PNAS RESEARCH ARTICLE | STATISTICS OPEN ACCESS

Conformal prediction under feedback covariate shift for biomolecular design

Clara Fannjiang[a], Stephen Bates[b], Anastasios N. Angelopoulos[a], Jennifer Listgarten[a,c], and Michael I. Jordan[a,b,1]

Contributed by Michael I. Jordan; received March 15, 2022; accepted June 20, 2022; reviewed by Ryan Adams and Jing Lei

Many applications of machine-learning methods involve an iterative protocol in which data are collected, a model is trained, and then outputs of that model are used to choose what data to consider next. For example, a data-driven approach for designing proteins is to train a regression model to predict the fitness of protein sequences and then use it to propose new sequences believed to exhibit greater fitness than observed in the training data. Since validating designed sequences in the wet laboratory is typically costly, it is important to quantify the uncertainty in the model's predictions. This is challenging because of a characteristic type of distribution shift between the training and test data that arises in the design setting—one in which the training and test data are statistically dependent, as the latter is chosen based on the former. Consequently, the model's error on the test data—that is, the designed sequences—has an unknown and possibly complex relationship with its error on the training data. We introduce a method to construct confidence sets for predictions in such settings, which account for the dependence between the training and test data. The confidence sets we construct have finite-sample guarantees that hold for any regression model, even when it is used to choose the test-time input distribution. As a motivating use case, we use real datasets to demonstrate how our method quantifies uncertainty for the predicted fitness of designed proteins and can therefore be used to select design algorithms that achieve acceptable tradeoffs between high predicted fitness and low predictive uncertainty.

uncertainty quantification | protein engineering | conformal prediction | machine learning

Significance

An increasingly high-impact application of machine learning in scientific discovery is its use in the design of novel objects with desired properties, such as the design of proteins, small molecules, and materials. Although a variety of algorithms have been developed for this purpose, it remains unclear when practitioners can trust the predictions made by learned models for designed objects, since design algorithms induce a distinctive shift between the training and test data distributions. We propose a method that provides confidence sets for designed objects, which we show satisfy finite-sample guarantees of statistical validity, for any design algorithm involving any learned regression model. Our work enables more trustworthy use of machine learning for design.

1. Uncertainty Quantification under Feedback Loops

Consider a protein engineer who is interested in designing a protein with high fitness—some real-valued measure of its desirability, such as fluorescence or therapeutic efficacy. The engineer has a dataset of various protein sequences, denoted X_i, labeled with experimental measurements of their fitnesses, denoted Y_i, for $i = 1, \ldots, n$. The design problem is to propose a novel sequence, X_{test}, that has higher fitness, Y_{test}, than any of these. To this end, the engineer trains a regression model on the dataset and then identifies a novel sequence that the model predicts to be more fit than the training sequences. Can the engineer trust the model's prediction for the designed sequence?

This is an important question to answer, not just for the protein design problem just described, but for any deployment of machine learning where the test data depend on the training data. More broadly, settings ranging from Bayesian optimization to active learning to strategic classification involve feedback loops in which the learned model and data influence each other in turn. As feedback loops violate the standard assumptions of machine-learning algorithms, we must be able to diagnose when a model's predictions can and cannot be trusted in their presence.

In this work, we address the problem of uncertainty quantification when the training and test data exhibit a type of dependence that we call feedback covariate shift (FCS). A joint distribution of training and test data falls under FCS if it satisfies two conditions (Fig. 1). First, the test input, X_{test}, is selected based on independently and identically distributed (i.i.d.) training data, $(X_1, Y_1), \ldots, (X_n, Y_n)$. That is, the distribution of X_{test} is a function of the training data. Second, $P_{Y|X}$, the ground-truth distribution of the label, Y, given any input, X, does not change between the training and test data distributions. For example, returning to the example of protein design, the training data are used to select the designed protein, X_{test}; the distribution of X_{test} is determined by some optimization algorithm that calls the regression model to design the protein. However, since the fitness of any given sequence is some property dictated by nature, $P_{Y|X}$ stays fixed. Representative examples of FCS are algorithms that use predictive models to choose the test input distribution, such as in:

- The design of proteins, small molecules, and materials with favorable properties.

Author affiliations: [a]Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720; [b]Department of Statistics, University of California, Berkeley, CA 94720; and [c]Center for Computational Biology, University of California, Berkeley, CA 94720

Author contributions: C.F., S.B., A.N.A., J.L., and M.I.J. designed research; C.F., S.B., and A.N.A. performed research; and C.F., S.B., A.N.A., J.L., and M.I.J. wrote the paper.

Reviewers: R.A., Harvard University; and J.L., Carnegie Mellon University.

The authors declare no competing interest.

[1]To whom correspondence may be addressed. Email: jordan@cs.berkeley.edu.

This article contains supporting information online at https://www.pnas.org/lookup/suppl/doi:10.1073/pnas.2204569119/-/DCSupplemental.

Published October 18, 2022.

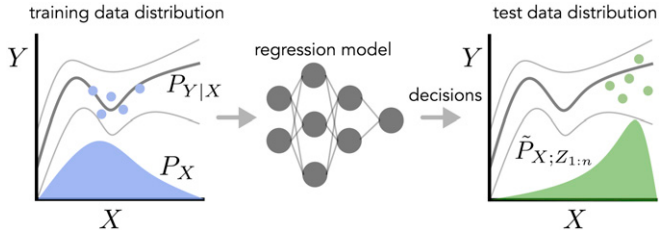


Fig. 1. Illustration of feedback covariate shift. In *Left* graph, the blue distribution represents the training input distribution, P_X. The dark gray line sandwiched by lighter gray lines represents the mean ± the SD of $P_{Y|X}$, the conditional distribution of the label given the input, which does not change between the training and test data distributions (*Left* and *Right* graphs, respectively). The blue dots represent training data, $Z_{1:n} = \{Z_1, \ldots, Z_n\}$ where $Z_i = (X_i, Y_i)$, which is used to fit a regression model (*Center*). Algorithms that use that trained model to make decisions, such as in design problems, active learning, and Bayesian optimization, give rise to a new test–time input distribution, $P_{X;Z_{1:n}}$ (*Right* graph, green distribution). The green dots represent test data.

- Active learning, adaptive experimental design, Bayesian optimization, and machine-learning-guided scientific discovery.

We anchor our discussion and experiments by focusing on protein design problems. However, the methods and insights developed herein are applicable to a variety of FCS problems.

A. Quantifying Uncertainty with Valid Confidence Sets. Given a regression model of interest, μ, we quantify its uncertainty on an input with a confidence set. A confidence set is a function, $C : \mathcal{X} \to 2^{\mathbb{R}}$, that maps a point from some input space, $\mathcal{X}$, to a set of real values that the model considers to be plausible labels.* Informally, we examine the model's error on the training data to quantify its uncertainty about the label, Y_{test}, of an input, X_{test}. Formally, using the notation $Z_i = (X_i, Y_i), i = 1, \ldots, n$ and $Z_{\text{test}} = (X_{\text{test}}, Y_{\text{test}})$, our goal is to construct confidence sets that have the frequentist statistical property known as coverage.

Definition 1. *Consider data points from some joint distribution,* $(Z_1, \ldots, Z_n, Z_{\text{test}}) \sim \mathcal{P}$. *Given a miscoverage level,* $\alpha \in (0, 1)$, *a confidence set,* $C : \mathcal{X} \to 2^{\mathbb{R}}$, *which may depend on* $Z_1, \ldots, Z_n$, *provides* coverage *under* $\mathcal{P}$ *if*

$$\mathbb{P}\left(Y_{\text{test}} \in C(X_{\text{test}})\right) \geq 1 - \alpha, \qquad \textbf{[1]}$$

where the probability is over all $n + 1$ *data points,* $(Z_1, \ldots, Z_n, Z_{\text{test}}) \sim \mathcal{P}$.

There are three important aspects of this definition. First, coverage is with respect to a particular joint distribution of the training and test data, $\mathcal{P}$, as the probability statement in Eq. **1** is over random draws of all $n + 1$ data points. That is, if one draws $(Z_1, \ldots, Z_n, Z_{\text{test}}) \sim \mathcal{P}$ and constructs the confidence set for X_{test} based on a regression model fit to $(Z_1, \ldots, Z_n)$, then the confidence set contains the true test label, Y_{test}, a fraction of $1 - \alpha$ of the time. In this work, $\mathcal{P}$ can be any distribution captured by FCS, as we describe later in more detail.

Second, note that Eq. **1** is a finite-sample statement: It holds for any number of training data points, n. Finally, coverage is a marginal probability statement, which averages over all the randomness in the training and test data; it is not a statement about conditional probabilities, such as $\mathbb{P}(Y_{\text{test}} \in C(X_{\text{test}}) \mid X_{\text{test}})$. We will call a family of confidence sets, C_α, indexed by the miscoverage level, $\alpha \in (0, 1)$, valid if they provide coverage for all $\alpha \in (0, 1)$.

When the training and test data are exchangeable (e.g., independently and identically distributed), conformal prediction is an approach for constructing valid confidence sets for any regression model (1–3). Although recent work has extended the methodology to certain forms of distribution shift (4–8), to our knowledge no existing approach can produce valid confidence sets when the test data depend on the training data. Here, we generalize conformal prediction to the FCS setting, enabling uncertainty quantification under this prevalent type of dependence between training and test data.

B. Our Contributions. First, we formalize the concept of feedback covariate shift, which describes a type of distribution shift that emerges under feedback loops between learned models and the data they operate on. Second, we introduce a generalization of conformal prediction that produces valid confidence sets under feedback covariate shift for any regression model. We also introduce randomized versions of these confidence sets that achieve a stronger property called exact coverage. Finally, we demonstrate the use of our method to quantify uncertainty for the predicted fitness of designed proteins, using several real datasets.

We recommend using our method for design algorithm selection, as it enables practitioners to identify settings of algorithm hyperparameters that achieve acceptable tradeoffs between high predictions and low predictive uncertainty.

C. Prior Work. Our study investigates uncertainty quantification in a setting that brings together the well-studied concept of covariate shift (9–12) with feedback between learned models and data distributions, a widespread phenomenon in real-world deployments of machine learning (13, 14). Indeed, beyond the design problem, feedback covariate shift is one way of describing and generalizing the dependence between data at successive iterations of active learning, adaptive experimental design, and Bayesian optimization.

Our work builds upon conformal prediction, a framework for constructing confidence sets that satisfy the finite-sample coverage property in Eq. **1** for arbitrary model classes (2, 15, 16). Although originally based on the premise of exchangeable (e.g., independently and identically distributed) training and test data, the framework has since been generalized to handle various forms of distribution shift, including covariate shift (4, 7), label shift (8), arbitrary distribution shifts in an online setting (6), and test distributions that are nearby the training distribution (5). Conformal approaches have also been used to detect distribution shift (17–23).

We call particular attention to the work of Tibshirani et al. (4) on conformal prediction in the context of covariate shift, whose technical machinery we adapt to generalize conformal prediction to feedback covariate shift. In covariate shift, the training and test input distributions differ, but, critically, the training and test data are still independent; we henceforth refer to this setting as standard covariate shift. The chief innovation of our work is to formalize and address a ubiquitous type of dependence between training and test data that is absent from standard covariate shift and, to the best of our knowledge, absent from any other form of distribution shift to which conformal approaches have been generalized.

For the design problem, in which a regression model is used to propose new inputs—such as a protein with desired properties—it is important to consider the predictive uncertainty of the designed inputs, so that we do not enter "pathological" regions

*We use the term confidence set to refer to both this function and the output of this function for a particular input; the distinction will be clear from the context.

of the input space where the model's predictions are desirable but untrustworthy (24, 25). Gaussian process regression (GPR) models are popular tools for addressing this issue, and algorithms that leverage their posterior predictive variance (26, 27) have been used to design enzymes with enhanced thermostability and catalytic activity (28, 29). Despite these successes, it is not clear how to obtain practically meaningful theoretical guarantees for the posterior predictive variance and consequently to understand in what sense we can trust it. Similarly, ensembling strategies such as in ref. 30, which are increasingly being used to quantify uncertainty for deep neural networks (24, 25, 31, 32), as well as uncertainty estimates that are explicitly learned by deep models (33) do not come with formal guarantees. A major advantage of conformal prediction is that it can be applied to any modeling strategy and can be used to calibrate any existing uncertainty quantification approach, including those aforementioned.

2. Conformal Prediction under Feedback Covariate Shift

A. Feedback Covariate Shift. We begin by formalizing FCS, which describes a setting in which the test data depend on the training data, but the relationship between inputs and labels remains fixed.

