

$$P\left(Y_{\text{test}} \in \mathcal{C}(X_{\text{test}})\right) \geq 1 - \alpha.$$

This is the same coverage property as (1) in the introduction, but written more formally. As a technical remark, the theorem also holds if the observations to satisfy the weaker condition of exchangeability; see [1]. Below, we prove the lower bound.

Proof of Theorem 1. Let $s_i = s(X_i, Y_i)$ for $i = 1, \ldots, n$ and $s_{\text{test}} = s(X_{\text{test}}, Y_{\text{test}})$. To avoid handling ties, we consider the case where the s_i are distinct with probability 1. See [25] for a proof in the general case.

Without loss of generality we assume the calibration scores are sorted so that $s_1 < \cdots < s_n$. In this case, we have that $\hat{q} = s_{\lceil (n+1)(1-\alpha) \rceil}$ when $\alpha \geq \frac{1}{n+1}$ and $\hat{q} = \infty$ otherwise. Note that in the case $\hat{q} = \infty$, $\mathcal{C}(X_{\text{test}}) = \mathcal{Y}$, so the coverage property is trivially satisfied; thus, we only have to handle the case when $\alpha \geq \frac{1}{n+1}$. We proceed by noticing the equality of the two events

$$\{Y_{\text{test}} \in \mathcal{C}(X_{\text{test}})\} = \{s_{\text{test}} \leq \hat{q}\}.$$

Combining this with the definition of $\hat{q}$ yields

$$\{Y_{\text{test}} \in \mathcal{C}(X_{\text{test}})\} = \{s_{\text{test}} \leq s_{\lceil (n+1)(1-\alpha) \rceil}\}.$$

Now comes the crucial insight. By exchangeability of the variables $(X_1, Y_1), \ldots, (X_{\text{test}}, Y_{\text{test}})$, we have

$$P(s_{\text{test}} \leq s_k) = \frac{k}{n+1}$$

for any integer k. In words, s_{test} is equally likely to fall in anywhere between the calibration points $s_1, \ldots, s_n$. Note that above, the randomness is over all variables $s_1, \ldots, s_n, s_{\text{test}}$

From here, we conclude

$$P(s_{\text{test}} \leq s_{\lceil (n+1)(1-\alpha)) \rceil}) = \frac{\lceil (n+1)(1-\alpha) \rceil}{(n+1)} \geq 1 - \alpha,$$

which implies the desired result. □

Now we will discuss the upper bound. Technically, the upper bound only holds when the distribution of the conformal score is continuous, avoiding ties. In practice, however, this condition is not important, because the user can always add a vanishing amount of random noise to the score. We will state the theorem now, and defer its proof.

Theorem D.2 (Conformal calibration upper bound)**.** *Additionally, if the scores $s_1, ..., s_n$ have a continuous joint distribution, then*

$$P\Big(Y_{\text{test}} \in \mathcal{C}(X_{\text{test}}, U_{\text{test}}, \hat{q})\Big) \leq 1 - \alpha + \frac{1}{n+1}.$$

Proof. See Theorem 2.2 of [83]. □