We first set up our notation. Recall that we let $Z_i = (X_i, Y_i)$, $i = 1, \ldots, n$, denote n i.i.d. training data points comprising inputs, $X_i \in \mathcal{X}$, and labels, $Y_i \in \mathbb{R}$. Similarly, let $Z_{\text{test}} = (X_{\text{test}}, Y_{\text{test}})$ denote the test data point. We use $Z_{1:n} = \{Z_1, \ldots, Z_n\}$ to denote the multiset of the training data, in which values are unordered but multiple instances of the same value appear according to their multiplicity. We also use the shorthand $Z_{-i} = Z_{1:n} \setminus \{Z_i\}$, which is a multiset of $n-1$ values that we refer to as the ith leave-one-out training dataset.

FCS describes a class of joint distributions over $(Z_1, \ldots, Z_n, Z_{\text{test}})$ that have the dependency structure described informally in *Section 1*. Formally, we say that training and test data exhibit FCS when they can be generated according to the following three steps:

1) The training data, $(Z_1, \ldots, Z_n)$, are drawn i.i.d. from some distribution
$$X_i \overset{\text{i.i.d}}{\sim} P_X,$$
$$Y_i \sim P_{Y|X_i},\ i = 1, \ldots, n.$$
2) The realized training data induce a new input distribution over $\mathcal{X}$, denoted $\tilde{P}_{X;Z_{1:n}}$ to emphasize its dependence on the training data, $Z_{1:n}$.
3) The test input is drawn from this new input distribution, and its label is drawn from the unchanged conditional distribution
$$X_{\text{test}} \sim \tilde{P}_{X;Z_{1:n}}$$
$$Y_{\text{test}} \sim P_{Y|X_{\text{test}}}.$$

The key object in this formulation is the test input distribution, $\tilde{P}_{X;Z_{1:n}}$. Prior to collecting the training data, $Z_{1:n}$, the specific test input distribution is not yet known. The observed training data induce a distribution of test inputs, $\tilde{P}_{X;Z_{1:n}}$, that the model encounters at test time (for example, through any of the mechanisms summarized in *Section 1*).

This is an expressive framework: The object $\tilde{P}_{X;Z_{1:n}}$ can be an arbitrarily complicated mapping from a dataset of size n to an input distribution, as long as it is invariant to the order of the data points. There are no other constraints on this mapping; it need not exhibit any notion of smoothness, for example. In particular, FCS encapsulates any design algorithm that makes use of a regression model fitted to the training data, $Z_{1:n}$, to propose designed inputs.

B. Conformal Prediction for Exchangeable Data. To explain how to construct valid confidence sets under FCS, we first walk through the intuition behind conformal prediction in the setting of exchangeable data and then present the adaptation to accommodate FCS.

Score function. First, we introduce the notion of a score function, $S : (\mathcal{X} \times \mathbb{R}) \times (\mathcal{X} \times \mathbb{R})^m \to \mathbb{R}$, which is an engineering choice that quantifies how well a given data point "conforms" to a multiset of m data points, in the sense of evaluating whether the data point comes from the same conditional distribution, $P_{Y|X}$, as the data points in the multiset.[†] A representative example is the residual score function, $S((X, Y), D) = |Y - \mu_D(X)|$, where D is a multiset of m data points and μ_D is a regression model trained on D. A large residual signifies a data point that the model cannot easily predict, which suggests it does not obey the input-label relationship present in the training data.

More generally, we can choose the score to be any notion of uncertainty of a trained model on the point (X, Y), heuristic or otherwise, such as the posterior predictive variance of a Gaussian process regression model (28, 29), the variance of the predictions from an ensemble of neural networks (16, 30–32), uncertainty estimates learned by deep models (34), or even the outputs of other calibration procedures (35). Regardless of the choice of the score function, conformal prediction produces valid confidence sets; however, the particular choice of score function will determine the size, and therefore informativeness, of the resulting sets. Roughly speaking, a score function that better reflects the likelihood of observing the given point, (X, Y), under the true conditional distribution that governs D, $P_{Y|X}$, results in smaller valid confidence sets.

Imitating exchangeable scores. At a high level, conformal prediction is based on the observation that when the training and test data are exchangeable, their scores are also exchangeable. More concretely, assume we use the residual score function, $S((X, Y), D) = |Y - \mu_D(X)|$, for some regression model class. Now imagine that we know the label, Y_{test}, for the test input, X_{test}. For each of the $n+1$ training and test data points, $(Z_1, \ldots, Z_n, Z_{\text{test}})$, we can compute the score using a regression model trained on the remaining n data points; the resulting $n+1$ scores are exchangeable.

In reality, of course, we do not know the true label of the test input. However, this key property—that the scores of exchangeable data yield exchangeable scores—enables us to construct valid confidence sets by including all "candidate" values of the test label, $y \in \mathbb{R}$, that yield scores for the $n+1$ data points (the training data points along with the candidate test data point, (X_{test}, y)) that appear to be exchangeable. For a given candidate label, the conformal approach assesses whether or not this is true by comparing the score of the candidate test data point to an appropriately chosen quantile of the training data scores.

C. Conformal Prediction under FCS. When the training and test data are under FCS, their scores are no longer exchangeable, since the training and test inputs are neither independent nor from the same distribution. Our solution to this problem is to weight

[†]Since the second argument is a multiset of data points, the score function must be invariant to the order of these data points. For example, when using the residual as the score, the regression model must be trained in a way that is agnostic to the order of the data points.

each training and test data point to take into account these two factors. Thereafter, we can proceed with the conformal approach of including all candidate labels such that the (weighted) candidate test data point is sufficiently similar to the (weighted) training data points. Toward this end, we introduce two quantities: 1) a likelihood-ratio function, which will be used to define the weights, and 2) the quantile of a distribution, which will be used to assess whether a candidate test data point conforms to the training data.

The likelihood-ratio function, which depends on a multiset of data points, D, is given by

$$v(X; D) = \frac{\tilde{p}_{X;D}(X)}{p_X(X)}, \quad [2]$$

where $\tilde{p}_{X;D}$ and p_X denote the densities of the test and training input distributions, respectively, and the test input distribution is the particular one indexed by the dataset, D.

This quantity is the ratio of the likelihoods under these two distributions and, as such, is reminiscent of weights used to modify various statistical procedures to accommodate standard covariate shift (4, 10, 11). What distinguishes its use here is that our particular likelihood ratio is indexed by a multiset and depends on which data point is being evaluated as well as the candidate label, as will become clear shortly.

Consider a discrete distribution with probability masses $p_1, \ldots, p_n$ located at support points $s_1, \ldots, s_n$, respectively, where $s_i \in \mathbb{R}$ and $p_i \geq 0, \sum_i p_i = 1$. We define the β-quantile of this distribution as

$$\text{QUANTILE}_\beta\left(\sum_{i=1}^{n} p_i\, \delta_{s_i}\right) = \inf\left\{s : \sum_{i:s_i \leq s} p_i \geq \beta\right\},$$

where δ_{s_i} is a unit point mass at s_i.

We now define the confidence set. For any score function, S; any miscoverage level, $\alpha \in (0, 1)$; and any test input, $X_{\text{test}} \in \mathcal{X}$, define the full conformal confidence set as

$$C_\alpha(X_{\text{test}}) = \left\{ y \in \mathbb{R} : S_{n+1}(X_{\text{test}}, y) \leq \text{QUANTILE}_{1-\alpha}\left(\sum_{i=1}^{n+1} w_i^y(X_{\text{test}})\, \delta_{S_i(X_{\text{test}}, y)}\right)\right\}, \quad [3]$$

where

$$S_i(X_{\text{test}}, y) = S(Z_i, Z_{-i} \cup \{(X_{\text{test}}, y)\}),\ i = 1, \ldots, n,$$
$$S_{n+1}(X_{\text{test}}, y) = S((X_{\text{test}}, y), Z_{1:n}),$$

which are the scores for each of the training and candidate test data points, when compared to the remaining n data points, and the weights for these scores are given by

$$w_i^y(X_{\text{test}}) \propto v(X_i; Z_{-i} \cup \{(X_{\text{test}}, y)\}),\ i = 1, \ldots, n,$$
$$w_{n+1}^y(X_{\text{test}}) \propto v(X_{\text{test}}; Z_{1:n}), \quad [4]$$

which are normalized such that $\sum_{i=1}^{n+1} w_i^y(X_{\text{test}}) = 1$.

In words, the confidence set in Eq. **3** includes all real values, $y \in \mathbb{R}$, such that the candidate test data point, (X_{test}, y), has a score that is sufficiently similar to the scores of the training data. Specifically, the score of the candidate test data point needs to be smaller than the $(1-\alpha)$-quantile of the weighted scores of all $n+1$ data points (the n training data points as well as the candidate test data point), where the ith data point is weighted by $w_i^y(X_{\text{test}})$.

Our main result is that this confidence set provides coverage under FCS (see *SI Appendix*, section S1.A for the proof).

Theorem 1. *Suppose data are generated under feedback covariate shift and assume $\tilde{P}_{X;D}$ is absolutely continuous with respect to P_X for all possible values of D. Then, for any miscoverage level, $\alpha \in (0, 1)$, the full conformal confidence set, C_α, in Eq.* **3** *satisfies the coverage property in* Eq. **1**; *namely*, $\mathbb{P}(Y_{\text{test}} \in C_\alpha(X_{\text{test}})) \geq 1 - \alpha$.

Since we can supply any domain-specific notion of uncertainty as the score function, this result implies we can interpret the condition in Eq. **3** as a calibration of the provided score function that guarantees coverage. That is, our conformal approach can complement any existing uncertainty quantification method by endowing it with coverage under FCS.

We note that although *Theorem 1* provides a lower bound on the probability $\mathbb{P}(Y_{\text{test}} \in C_\alpha(X_{\text{test}}))$, one cannot establish a corresponding upper bound without further assumptions on the training and test input distributions. However, by introducing randomization to the β-quantile, we can construct a randomized version of the confidence set, $C_\alpha^{\text{rand}}(X_{\text{test}})$, that is not conservative and satisfies $\mathbb{P}(Y_{\text{test}} \in C_\alpha^{\text{rand}}(X_{\text{test}})) = 1 - \alpha$, a property called exact coverage. See *SI Appendix*, section S1.B for details.

Estimating confidence sets in practice. In practice, one cannot check all possible candidate labels, $y \in \mathbb{R}$, to construct a confidence set. Instead, as done in previous work on conformal prediction, we estimate $C_\alpha(X_{\text{test}})$ by defining a finite grid of candidate labels, $\mathcal{Y} \subset \mathbb{R}$, and checking the condition in Eq. **3** for all $y \in \mathcal{Y}$. *Algorithm 1* outlines a generic recipe for computing $C_\alpha(X_{\text{test}})$ for a given test input; see *Section 2.D* for important special cases in which $C_\alpha(X_{\text{test}})$ can be computed more efficiently.

Algorithm 1 Pseudocode for approximately computing $C_\alpha(X_{\text{test}})$

Input: Training data, $(Z_1, \ldots, Z_n)$, where $Z_i = (X_i, Y_i)$; test input, X_{test}; finite grid of candidate labels, $\mathcal{Y} \subset \mathbb{R}$; likelihood ratio function subroutine, $v(\cdot; \cdot)$; and score function subroutine $S(\cdot, \cdot)$.
Output: Confidence set, $C_\alpha(X_{\text{test}}) \subset \mathcal{Y}$.

$C_\alpha(X_{\text{test}}) \leftarrow \emptyset$
Compute $v(X_{\text{test}}; Z_{1:n})$
for $y \in \mathcal{Y}$ **do**
  for $i = 1, \ldots, n$ **do**
    Compute $S_i(X_{\text{test}}, y)$ and $v(X_i; Z_{-i} \cup \{(X_{\text{test}}, y)\})$
  Compute $S_{n+1}(X_{\text{test}}, y)$
  for $i = 1, \ldots, n+1$ **do**
    Normalize $w_i^y(X_{\text{test}})$ according to Eq. (4)
  $q_y \leftarrow \text{QUANTILE}_{1-\alpha}\left(\sum_{i=1}^{n+1} w_i^y(X_{\text{test}})\, \delta_{S_i(X_{\text{test}}, y)}\right)$
  if $S_{n+1}(X_{\text{test}}, y) \leq q_y$ **then**
    $C_\alpha(X_{\text{test}}) \leftarrow C_\alpha(X_{\text{test}}) \cup \{y\}$

Relationship with exchangeable and standard covariate shift settings. The weights assigned to each score, $w_i^y(X_{\text{test}})$ in Eq. **4**, are the distinguishing factor between the confidence sets constructed by conformal approaches for the exchangeable, standard covariate shift, and FCS settings. When the training and test data are exchangeable, these weights are simply $1/(n+1)$. To accommodate standard covariate shift, where the training and test data are independent, these weights are also normalized likelihood ratios—but, importantly, the test input distribution in the numerator is fixed, rather than data dependent as in the FCS setting (4). That is, the weights are defined using one fixed likelihood-ratio function,

$v(\cdot) = \tilde{p}_X(\cdot)/p_X(\cdot)$, where $\tilde{p}_X$ is the density of the single test input distribution under consideration.

In contrast, under FCS, observe that the likelihood ratio that is evaluated in Eq. **4**, $v(\cdot; D)$, is different for each of the $n+1$ training and candidate test data points and for each candidate label, $y \in \mathbb{R}$. To weight the ith training score, we evaluate the likelihood ratio of X_i where the test input distribution is the one induced by $Z_{-i} \cup \{(X_{\text{test}}, y)\}$,

$$v(X_i; Z_{-i} \cup \{(X_{\text{test}}, y)\}) = \frac{\tilde{p}_{X;Z_{-i}\cup\{(X_{\text{test}},y)\}}(X_i)}{p_X(X_i)}.$$

That is, the weights under FCS take into account not just a single test input distribution, but every test input distribution that can be induced when we treat a leave-one-out training dataset combined with a candidate test data point, $Z_{-i} \cup \{(X_{\text{test}}, y)\}$, as the training data.

To further appreciate the relationship between the standard and feedback covariate shift settings, consider the weights used in the standard covariate shift approach if we treat $P_{X;Z_{1:n}}$ as the test input distribution. The extent to which $P_{X;Z_{1:n}}$ differs from $P_{X;Z_{-i}\cup\{(X_{\text{test}},y)\}}$, for any $i = 1, \ldots, n$ and $y \in \mathbb{R}$, determines the extent to which the weights used under standard covariate shift deviate from those used under FCS. In other words, since $Z_{1:n}$ and $Z_{-i} \cup \{(X_{\text{test}}, y\}$ differ in exactly one data point, the similarity between the standard covariate shift and FCS weights depends on the "smoothness" of the mapping from D to $\tilde{P}_{X;D}$.

Input distributions are known in the design problem. The design problem is a unique setting in which we have control over the data-dependent test input distribution, $P_{X;D}$, since we choose the procedure used to design an input. In the simplest case, some design procedures sample from a distribution whose form is explicitly chosen, such as an energy-based model whose energy function is proportional to the predictions from a trained regression model (36) or a model whose parameters are set by solving an optimization problem (e.g., the training of a generative model) (24, 25, 37–43). In either setting, we know the exact form of the test input distribution, which also absolves the need for density estimation.

In other cases, the design procedure involves iteratively applying a gradient to, or otherwise locally modifying, an initial input to produce a designed input (44–49). Due to randomness in either the initial input or the local modification rule, such procedures implicitly result in some distribution of test inputs. Although we do not have access to its explicit form, knowledge of the design procedure can enable us to estimate it much more readily than in a naive density estimation setting. For example, we can simulate the design procedure as many times as needed to sufficiently estimate the resulting density, whereas in density estimation in general, we cannot control how many test inputs we can access.

The training input distribution, P_X, is also often known explicitly. In protein design problems, for example, training sequences are often generated by introducing random substitutions to a single wild-type sequence (24, 36, 49), by recombining segments of several "parent" sequences (28, 29, 50, 51), or by independently sampling the amino acid at each position from a known distribution (43, 52). Conveniently, we can then compute the weights in Eq. **4** exactly without introducing approximation error due to density ratio estimation.

Finally, we note that, by construction, the design problem tends to result in test input distributions that place considerable probability mass on regions where the training input distribution does not. The farther the test distribution is from the training distribution in this regard, the larger the resulting weights on candidate test points, and the larger the confidence set in Eq. **3** will tend to be. This phenomenon agrees with our intuition about epistemic uncertainty: We should have more uncertainty—that is, larger confidence sets—in regions of input space where there are fewer training data.

D. Efficient Computation of Confidence Sets under Feedback Covariate Shift. Using *Algorithm 1* to construct the full conformal confidence set, $C_\alpha(X_{\text{test}})$, requires computing the scores and weights, $S_i(X_{\text{test}}, y)$ and $w_i^y(X_{\text{test}})$, for all $i = 1, \ldots, n+1$ and all candidate labels, $y \in \mathcal{Y}$. When the dependence of $\tilde{P}_{X;D}$ on D arises from a model trained on D, then naively, we must train $(n+1) \times |\mathcal{Y}|$ models to compute these quantities. We now describe two important, practical cases in which this computational burden can be reduced to fitting $n+1$ models, removing the dependence on the number of candidate labels. In such cases, we can postprocess the outputs of these $n+1$ models to calculate all $(n+1) \times |\mathcal{Y}|$ required scores and weights (see *SI Appendix, Algorithm S2* for pseudocode); we refer to this as computing the confidence set efficiently.

In the following two examples and in our experiments, we use the residual score function, $S((X, Y), D) = |Y - \mu_D(X)|$, where μ_D is a regression model trained on the multiset D. To understand at a high level when efficient computation is possible, first let μ^y_{-i} denote the regression model trained on $Z^y_{-i} = Z_{-i} \cup \{(X_{\text{test}}, y)\}$, the ith leave-one-out training dataset combined with a candidate test data point. The scores and weights can be computed efficiently when $\mu^y_{-i}(X_i)$ is a computationally simple function of the candidate label, y, for all i—for example, a linear function of y. We discuss two such cases in detail.

Ridge regression. Suppose we fit a ridge regression model, with ridge regularization hyperparameter γ, to the training data. Then, we draw the test input vector from a distribution that places more mass on regions of $\mathcal{X}$ where the model predicts more desirable values, such as high fitness in protein design problems. Recent studies have employed this relatively simple approach to successfully design novel enzymes with enhanced catalytic efficiencies and thermostabilities (36, 50, 53).

In the ridge regression setting, the quantity $\mu^y_{-i}(X_i)$ can be written in closed form as

$$\begin{aligned}\mu^y_{-i}(X_i) &= \left[\left(\mathbf{X}^T_{-i}\mathbf{X}_{-i} + \gamma I\right)^{-1}\mathbf{X}^T_{-i}\,Y^y_{-i}\right]^T X_i \\ &= \left(\sum_{j=1}^{n-1} Y_{-i;j}\mathbf{A}_{-i;j}\right)^T X_i + (\mathbf{A}^T_{-i;n}X_i)y,\end{aligned} \qquad [5]$$

where the rows of the matrix $\mathbf{X}_{-i} \in \mathbb{R}^{n\times p}$ are the input vectors in Z^y_{-i}, $Y^y_{-i} = (Y_{-i}, y) \in \mathbb{R}^n$ contains the labels in Z^y_{-i}, the matrix $\mathbf{A}_{-i} \in \mathbb{R}^{n\times p}$ is defined as $\mathbf{A}_{-i} = \left(\mathbf{X}^T_{-i}\mathbf{X}_{-i} + \gamma I\right)^{-1}\mathbf{X}^T_{-i}$, $\mathbf{A}_{-i;j}$ denotes the jth column of A_{-i}, and $Y_{-i;j}$ denotes the jth element of Y_{-i}.

Note that the expression in Eq. **5** is a linear function of the candidate label, y. Consequently, as formalized by *SI Appendix, Algorithm S2*, we first compute and store the slopes and intercepts of these linear functions for all i, which can be calculated as byproducts of fitting $n+1$ ridge regression models. Using these parameters, we can then compute $\mu^y_{-i}(X_i)$ for all candidate labels, $y \in \mathcal{Y}$, by simply evaluating a linear function of y instead of retraining a regression model on Z^y_{-i}. Altogether, beyond fitting $n+1$ ridge regression models, *SI Appendix, Algorithm S2* requires $O(n \cdot p \cdot |\mathcal{Y}|)$ additional floating-point operations to compute the scores and weights for all the candidate labels, the bulk of which can be implemented as

one outer product between an n vector and a $|\mathcal{Y}|$ vector and one Kronecker product between an $(n \times p)$ matrix and a $|\mathcal{Y}|$ vector.

Gaussian process regression. Similarly, suppose we fit a Gaussian process regression model to the training data. We then select a test input vector according to a likelihood that is a function of the mean and variance of the model's prediction; such functions are referred to as acquisition functions in the Bayesian optimization literature.

For a linear kernel, the expression for the mean prediction, $\mu^{y}_{-i}(X_i)$, is the same as for ridge regression (Eq. **5**). For arbitrary kernels, the expression can be generalized and remains a linear function of y (see *SI Appendix*, section S2.B for details). We can therefore mimic the computations described for the ridge regression case to compute the scores and weights efficiently.

E. Data Splitting. For settings with abundant training data, or model classes that do not afford efficient computations of the scores and weights, one can turn to data splitting to construct valid confidence sets. To do so, we first randomly partition the labeled data into disjoint training and calibration sets. Next, we use the training data to fit a regression model, which induces a test input distribution. If we condition on the training data, thereby treating the regression model as fixed, we have a setting in which 1) the calibration and test data are drawn from different input distributions, but 2) are independent (even though the test and training data are not). Thus, data splitting returns us to the setting of standard covariate shift, under which we can use the data-splitting approach in ref. 4 to construct valid split conformal confidence intervals (*SI Appendix*, section S1.C).

We also introduce randomized data-splitting approaches that give exact coverage; see *SI Appendix*, section S1.D for details.

3. Experiments with Protein Design

To demonstrate practical applications of our work, we turn to examples of uncertainty quantification for designed proteins. Given a fitness function of interest, such as fluorescence, a typical goal of protein design is to seek a protein with high fitness—in particular, higher than we have observed in known proteins. [‡] Historically, this has been accomplished through several iterations of expensive, time-consuming experiments. Recently, efforts have been made to augment such approaches with machine-learning–based strategies; see reviews by Yang et al. (54), Sinai and Kelsic (55), and Wu et al. (56) and references therein. For example, one might train a regression model on protein sequences with experimentally measured fitnesses and then use an optimization algorithm or fit a generative model that leverages that regression model to propose promising new proteins (24, 28, 29, 36, 43, 46, 51, 53, 57–59). Special attention has been given to the single-shot case in which we are given just a single batch of training data, due to its obvious practical convenience.

The use of regression models for design involves balancing 1) the desire to explore regions of input space far from the training inputs, to find new desirable inputs, with 2) the need to stay close enough to the training inputs that we can trust the regression model. As such, estimating predictive uncertainty in this setting is important. Furthermore, the training and designed data are described by feedback covariate shift: Since the fitness is some quantity dictated by nature, the conditional distribution of fitness given any sequence stays fixed, but the distribution of designed sequences is chosen based on a trained regression model.[§]

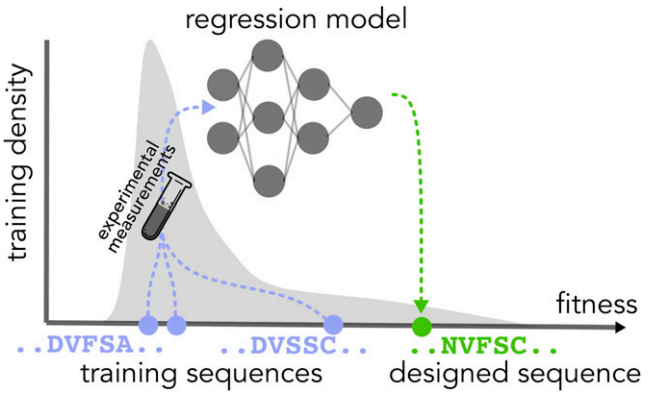


Fig. 2. Illustration of single-shot protein design. The gray distribution represents the distribution of fitnesses under the training sequence distribution. The blue circles represent the fitnesses of three training sequences, and the goal is to propose a sequence with even higher fitness. To that end, we fit a regression model to the training sequences labeled with experimental measurements of their fitnesses and then deploy some design procedure that uses that trained model to propose a new sequence believed to have a higher fitness (green circle).

Our experimental protocol is as follows: Given training data consisting of protein sequences labeled with experimental measurements of their fitnesses, we fit a regression model, then sample test sequences (representing designed proteins) according to design algorithms used in recent work (36, 43) (Fig. 2). We then construct confidence sets with guaranteed coverage for the designed proteins and examine various characteristics of those sets to evaluate the utility of our approach. In particular, we show how our method can be used to select design algorithm hyperparameters that achieve acceptable tradeoffs between high predicted fitness and low predictive uncertainty for the designed proteins. Code reproducing these experiments is available at https://github.com/clarafy/conformal-for-design.

A. Design Experiments Using Combinatorially Complete Fluorescence Datasets. The challenge when evaluating in silico design methods is that in general, we do not have labels for the designed sequences. One workaround, which we take here, is to make use of combinatorially complete protein datasets (57, 59–61), in which a small number of fixed positions are selected from some wild-type sequence, and all possible variants of the wild type that vary in those selected positions are measured experimentally. Such datasets enable us to simulate protein design problems where we always have labels for the designed sequences. In particular, we can use a small subset of the data for training and then deploy a design procedure that proposes novel proteins (restricted to being variants of the wild type at the selected positions), for which we have labels.

We used data of this kind from Poelwijk et al. (60), which focused on two parent fluorescent proteins that differ at exactly 13 positions in their sequences and are identical at every other position. All $2^{13} = 8,192$ sequences that have the amino acid of either parent at those 13 sites (and whose remaining positions are identical to the parents) were experimentally labeled with a measurement of brightness at both a "red" wavelength and a "blue" wavelength, resulting in combinatorially complete datasets for two

[‡] We use the term fitness function to refer to a particular property that can be exhibited by proteins, while the fitness of a protein refers to the extent to which it exhibits that property.

[§] In this section, we use "test" and "designed" interchangeably when describing data. We also sometimes say "sequence" instead of "input," but this does not imply any constraints on how the protein is represented or featurized.

different fitness functions. In particular, for both wavelengths, the label for each sequence was an enrichment score based on the ratio of its counts before and after brightness-based selection through fluorescence-activated cell sorting. The enrichment scores were then normalized so that the same score reflects comparable brightness for both wavelengths.

Finally, each time we sampled from this dataset to acquire training or designed data, as described below, we added simulated measurement noise to each label by sampling from a noise distribution estimated from the combinatorially complete dataset (see *SI Appendix*, section S3 for details). This step simulates the fact that sampling and measuring the same sequence multiple times results in different measurements.

A.1. Protocol for design experiments. Our training datasets consisted of n data points, $Z_{1:n}$, sampled uniformly at random from the combinatorially complete dataset. We used $n \in \{96, 192, 384\}$ as is typical of realistic scenarios (28, 36, 51, 57, 59). We represented each sequence as a feature vector containing all first- and second-order interaction terms between the 13 variable sites and fitted a ridge regression model, $\mu_{Z_{1:n}}(x)$, to the training data, where the regularization strength was set to 10 for $n = 96$ and 1 otherwise. Linear models of interaction terms between sequence positions have been observed to be both theoretically justified and empirically useful as models of protein fitness functions (60–62) and thus may be particularly useful for protein design, particularly with small amounts of training data.

Sampling designed sequences. Following ideas in refs. 36 and 43, we designed a protein by sampling from a sequence distribution whose log-likelihood is proportional to the prediction of the regression model:

$$\tilde{p}_{X;Z_{1:n}}(X_{\text{test}}) \propto \exp(\lambda \cdot \mu_{Z_{1:n}}(X_{\text{test}})), \qquad [6]$$

where $\lambda > 0$, the inverse temperature, is a hyperparameter. Larger values of λ result in distributions of designed sequences that are more likely to have high predicted fitnesses according to the model, but are also, for this same reason, more likely to be in regions of sequence space that are farther from the training data and over which the model is more uncertain. Analogous hyperparameters have been used in recent protein design work to control this tradeoff between exploration and exploitation (36, 39, 43, 63). We took $\lambda \in \{0, 2, 4, 6\}$ to investigate how the behavior of our confidence sets varies along this tradeoff.

Constructing confidence sets for designed sequences. For each setting of n and λ, we generated n training data points and one designed data point as just described $T = 2,000$ times. For each of these T trials, we used *SI Appendix, Algorithm S2* to construct the full conformal confidence set, $C_\alpha(X_{\text{test}})$, using a grid of real values between 0 and 2.2 spaced $\Delta = 0.02$ apart as the set of candidate labels, $\mathcal{Y}$. This range contained the ranges of fitnesses in both the blue and red combinatorially complete datasets, $[0.091, 1.608]$ and $[0.025, 1.692]$, respectively.[¶]

We used $\alpha = 0.1$ as a representative miscoverage value, corresponding to coverage of $1 - \alpha = 0.9$. We then computed the empirical coverage achieved by the confidence sets, defined as the fraction of the T trials where the true fitness of the designed protein was within half a grid spacing from some value in the confidence set; namely, $\min\{|Y_{\text{test}} - y| : y \in C_\alpha(X_{\text{test}})\} \leq \Delta/2$. Based on *Theorem 1*, assuming $\mathcal{Y}$ is both a large and a fine enough grid to encompass all possible fitness values, the expected empirical coverage is lower bounded by $1 - \alpha = 0.9$. However, there is no corresponding upper bound, so it will be of interest to examine any excess in the empirical coverage, which corresponds to the confidence sets being conservative (larger than necessary). Ideally, the empirical coverage is exactly 0.9, in which case the sizes of the confidence sets reflect the minimal predictive uncertainty we can have about the designed proteins while achieving coverage.

In our experiments, the computed confidence sets tended to comprise grid-adjacent candidate labels, suggestive of confidence intervals. As such, we hereafter refer to the width of confidence intervals, defined as the grid spacing size times the number of values in the confidence set, $\Delta \cdot |C_\alpha(X_{\text{test}})|$.

A.2. Results. Here we discuss results for the blue fluorescence dataset. Analogous results for the red fluorescence dataset are presented in *SI Appendix*, section S3.

Effect of inverse temperature. First, we examined the effect of the inverse temperature, λ, on the fitnesses of designed proteins (Fig. 3*A*). Note that $\lambda = 0$ corresponds to a uniform distribution over all sequences in the combinatorially complete dataset (i.e., the training distribution), which mostly yields label values less than 0.5. For $\lambda \geq 4$, we observe a considerable mass of designed proteins attaining fitnesses around 1.5, so these values of λ represent settings where the designed proteins are more likely to be fitter than the training proteins. This observation is consistent with the use of this and other analogous hyperparameters to tune the outcomes of design algorithms (36, 39, 43, 63) and is meant to provide an intuitive interpretation of the hyperparameter to readers unfamiliar with its use in design problems.

Empirical coverage and confidence interval widths. Despite the lack of a theoretical upper bound, the empirical coverage does not tend to exceed the theoretical lower bound of $1 - \alpha = 0.9$ by much (Fig. 3*B*), reaching at most 0.924 for $n = 96, \lambda = 6$. Loosely speaking, this observation suggests that the confidence intervals are nearly as small, and therefore as informative, as they can be while achieving coverage.

As for the widths of the confidence intervals, we observe that for any value of λ, the intervals tend to be smaller for larger amounts of training data (Fig. 3*C*). Also, for any value of n, the intervals tend to get larger as λ increases. The first phenomenon agrees with the intuition that training a model on more data should generally reduce predictive uncertainty. The second phenomenon arises because greater values of λ lead to designed sequences with higher predicted fitnesses, which the model is more uncertain about. Indeed, for $\lambda \geq 4, n \leq 192$ and $\lambda = 6, n = 384$, some confidence intervals equal the whole set of candidate labels. In these regimes, the regression model cannot glean enough information from the training data to have much certainty about the designed protein.

Comparison to standard covariate shift. Deploying full conformal prediction as prescribed for standard covariate shift (SCS) (4), a heuristic with no formal guarantees in this setting, often results in more conservative confidence sets than those produced by our method (Fig. 3). To understand when the outputs of these two methods will differ more or less, we can compare the forms of the weights that both methods introduce on the training and candidate test data points when considering a candidate label.

First, recall that for FCS and SCS, the weight assigned to the ith training score is a normalized ratio of the likelihood of X_i under a test input distribution and the training input distribution, p_X; namely,

[¶] In general, a reasonable approach for constructing a finite grid of candidate labels, $\mathcal{Y}$, is to span an interval beyond which one knows label values are impossible in practice, based on prior knowledge about the measurement technology. The presence or absence of any such value in a confidence set would not be informative to a practitioner. The size of the grid spacing, Δ, determines the resolution at which we evaluate coverage; that is, in terms of coverage, including a candidate label is equivalent to including the Δ-width interval centered at that label value. Generally, one should therefore set Δ as small as possible, subject to one's computational budget.

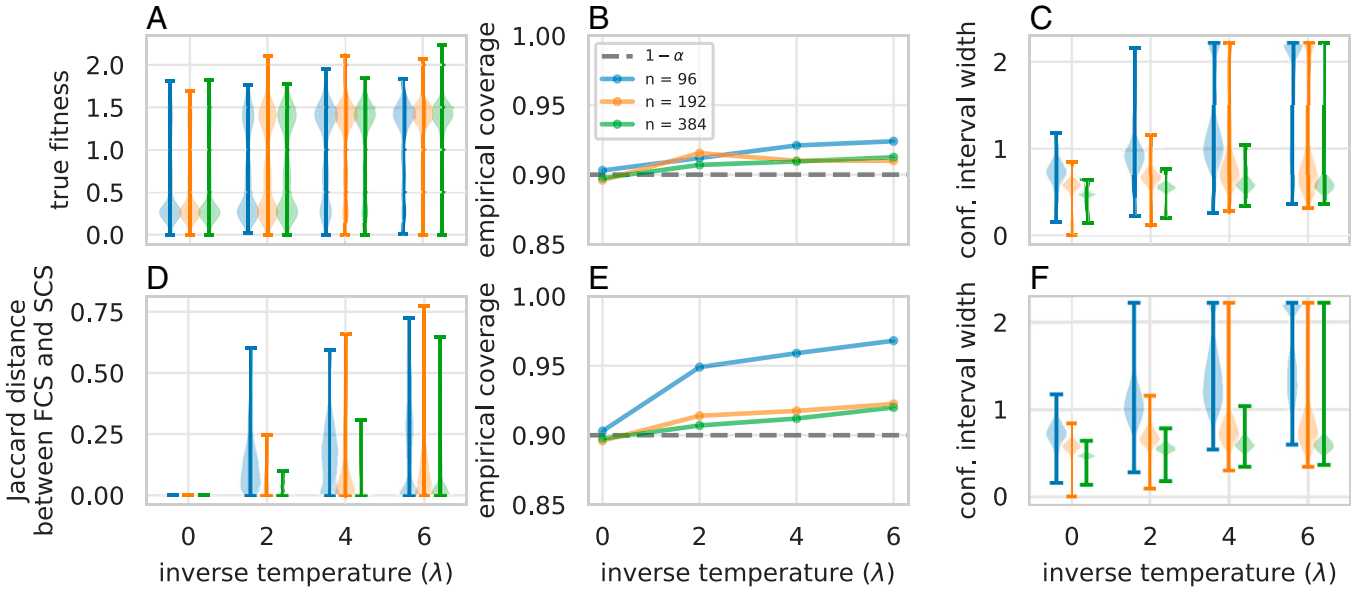


Fig. 3. Quantifying predictive uncertainty for designed proteins, using the blue fluorescence dataset. (*A*) Distributions of labels of designed proteins, for different values of the inverse temperature, λ, and different amounts of training data, n. Labels surpass the fitness range observed in the combinatorially complete dataset, [0.091, 1.608], due to additional simulated measurement noise. (*B* and *C*) Empirical coverage (*B*), compared to the theoretical lower bound of $1-\alpha = 0.9$ (dashed gray line), and (*C*) distributions of confidence interval widths achieved by full conformal prediction for feedback covariate shift (our method) over $T = 2{,}000$ trials. In *A* and *C*, the whiskers signify the minimum and maximum observed values. (*D*) Distributions of Jaccard distances between the confidence intervals produced by full conformal prediction for feedback covariate shift and standard covariate shift (4). (*E* and *F*) Same as in *B* and *C* but using full conformal prediction for standard covariate shift.

$$v(X_i; Z_{-i} \cup \{(X_{\text{test}}, y)\}) = \tilde{p}_{X;Z_{-i}\cup\{(X_{\text{test}},y)\}}(X_i)/p_X(X_i),$$
$$v(X_i) = \tilde{p}_{X;Z_{1:n}}(X_i)/p_X(X_i),$$

for FCS and SCS, respectively. For FCS, the test input distribution, $\tilde{p}_{X;Z_{-i}\cup\{(X_{\text{test}},y)\}}$, is induced by a regression model trained on $Z_{-i} \cup \{(X_{\text{test}}, y)\}$ and therefore depends on the candidate label, y, and also differs for each of the n training inputs. Consequently, for FCS the weight on the ith training score depends on the candidate label under consideration, y. In contrast, for SCS the test input distribution, $p_{X;Z_{1:n}}$, is simply the one induced by the training data, $Z_{1:n}$, and is therefore fixed for all training scores and all candidate labels.

Note, however, that the SCS and FCS weights depend on datasets, $Z_{1:n}$ and $Z_{-i} \cup \{(X_{\text{test}}, y)\}$, respectively, that differ only in a single data point: The former contains Z_i, while the latter contains (X_{test}, y). Therefore, the difference between the weights—and the resulting confidence sets—is a direct consequence of how sensitive the mapping from dataset to test input distribution, $D \to \tilde{p}_{X;D}$ (given by Eq. **6** in this setting), is to changes of a single data point in D. Roughly speaking, the less sensitive this mapping is, the more similar the FCS and SCS confidence sets will be. For example, using more training data (e.g., $n = 384$ compared to $n = 96$ for a fixed λ) or a lower inverse temperature (e.g., $\lambda = 2$ compared to $\lambda = 6$ for a fixed n) results in more similar SCS and FCS confidence sets (Fig. 3*D* and *SI Appendix*, Figs. S2*D* and S5). Similarly, using regression models with fewer features or stronger regularization also results in more similar confidence sets (*SI Appendix*, Figs. S3, S4, and S6).

One can therefore think of and use SCS confidence sets as a computationally cheaper approximation to FCS confidence sets, where the approximation is better for mappings $D \to \tilde{p}_{X;D}$ that are less sensitive to changes in D. Conversely, the extent to which SCS confidence sets are similar to FCS confidence sets will generally reflect this sensitivity. In our protein design experiments, SCS confidence sets tend to be more conservative than their FCS counterparts, where the extent of overcoverage generally increases with fewer training data, higher inverse temperature (Fig. 3*D* and *SI Appendix*, Fig. S2*D*), more complex features, and weaker regularization (*SI Appendix*, Figs. S3 and S4).

Using uncertainty quantification to set design procedure hyperparameters. As the inverse temperature, λ, in Eq. **6** varies, there is a tradeoff between the mean predicted fitness and predictive certainty for designed proteins: Both mean predicted fitness and mean confidence interval width grow as λ increases (Fig. 4*A*). To demonstrate how our method might be used to inform the design procedure itself, one can visualize this tradeoff (Fig. 4) and use it to decide on a setting of λ that achieves both a mean predicted fitness and degree of certainty that one finds acceptable, given, for example, some resource budget for evaluating designed proteins in the wet laboratory. For datasets of different fitness functions, which may be better or worse approximated by our chosen regression model class and may have different amounts of measurement noise, this tradeoff—and therefore the appropriate setting of λ—will be different (Fig. 4).

For example, protein design experiments on the red fluorescence dataset result in a less favorable tradeoff between mean predicted fitness and predictive certainty than the blue fluorescence dataset: The same amount of increase in mean predicted fitness corresponds to a greater increase in mean interval width for red compared to blue fluorescence (Fig. 4*A*). We might therefore choose a smaller value of λ when designing proteins for the former compared to the latter. Indeed, predictive uncertainty grows so quickly for red fluorescence that, for $\lambda > 2$, the empirical probability that the smallest value in the confidence interval is greater than the true fitness of a wild-type sequence decreases rather than increases (Fig. 4*B*), which suggests we may not want to set $\lambda > 2$. In contrast, if we had looked at the mean predicted fitness alone without assessing the uncertainty of those predictions, it grows monotonically with λ (Fig. 4*A*), which would not suggest any harm from setting λ to a higher value.

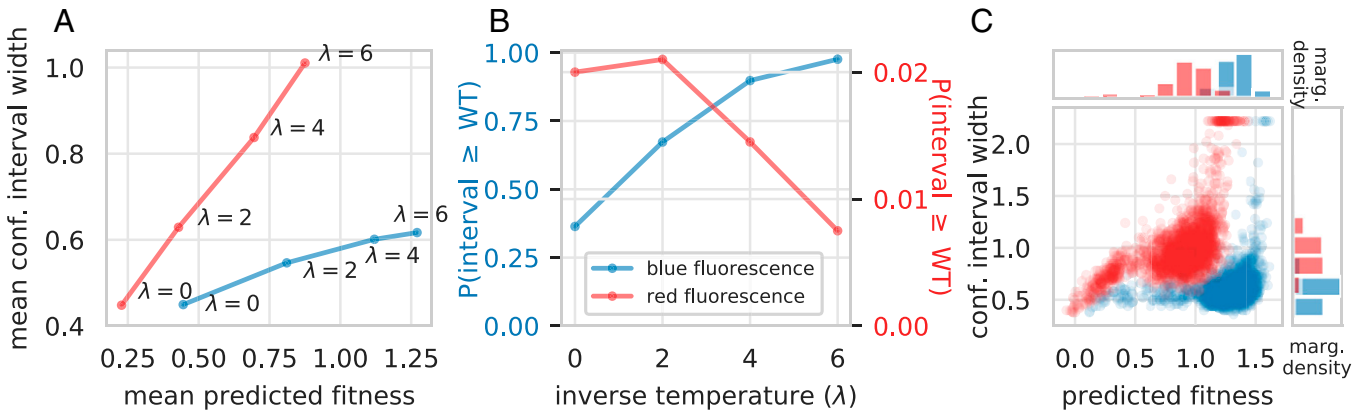


Fig. 4. Comparison of tradeoff between predicted fitness and predictive certainty on the red and blue fluorescence datasets. (*A*) Tradeoff between mean confidence interval width and mean predicted fitness for different values of the inverse temperature, λ, and $n = 384$ training data points. (*B*) Empirical probability that the smallest fitness value in the confidence intervals of designed proteins exceeds the true fitness of one of the wild-type parent sequences, mKate2. (*C*) For $n = 384$ and $\lambda = 6$, the distributions of both confidence interval width and predicted fitnesses of designed proteins.

In contrast, for blue fluorescence, although the mean interval width also grows with λ, it does so at a much slower rate than for red fluorescence (Fig. 4*A*); correspondingly, the empirical frequency at which the confidence interval surpasses the fitness of the wild type also grows monotonically (Fig. 4*B*).

We can observe these differences in the tradeoff between blue and red fluorescence even for a fixed value of λ. For example, for $n = 384$, $\lambda = 6$ (Fig. 4*C*), observe that proteins designed for blue fluorescence (blue circles) have confidence intervals with widths mostly less than 1. That is, those with higher predicted fitnesses do not have much wider intervals than those with lower predicted fitnesses, except for a small fraction of proteins with the highest predicted fitnesses. In contrast, for red fluorescence, designed proteins with higher predicted fitnesses also tend to have wider confidence intervals.

B. Design Experiments Using Adeno-Associated Virus Capsid Packaging Data. In contrast with *Section 3.A*, which represented a protein design problem with limited amounts of labeled data (at most a few hundred sequences), here we focus on a setting in which there are abundant labeled data. We can therefore employ data splitting as described in *Section 2.E* to construct confidence sets, as an alternative to computing full conformal confidence sets (Eq. **3**) as done in *Section 3.A*. Specifically, we construct a randomized version of the split conformal confidence set (*SI Appendix*, Algorithm S1), which achieves exact coverage.

This subsection, together with the previous subsection, demonstrates that in both regimes—limited and abundant labeled data—our proposed methods provide confidence sets that give coverage, are not overly conservative, and can be used to visualize the tradeoff between predicted fitness and predictive uncertainty inherent to a design algorithm.

B.1. Protein design problem: Adeno-associated virus capsid proteins with improved packaging ability. Adeno-associated viruses (AAVs) are a class of viruses whose capsid, the protein shell that encapsulates the viral genome, is a promising delivery vehicle for gene therapy. As such, the proteins that constitute the capsid have been modified to enhance various fitness functions, such as the ability to enter specific cell types and evade the immune system (64–66). Such efforts usually start by sampling millions of proteins from some sequence distribution and then performing an experiment that selects out the fittest sequences. Sequence distributions commonly used today have relatively high entropy, and the resulting sequence diversity can lead to successful outcomes for a myriad of downstream selection experiments (43, 49). However, most of these sequences fail to assemble into a capsid that packages the genetic payload (66–68)—a function called packaging, which is the minimum requirement of a gene therapy delivery mechanism and therefore a prerequisite to any other desiderata.

If sequence distributions could be developed with higher packaging rate, without compromising sequence diversity, then the success rate of downstream selection experiments should improve. To this end, Zhu et al. (43) use neural networks trained on sequence-packaging data to specify the parameters of DNA sequence distributions that simultaneously have high entropy and yield protein sequences with high predicted packaging ability. The protein sequences in these data varied at seven promising contiguous positions identified in previous work (65) and elsewhere matched a wild type. To accommodate commonly used DNA synthesis protocols, Zhu et al. (43) parameterized their DNA sequence distributions as independent categorical distributions over the four nucleotides at each of 21 contiguous sites, corresponding to codons at each of the seven sites of interest.

B.2. Protocol for design experiments. We followed the methodology of Zhu et al. (43) to find sequence distributions with high mean predicted fitness—in particular, higher than that of the commonly used "NNK" sequence distribution (65). Specifically, we used their high-throughput data, which sampled millions of sequences from the NNK distribution, and labeled each with an enrichment score quantifying its packaging fitness, based on its count before and after a packaging-based selection experiment. We introduced additional simulated measurement noise to these labels, where the parameters of the noise distribution were also estimated from the pre- and postselection counts, resulting in labels ranging from -7.53 to 8.80 for 8,552,729 sequences (see *SI Appendix*, section S4 for details).

We then randomly selected and held out 1 million of these data points, for calibration and test purposes described shortly, and then trained a neural network on the remaining data to predict fitness from sequence. Finally, following ref. 43, we approximately solved an optimization problem that leveraged this regression model to specify the parameters of sequence distributions with high mean predicted fitness. Specifically, let $\{p_\phi : \phi \in \Phi\}$ denote the class of sequence distributions parameterized as independent categorical distributions over the four nucleotides at each of 21 contiguous sequence positions. We set the parameters of the designed sequence distribution by using stochastic gradient descent to approximately solve the following problem:

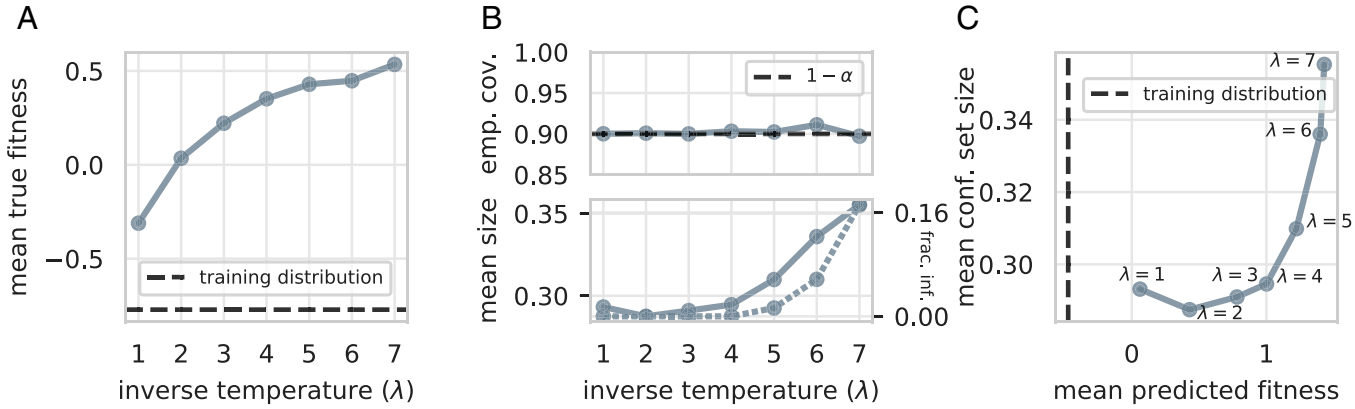


Fig. 5. Quantifying uncertainty for predicted fitnesses of designed AAV capsid proteins. (*A*) Mean true fitness of designed sequences resulting from different values of the inverse temperature, $\lambda \in \{1, 2, \ldots, 7\}$. The dashed black line is the mean true fitness of sequences drawn from the NNK sequence distribution (i.e., the training distribution). (*B*) *Top*, empirical coverage of randomized staircase confidence sets (*SI Appendix*, section S1.D) constructed for designed sequences. The dashed black line is the expected empirical coverage of $1 - \alpha = 0.9$. *Bottom*, fraction of confidence sets with infinite size (dashed gray line) and mean size of noninfinite confidence sets (solid gray line). The set size is reported as a fraction of the range of fitnesses in all the labeled data, $[-7.53, 8.80]$. (*C*) Tradeoff between mean predicted fitness and mean confidence set size for $\lambda \in \{1, 2, \ldots, 7\}$. The dashed black line is the mean predicted fitness for sequences from the training distribution.

$$\phi_\lambda = \underset{\phi \in \Phi}{\arg\min}\, D_{\mathrm{KL}}(p^\star_\lambda || p_\phi), \qquad [7]$$

where $p^\star_\lambda(X) \propto \exp(\lambda \cdot \mu(X))$, μ is the neural network fitted to the training data, and $\lambda \geq 0$ is an inverse temperature hyperparameter. After solving for ϕ_λ for a range of inverse temperature values, $\lambda \in \{1, 2, 3, 4, 5, 6, 7\}$, we sampled designed sequences from p_{ϕ_λ} as described below and then used a randomized data-splitting approach to construct confidence sets that achieve exact coverage.

Sampling designed sequences. Unlike in *Section 3.A*, here we did not have a label for every sequence in the input space—that is, all sequences that vary at the seven positions of interest and that elsewhere match a wild type. As an alternative, we used rejection sampling to sample from p_{ϕ_λ}. Specifically, recall that we held out 1 million of the labeled sequences. The input space was sampled uniformly and densely enough by the high-throughput dataset that we treated 990,000 of these held-out labeled sequences as samples from a proposal distribution (that is, the NNK distribution) and were able to perform rejection sampling to sample designed sequences from p_{ϕ_λ} for which we have labels.

Constructing confidence sets for designed sequences. Note that rejection sampling results in some random number, at most 990,000, of designed sequences; in practice, this number ranged from single digits to several thousand for $\lambda = 7$ to $\lambda = 1$, respectively. To account for this variability, for each value of the inverse temperature, we performed $T = 500$ trials of the following steps. We randomly split the 1 million held-out labeled sequences into 990,000 proposal distribution sequences and 10,000 sequences to be used as calibration data. We used the former to sample some number of designed sequences and then used the latter to construct randomized staircase confidence sets (*SI Appendix*, *Algorithm S1*) for each of the designed sequences. The results we report next concern properties of these sets averaged over all $T = 500$ trials.

B.3. Results.

Effect of inverse temperature. The inverse temperature hyperparameter, λ, in Eq. 7 plays a similar role to that in *Section 3.A*: Larger values result in designed sequences with higher mean true fitness (Fig. 5*A*). Note that the mean true fitness for all considered values of the inverse temperature is higher than that of the training distribution (dashed black line, Fig. 5*A*).

Empirical coverage and confidence set sizes. For all considered values of the inverse temperature, the empirical coverage of the confidence sets is very close to the expected value of $1 - \alpha = 0.9$ (Fig. 5*B*, *Top*). Note that some designed sequences, which the neural network is particularly uncertain about, are given a confidence set with infinite size (Fig. 5*B*, *Bottom*). The fraction of sets with infinite size and the mean size of noninfinite sets both increase with the inverse temperature (Fig. 5*B*, *Bottom*), which is consistent with our intuition that the neural network should be less confident about predictions that are much higher than fitnesses seen in the training data.

Using uncertainty quantification to set design procedure hyperparameters. As in *Section 3.A.2*, the confidence sets we construct expose a tradeoff between predicted fitness and predictive uncertainty as we vary the inverse temperature. Generally, the higher the mean predicted fitness of the sequence distributions is, the greater the mean confidence set size as well (Fig. 5*C*).[#] One can inspect this tradeoff to decide on an acceptable setting of the inverse temperature. For example, observe that the mean set size does not grow appreciably between $\lambda = 1$ and $\lambda = 4$, even though the mean predicted fitness monotonically increases (Fig. 5*B*, *Bottom* and *C*); similarly, the fraction of sets with infinite size also remains near zero for these values of λ (Fig. 5*B*, *Bottom*). However, both of these quantities start to increase for $\lambda \geq 5$. By $\lambda = 7$, for instance, more than 17% of designed sequences are given a confidence set with infinite size, suggesting that p_{ϕ_7} has shifted too far from the training distribution for the neural network to be reasonably certain about its predictions. Therefore, one might conclude that using $\lambda \in \{4, 5\}$ achieves an acceptable balance of designed sequences with higher predicted fitness than the training sequences and low enough predictive uncertainty.

4. Discussion

The predictions made by machine-learning models are increasingly being used to make consequential decisions, which in turn influence the data that the models encounter. Our work presents a methodology that allows practitioners to trust the predictions of

[#]The exceptions are the sequence distributions corresponding to $\lambda = 2$ and $\lambda = 3$, which have a higher mean predicted fitness but on average smaller sets than $\lambda = 1$. One likely explanation is that experimental measurement noise is particularly high for very low fitnesses, making low-fitness sequences inherently difficult to predict.

learned models in such settings. In particular, our protein design examples demonstrate how our approach can be used to navigate the tradeoff between desirable predictions and predictive certainty inherent to design problems.

Looking beyond the design problem, the formalism of FCS introduced here captures a range of problem settings pertinent to modern-day deployments of machine learning. In particular, FCS often occurs at each iteration of a feedback loop—for example, at each iteration of active learning, adaptive experimental design, and Bayesian optimization methods. Applications and extensions of our approach to such settings are exciting directions for future investigation.

Data, Materials, and Software Availability. Code and all data for reproducing our experiments are available at https://github.com/clarafy/conformal-for-design (69). Previously published data were also used for this work (60).

ACKNOWLEDGMENTS. We are grateful to Danqing Zhu and David Schaffer for generously allowing us to work with their AAV packaging data and to David H. Brookes and Akosua Busia for guidance on its analysis. We also thank David H. Brookes, Hunter Nisonoff, Tijana Zrnic, and Meena Jagadeesan for insightful discussions and feedback. C.F. was supported by a NSF Graduate Research Fellowship under Grant DGE 2146752. A.N.A. was partially supported by a NSF Graduate Research Fellowship and a Berkeley Fellowship. S.B. was supported by the Foundations of Data Science Institute and the Simons Institute.

1. V. Vovk, A. Gammerman, C. Saunders, "Machine -learning applications of algorithmic randomness" in *Proceedings of the Sixteenth International Conference on Machine Learning, ICML '99* I. Bratko, S. Dzeroski, Eds. (Morgan Kaufmann Publishers Inc., San Francisco, CA, 1999), pp. 444–453.
2. V. Vovk, A. Gammerman, G. Shafer, *Algorithmic Learning in a Random World* (Springer, New York, NY, 2005).
3. J. Lei, M. G'Sell, A. Rinaldo, R. J. Tibshirani, L. Wasserman, Distribution-free predictive inference for regression. *J. Am. Stat. Assoc.* **113**, 1094–1111 (2018).
4. R. J. Tibshirani, R. Foygel Barber, E. Candes, A. Ramdas, Conformal prediction under covariate shift. *Adv. Neural Inf. Process. Syst.* **32**, 2530–2540 (2019).
5. M. Cauchois, S. Gupta, A. Ali, J. C. Duchi, Robust validation: Confident predictions even when distributions shift. *arXiv* [Preprint] (2020). https://arxiv.org/abs/2008.04267 (Accessed 1 February 2022).
6. I. Gibbs, E. Candès, Adaptive conformal inference under distribution shift. *Adv. Neural Inf. Process. Syst.* **34**, 1660–1672 (2021).
7. S. Park, S. Li, O. Bastani, I. Lee, "PAC confidence predictions for deep neural network classifiers" in *Proceedings of the Ninth International Conference on Learning Representations* (OpenReview.net, 2021).
8. A. Podkopaev, A. Ramdas, "Distribution -free uncertainty quantification for classification under label shift" in *Proceedings of the 37th Uncertainty in Artificial Intelligence*, C. de Campos, M. H. Maathuis, Eds. (PMLR, 2021), pp. 844–853.
9. H. Shimodaira, Improving predictive inference under covariate shift by weighting the log-likelihood function. *J. Stat. Plan. Inference* **90**, 227–244 (2000).
10. M. Sugiyama, K. R. Müller, Input-dependent estimation of generalization error under covariate shift. *Stat. Decis.* **23**, 249–279 (2005).
11. M. Sugiyama, M. Krauledat, K. R. Müller, Covariate shift adaptation by importance weighted cross validation. *J. Mach. Learn. Res.* **8**, 985–1005 (2007).
12. J. Quiñonero Candela, M. Sugiyama, A. Schwaighofer, N. D. Lawrence, *Dataset Shift in Machine Learning* (The MIT Press, 2009).
13. M. Hardt, N. Megiddo, C. Papadimitriou, M. Wootters, "Strategic classification" in *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, M. Sudan, Ed. (Association for Computing Machinery, New York, NY, 2016) pp. 111–122.
14. J. Perdomo, T. Zrnic, C. Mendler-Dünner, M. Hardt, "Performative prediction" in *Proceedings of the 37th International Conference on Machine Learning*, H. Daumé III, A. Singh, Eds. (PMLR, 2020), vol. 119, pp. 7599–7609.
15. A. Gammerman, V. Vovk, V. Vapnik, Learning by transduction. *Proc. Fourteenth Conf. Uncertain. Artif. Intell.* **14**, 148–155 (1998).
16. A. N. Angelopoulos, S. Bates, A gentle introduction to conformal prediction and distribution-free uncertainty quantification. *arXiv* [Preprint] (2021). https://doi.org/10.48550/arXiv.2107.07511 (Accessed 1 February 2022).
17. V. Vovk, Testing for concept shift online. *arXiv* [Preprint] (2020). https://doi.org/10.48550/arXiv.2012.14246 (Accessed 1 February 2022).
18. X. Hu, J. Lei, A distribution-free test of covariate shift using conformal prediction. *arXiv* [Preprint] (2020). https://doi.org/10.48550/arXiv.2010.07147 (Accessed 1 February 2022).
19. R. Luo *et al.*, "Sample-efficient safety assurances using conformal prediction". Workshop on Algorithmic Foundations of Robotics. arXiv [Preprint] (2022). https://arxiv.org/abs/2109.14082 (Accessed 1 February 2022).
20. S. Bates, E. Candès, L. Lei, Y. Romano, M. Sesia, Testing for outliers with conformal p-values. *arXiv* [Preprint] (2021). https://doi.org/10.48550/arXiv.2104.08279 (Accessed 1 February 2022).
21. A. N. Angelopoulos, S. Bates, E. J. Candès, M. I. Jordan, L. Lei, Learn then test: Calibrating predictive algorithms to achieve risk control. *arXiv* [Preprint] (2021). https://doi.org/10.48550/arXiv.2110.01052 (Accessed 1 February 2022).
22. A. Podkopaev, A. Ramdas, "Tracking the risk of a deployed model and detecting harmful distribution shifts" in *Proceedings of the Tenth International Conference on Learning Representations* (2022).
23. R. Kaur *et al.*, "iDECODe: In-distribution equivariance for conformal out-of-distribution detection" in *Proceedings of the 36th AAAI Conference on Artificial Intelligence* (AAAI Press, Palo Alto, CA, 2022).
24. D. H. Brookes, H. Park, J. Listgarten, "Conditioning by adaptive sampling for robust design" in *Proceedings of the International Conference on Machine Learning (ICML)*, K. Chaudhuri, R. Salakhutdinov, Eds. (PMLR, 2019).
25. C. Fannjiang, J. Listgarten, "Autofocused oracles for model-based design" in *Advances in Neural Information Processing Systems 33* H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, H. Lin, Eds. (Curran Associates, Inc., Red Hook, NY, 2020) pp. 12945–12956.
26. P. Auer, Using confidence bounds for exploitation-exploration trade-offs. *J. Mach. Learn. Res.* **3**, 397–422 (2002).
27. J. Snoek, H. Larochelle, R. P. Adams, "Practical Bayesian optimization of machine learning algorithms" in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. C. Burges, L. Bottou, K. Q. Weinberger, Eds., (Curran Associates, Inc., 2012), vol. 25, pp. 2960–2968.
28. P. A. Romero, A. Krause, F. H. Arnold, Navigating the protein fitness landscape with Gaussian processes. *Proc. Natl. Acad. Sci. U.S.A.* **110**, E193–E201 (2013).
29. J. C. Greenhalgh, S. A. Fahlberg, B. F. Pfleger, P. A. Romero, Machine learning-guided acyl-ACP reductase engineering for improved in vivo fatty alcohol production. *Nat. Commun.* **12**, 5825 (2021).
30. B. Lakshminarayanan, A. Pritzel, C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles" in *Advances in Neural Information Processing Systems*, I. Guyon et al., Eds. (Curran Associates, Inc. Red Hook, NY, 2017), pp. 6402–6413.
31. H. Zeng, D. K. Gifford, Quantification of uncertainty in Peptide-MHC binding prediction improves high-affinity peptide selection for therapeutic design. *Cell Syst.* **9**, 159–166.e3 (2019).
32. G. Liu *et al.*, Antibody complementarity determining region design using high-capacity machine learning. *Bioinformatics* **36**, 2126–2133 (2020).
33. A. P. Soleimany *et al.*, Evidential deep learning for guided molecular property prediction and discovery. *ACS Cent. Sci.* **7**, 1356–1367 (2021).
34. A. Amini, W. Schwarting, A. Soleimany, D. Rus, "Deep evidential regression" in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, H. Lin, Eds., (Curran Associates, Inc., 2020), vol. 33, pp. 14927–14937.
35. V. Kuleshov, N. Fenner, S. Ermon, "Accurate uncertainties for deep learning using calibrated regression" in *Proceedings of the 35th International Conference on Machine Learning*, J. G. Dy, A. Krause, Eds. (PMLR, 2018).
36. S. Biswas, G. Khimulya, E. C. Alley, K. M. Esvelt, G. M. Church, Low-N protein engineering with data-efficient deep learning. *Nat. Methods* **18**, 389–396 (2021).
37. M. Popova, O. Isayev, A. Tropsha, Deep reinforcement learning for de novo drug design. *Sci. Adv.* **4**, eaap7885 (2018).
38. S. Kang, K. Cho, Conditional molecular design with deep generative models. *J. Chem. Inf. Model.* **59**, 43–52 (2019).
39. W. P. Russ *et al.*, An evolution-based model for designing chorismate mutase enzymes. *Science* **369**, 440–445 (2020).
40. Z. Wu *et al.*, Signal peptides generated by attention-based neural networks. *ACS Synth. Biol.* **9**, 2154–2161 (2020).
41. A. Hawkins-Hooker *et al.*, Generating functional protein variants with variational autoencoders. *PLOS Comput. Biol.* **17**, e1008736 (2021).
42. J. E. Shin *et al.*, Protein design and variant prediction using autoregressive generative models. *Nat. Commun.* **12**, 2403 (2021).
43. D. Zhu *et al.*, Optimal trade-off control in machine learning-based library design, with application to adeno-associated virus (AAV) for gene therapy *bioRxiv* [Preprint] (2021). https://doi.org/10.1101/2021.11.02.467003 (Accessed 1 February 2022).
44. N. Killoran, L. J. Lee, A. Delong, D. Duvenaud, B. J. Frey, "Generating and designing DNA with deep generative models" in *Neural Information Processing Systems (NeurIPS)* (Computational Biology Workshop, 2017). https://arxiv.org/abs/1712.06148 (Accessed 1 February 2022).
45. R. Gómez-Bombarelli *et al.*, Automatic chemical design using a data-driven continuous representation of molecules. *ACS Cent. Sci.* **4**, 268–276 (2018).
46. J. Linder, N. Bogard, A. B. Rosenberg, G. Seelig, A generative neural network for maximizing fitness and diversity of synthetic DNA and protein sequences. *Cell Syst.* **11**, 49–62.e16 (2020).
47. S. Sinai *et al.*, AdaLead: A simple and robust adaptive greedy search algorithm for sequence design. *arXiv* [Preprint] (2020). https://doi.org/10.48550/arXiv.2010.02141 (Accessed 1 February 2022).
48. A. Bashir *et al.*, Machine learning guided aptamer refinement and discovery. *Nat. Commun.* **12**, 2366 (2021).
49. D. H. Bryant *et al.*, Deep diversification of an AAV capsid protein by machine learning. *Nat. Biotechnol.* **39**, 691–696 (2021).
50. Y. Li *et al.*, A diverse family of thermostable cytochrome P450s created by recombination of stabilizing fragments. *Nat. Biotechnol.* **25**, 1051–1056 (2007).
51. C. N. Bedbrook *et al.*, Machine learning-guided channelrhodopsin engineering enables minimally invasive optogenetics. *Nat. Methods* **16**, 1176–1184 (2019).
52. E. N. Weinstein *et al.*, "Optimal design of stochastic DNA synthesis protocols based on generative sequence models" in *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics*, G. Camps-Valls, F. J. R. Ruiz, I. Valera, Eds. (PMLR, 2022).
53. R. J. Fox *et al.*, Improving catalytic function by ProSAR-driven enzyme evolution. *Nat. Biotechnol.* **25**, 338–344 (2007).
54. K. K. Yang, Z. Wu, F. H. Arnold, Machine-learning-guided directed evolution for protein engineering. *Nat. Methods* **16**, 687–694 (2019).
55. S. Sinai, E. D. Kelsic, A primer on model-guided exploration of fitness landscapes for biological sequence design. *arXiv* [Preprint] (2020). https://arxiv.org/abs/2010.10614 (Accessed 1 February 2022).
56. Z. Wu, K. E. Johnston, F. H. Arnold, K. K. Yang, Protein sequence design with deep generative models. *Curr. Opin. Chem. Biol.* **65**, 18–27 (2021).
57. Z. Wu, S. B. J. Kan, R. D. Lewis, B. J. Wittmann, F. H. Arnold, Machine learning-assisted directed protein evolution with combinatorial libraries. *Proc. Natl. Acad. Sci. U.S.A.* **116**, 8852–8858 (2019).
58. C. Angermueller *et al.*, "Model-based reinforcement learning for biological sequence design" in *Proceedings of the International Conference on Learning Representations (ICLR)* (OpenReview.net, 2019).

59. B. J. Wittmann, Y. Yue, F. H. Arnold, Informed training set design enables efficient machine learning-assisted directed protein evolution. *Cell Syst.* **12**, 1026–1045.e7 (2021).
60. F. J. Poelwijk, M. Socolich, R. Ranganathan, Learning the pattern of epistasis linking genotype and phenotype in a protein. *Nat. Commun.* **10**, 4213 (2019).
61. D. H. Brookes, A. Aghazadeh, J. Listgarten, On the sparsity of fitness functions and implications for learning. *Proc. Natl. Acad. Sci. U.S.A.* **119**, e2109649118 (2022).
62. C. Hsu, H. Nisonoff, C. Fannjiang, J. Listgarten, Learning protein fitness models from evolutionary and assay-labeled data. *Nat. Biotechnol.* **40**, 1114–1122 (2022).
63. A. Madani *et al.*, Deep neural language modeling enables functional protein generation across families. *bioRxiv* [Preprint] (2021). https://doi.org/10.1101/2021.07.18.452833 (Accessed 1 February 2022).
64. N. Maheshri, J. T. Koerber, B. K. Kaspar, D. V. Schaffer, Directed evolution of adeno-associated virus yields enhanced gene delivery vectors. *Nat. Biotechnol.* **24**, 198–204 (2006).
65. D. Dalkara *et al.*, In vivo-directed evolution of a new adeno-associated virus for therapeutic outer retinal gene delivery from the vitreous. *Sci. Transl. Med.* **5**, 189ra76 (2013).
66. L. V. Tse *et al.*, Structure-guided evolution of antigenically distinct adeno-associated virus variants for immune evasion. *Proc. Natl. Acad. Sci. U.S.A.* **114**, E4812–E4821 (2017).
67. K. Adachi, T. Enoki, Y. Kawano, M. Veraz, H. Nakai, Drawing a high-resolution functional map of adeno-associated virus capsid by massively parallel sequencing. *Nat. Commun.* **5**, 3075 (2014).
68. P. J. Ogden, E. D. Kelsic, S. Sinai, G. M. Church, Comprehensive AAV capsid fitness landscape reveals a viral gene and enables machine-guided design. *Science* **366**, 1139–1143 (2019).
69. C. Fannjiang, Data for protein design experiments, Conformal Prediction for the Design Problem. GitHub. https://github.com/clarafy/conformal-for-design. Deposited 31 May 2022.

PNAS
www.pnas.org

Supplementary Information for

Conformal Prediction for Biological Design

Clara Fannjiang, Stephen Bates, Anastasios N. Angelopoulos, Jennifer Listgarten, and Michael I. Jordan

Clara Fannjiang and Michael I. Jordan.
E-mails: clarafy@berkeley.edu, jordan@cs.berkeley.com

This PDF file includes:

Supporting Information Text

S1. Proofs

A. Proof of Theorem 1. Data from feedback covariate shift (FCS) are a special case of what we call *pseudo-exchangeable*[*] random variables.

Definition S1. *Random variables $V_1, \ldots, V_{n+1}$ are* pseudo-exchangeable *with factor functions $g_1, \ldots, g_{n+1}$ and core function h if the density, f, of their joint distribution can be factorized as*

$$f(v_1, \ldots, v_{n+1}) = \prod_{i=1}^{n+1} g_i(v_i;\, v_{-i}) \cdot h(v_1, \ldots, v_{n+1}),$$

where $v_{-i} = v_{1:(n+1)} \setminus v_i$,[†] each $g_i(\cdot;\, v_{-i})$ is a function that depends on the multiset v_{-i} (that is, on the values in v_{-i} but not on their ordering), and h is a function that does not depend on the ordering of its $n+1$ inputs.

The following lemma characterizes the distribution of the scores of pseudo-exchangeable random variables, which allows for a pseudo-exchangeable generalization of conformal prediction in Theorem S1. We then show that data generated under FCS are pseudo-exchangeable, and a straightforward application of Theorem S1 yields Theorem 1 as a corollary. Our technical development here builds upon the work of Tibshirani et al. (1), who generalized conformal prediction to handle "weighted exchangeable" random variables, including data under standard covariate shift.

The key insight is that if we condition on the values, but not the ordering, of the scores, we can exactly describe their distribution. The following proposition is a generalization of arguments found in the proof of Lemma 3 in (1); the subsequent result in Lemma 1 is a generalization of that lemma.

Proposition 1. *Let $Z_1, \ldots, Z_{n+1}$ be pseudo-exchangeable random variables with a joint density function, f, that can be written with factor functions $g_1, \ldots, g_{n+1}$ and core function h. Let S be any score function and denote $S_i = S(Z_i, Z_{-i})$ where $Z_{-i} = Z_{1:(n+1)} \setminus \{Z_i\}$ for $i = 1, \ldots, n+1$. Define*

$$w_i(z_1, \ldots, z_{n+1}) \equiv \frac{\sum_{\sigma:\sigma(n+1)=i} \prod_{j=1}^{n+1} g_j(z_{\sigma(j)}; z_{-\sigma(j)})}{\sum_{\sigma} \prod_{j=1}^{n+1} g_j(z_{\sigma(j)}; z_{-\sigma(j)})}, \quad i = 1, \ldots, n+1, \tag{S1}$$

where the summations are taken over permutations, σ, of the integers $1, \ldots, n+1$. For values $z = (z_1, \ldots, z_{n+1})$, let $s_i = S(z_i, z_{-i})$ and let E_z be the event that $\{Z_1, \ldots, Z_{n+1}\} = \{z_1, \ldots, z_{n+1}\}$ (that is, the multiset of values taken on by $Z_1, \ldots, Z_{n+1}$ equals the multiset of the values in z). Then

$$S_{n+1} \mid E_z \sim \sum_{i=1}^{n+1} w_i(z_1, \ldots, z_{n+1})\, \delta_{s_i}.$$

Proof. For simplicity, we treat the case where $S_1, \ldots, S_{n+1}$ are distinct almost surely; the result also holds in the general case, but the notation that accommodates duplicate values is cumbersome. For $i = 1, \ldots, n+1$,

$$\begin{aligned}
\mathbb{P}(S_{n+1} = s_i \mid E_z) = \mathbb{P}(Z_{n+1} = z_i \mid E_z) &= \frac{\sum_{\sigma:\sigma(n+1)=i} f(z_{\sigma(1)}, \ldots, z_{\sigma(n+1)})}{\sum_{\sigma} f(z_{\sigma(1)}, \ldots, z_{\sigma(n+1)})} \\
&= \frac{\sum_{\sigma:\sigma(n+1)=i} \prod_{j=1}^{n+1} g_j(z_{\sigma(j)}; z_{-\sigma(j)}) \cdot h(z_{\sigma(1)}, \ldots, z_{\sigma(n+1)})}{\sum_{\sigma} \prod_{j=1}^{n+1} g_j(z_{\sigma(j)}; z_{-\sigma(j)}) \cdot h(z_{\sigma(1)}, \ldots, z_{\sigma(n+1)})} \\
&= \frac{\sum_{\sigma:\sigma(n+1)=i} \prod_{j=1}^{n+1} g_j(z_{\sigma(j)}; z_{-\sigma(j)}) \cdot h(z_1, \ldots, z_{n+1})}{\sum_{\sigma} \prod_{j=1}^{n+1} g_j(z_{\sigma(j)}; z_{-\sigma(j)}) \cdot h(z_1, \ldots, z_{n+1})} \\
&= \frac{\sum_{\sigma:\sigma(n+1)=i} \prod_{j=1}^{n+1} g_j(z_{\sigma(j)}; z_{-\sigma(j)})}{\sum_{\sigma} \prod_{j=1}^{n+1} g_j(z_{\sigma(j)}; z_{-\sigma(j)})} \\
&= w_i(z_1, \ldots, z_{n+1}).
\end{aligned}$$

□

[*] The name *pseudo-exchangeable* hearkens to the similarity of the factorized form to the pseudo-likelihood approximation of a joint density. Note, however, that each factor, $g_i(v_i;\, v_{-i})$, can only depend on the values and not the ordering of the other variables, $v_1, \ldots, v_{i-1}, v_{i+1}, \ldots, v_n$, whereas each factor in the pseudo-likelihood approximation also depends on the identities (i.e., the ordering) of the other variables.

[†] With some abuse of notation, we denote $z_{-i} = z_{1:(n+1)} \setminus z_i$ whenever possible, as done here, but use $z_{-i} = z_{1:n} \setminus z_i$ whenever we need to append a candidate test point, as done in the main text and in Theorem S1 below. In either case, we will clarify.

Lemma 1. *Let $Z_1, \ldots, Z_{n+1}$ be pseudo-exchangeable random variables with a joint density function, f, that can be written with factor functions $g_1, \ldots, g_{n+1}$ and core function h. Let S be any score function and denote $S_i = S(Z_i, Z_{-i})$ where $Z_{-i} = Z_{1:(n+1)} \setminus \{Z_i\}$ for $i = 1, \ldots, n+1$. For any $\beta \in (0,1)$,*

$$\mathbb{P}\left\{ S_{n+1} \leq \textsc{Quantile}_\beta\left(\sum_{i=1}^{n+1} w_i(Z_1, \ldots, Z_{n+1})\, \delta_{S_i} \right) \right\} \geq \beta,$$

where $w_i(z_1, \ldots, z_{n+1})$ is defined in Eq. [S1].

Proof. Assume for simplicity of notation that $S_1, \ldots, S_{n+1}$ are distinct almost surely (but the result holds generally). For data point values $z = (z_1, \ldots, z_{n+1})$, let $s_i = S(z_i, z_{-i})$ and let E_z be the event that $\{Z_1, \ldots, Z_{n+1}\} = \{z_1, \ldots, z_{n+1}\}$. By Proposition 1,

$$S_{n+1} \mid E_z \sim \sum_{i=1}^{n+1} w_i(z_1, \ldots, z_{n+1})\, \delta_{s_i},$$

and consequently

$$\mathbb{P}\left(S_{n+1} \leq \textsc{Quantile}_\beta\left(\sum_{i=1}^{n+1} w_i(z_1, \ldots, z_{n+1})\, \delta_{s_i} \right) \,\middle|\, E_z \right) \geq \beta,$$

by definition of the β-quantile; equivalently, since we condition on E_z,

$$\mathbb{P}\left(S_{n+1} \leq \textsc{Quantile}_\beta\left(\sum_{i=1}^{n+1} w_i(Z_1, \ldots, Z_{n+1})\, \delta_{S_i} \right) \,\middle|\, E_z \right) \geq \beta.$$

Since this inequality holds for all events E_z, where z is a vector of $n+1$ data point values, smoothing gives

$$\mathbb{P}\left(S_{n+1} \leq \textsc{Quantile}_\beta\left(\sum_{i=1}^{n+1} w_i(Z_1, \ldots, Z_{n+1})\, \delta_{S_i} \right) \right) \geq \beta.$$

□

Lemma 1 yields the following theorem, which enables a generalization of conformal prediction to pseudo-exchangeable random variables.

Theorem S1. *Suppose $Z_1, \ldots, Z_{n+1}$ where $Z_i = (X_i, Y_i) \in \mathcal{X} \times \mathbb{R}$ are pseudo-exchangeable random variables with factor functions $g_1, \ldots, g_{n+1}$. For any score function, S, and any miscoverage level, $\alpha \in (0,1)$, define for any point $x \in \mathcal{X}$:*

$$C_\alpha(x) = \left\{ y \in \mathbb{R} : S_{n+1}(x,y) \leq \textsc{Quantile}_{1-\alpha}\left(\sum_{i=1}^{n+1} w_i(Z_1, \ldots, Z_n, (x,y))\, \delta_{S_i(x,y)} \right) \right\}, \qquad \text{[S2]}$$

where $S_i(x,y) = S(Z_i, Z_{-i} \cup \{(x,y)\})$ and $Z_{-i} = Z_{1:n} \setminus Z_i$ for $i = 1, \ldots, n$, $S_{n+1}(x,y) = S((x,y), Z_{1:n})$, and the weight functions w_i are as defined in Eq. [S1]. Then C_α satisfies

$$\mathbb{P}\left(Y_{n+1} \in C_\alpha(X_{n+1})\right) \geq 1 - \alpha,$$

where the probability is over all $n+1$ data points, $Z_1, \ldots, Z_{n+1}$.

Proof. By construction, we have

$$Y_{n+1} \in C_\alpha(X_{n+1}) \iff S_{n+1}(X_{n+1}, Y_{n+1}) \leq \textsc{Quantile}_{1-\alpha}\left(\sum_{i=1}^{n+1} w_i(Z_1, \ldots, Z_{n+1})\, \delta_{S_i(X_{n+1}, Y_{n+1})} \right).$$

Applying Lemma 1 gives the result. □

Finally, Theorem 1 follows as a corollary of Theorem S1. Denoting $Z_{n+1} = Z_{\text{test}}$ and $Z_{-i} = Z_{1:(n+1)} \setminus \{Z_i\}$, observe that data, $(Z_1, \ldots, Z_{n+1})$, under FCS are pseudo-exchangeable with the core function

$$h(z_1, \ldots, z_{n+1}) = \prod_{i=1}^{n+1} p_X(x_i)\, p_{Y|X}(y_i \mid x_i),$$

and factor functions $g_i(z_i;\, z_{-i}) = 1$ for $i = 1, \ldots, n$ and

$$g_{n+1}(z_{n+1};\, z_{1:n}) = \frac{\tilde{p}_{X;z_{1:n}}(x_{n+1})\, p_{Y|X}(y_{n+1} \mid x_{n+1})}{p_X(x_{n+1})\, p_{Y|X}(y_{n+1} \mid x_{n+1})} = \frac{\tilde{p}_{X;z_{1:n}}(x_{n+1})}{p_X(x_{n+1})} = v(x_{n+1}; z_{1:n})$$

where $v(\cdot;\cdot)$ is the likelihood ratio function defined in Eq. [2]. The weights, $w_i(z_1, \ldots, z_{n+1})$, in Eq. [S1] then simplify as

$$\begin{aligned}
w_i(z_1, \ldots, z_{n+1}) = \frac{\sum_{\sigma:\sigma(n+1)=i} \prod_{j=1}^{n+1} g_j(z_{\sigma(j)}; z_{-\sigma(j)})}{\sum_{\sigma} \prod_{j=1}^{n+1} g_j(z_{\sigma(j)}; z_{-\sigma(j)})} &= \frac{\sum_{\sigma:\sigma(n+1)=i} \prod_{j=1}^{n+1} g_j(z_{\sigma(j)}; z_{-\sigma(j)})}{\sum_{k=1}^{n+1} \sum_{\sigma:\sigma(n+1)=k} \prod_{j=1}^{n+1} g_j(z_{\sigma(j)}; z_{-\sigma(j)})} \\
&= \frac{\sum_{\sigma:\sigma(n+1)=i} g_{n+1}(z_{\sigma(n+1)}; z_{-\sigma(n+1)})}{\sum_{k=1}^{n+1} \sum_{\sigma:\sigma(n+1)=k} g_{n+1}(z_{\sigma(n+1)}; z_{-\sigma(n+1)})} \\
&= \frac{\sum_{\sigma:\sigma(n+1)=i} g_{n+1}(z_i; z_{-i})}{\sum_{k=1}^{n+1} \sum_{\sigma:\sigma(n+1)=k} g_{n+1}(z_k; z_{-k})} \\
&= \frac{n! \cdot g_{n+1}(z_i; z_{-i})}{\sum_{k=1}^{n+1} n! \cdot g_{n+1}(z_k; z_{-k})} \\
&= \frac{v(x_i; z_{-i})}{\sum_{k=1}^{n+1} v(x_k; z_{-k})}.
\end{aligned}$$

These quantities are exactly the weight functions, w_i^y, defined in Eq. [4] and used in the full conformal confidence set in Eq. [3]: $w_i^y(X_{\text{test}}) = w_i(Z_1, \ldots, Z_n, (X_{\text{test}}, y))$ for $i = 1, \ldots, n+1$. That is, Eq. [3] gives the confidence set defined in Eq. [S2] for data under FCS. Applying Theorem S1 then yields Theorem 1.

B. A randomized confidence set achieves exact coverage. Here, we introduce the *randomized β-quantile* and a corresponding randomized confidence set that achieves exact coverage. To lighten notation, for a discrete distribution with probability masses $w = (w_1, \ldots, w_{n+1})$ on points $s = (s_1, \ldots, s_{n+1})$, where $s_i \in \mathbb{R}$ and $w_i \geq 0, \sum_{i=1}^{n+1} w_i = 1$, we will write $\textsc{Quantile}_\beta(s, w) = \textsc{Quantile}_\beta(\sum_{i=1}^{n} w_i \delta_{s_i})$. Observe that $\textsc{Quantile}_\beta(s, w)$ is always one of the support points, s_i. Now define the *β-quantile lower bound*:

$$\textsc{QuantileLB}_\beta(s, w) = \inf \left\{ s : \sum_{i:s_i \leq s} w_i < \beta,\ \sum_{i:s_i \leq s} w_i + \sum_{j:s_j = \textsc{Quantile}_\beta(s,w)} w_j \geq \beta \right\},$$

which is either a support point strictly less than the β-quantile, or negative infinity. Finally, letting $\text{QF}_\beta(s, w)$ and $\text{LF}_\beta(s, w)$ denote the CDF of the discrete distribution at $\textsc{Quantile}_\beta(s, w)$ and $\textsc{QuantileLB}_\beta(s, w)$), respectively, the randomized β-quantile is a random variable that takes on the value of either the β-quantile or the β-quantile lower bound:

$$\textsc{RandomizedQuantile}_\beta(s, w) = \begin{cases} \textsc{QuantileLB}_\beta(s, w) & \text{w. p. } \frac{\text{QF}_\beta(s,w) - \beta}{\text{QF}_\beta(s,w) - \text{LF}_\beta(s,w)}, \\ \textsc{Quantile}_\beta(s, w) & \text{w. p. } 1 - \frac{\text{QF}_\beta(s,w) - \beta}{\text{QF}_\beta(s,w) - \text{LF}_\beta(s,w)}. \end{cases} \quad \text{[S3]}$$

We use this quantity to define the *randomized full conformal* confidence set, which, for any miscoverage level, $\alpha \in (0, 1)$, and $x \in \mathcal{X}$ is the following random variable:

$$C_\alpha^{\text{rand}}(x) = \left\{ y \in \mathbb{R} : S((x, y), Z_{1:n}) \leq \textsc{RandomizedQuantile}_{1-\alpha}(s(Z_1, \ldots, Z_n, (x, y)), w(Z_1, \ldots, Z_n, (x, y)) \right\}, \quad \text{[S4]}$$

where $s(Z_1, \ldots, Z_n, (x, y)) = (S_1, \ldots, S_n, S((x, y), Z_{1:n})$ and $S_i = S(Z_i, Z_{-i} \cup \{(x, y)\})$ for $i = 1, \ldots, n$, and $w(Z_1, \ldots, Z_n, (x, y)) = (w_1^y(x), \ldots, w_{n+1}^y(x))$ where $w_i^y(x)$ is defined in Eq. [4]. Note that for each candidate label, $y \in \mathbb{R}$, an independent randomized β-quantile is instantiated; some values will use the β-quantile as the threshold on the score, while the others will use the β-quantile lower bound. Randomizing the confidence set in this way yields the following result.

Theorem S2. *Suppose data, $Z_1, \ldots, Z_n, Z_{\text{test}}$, are generated under feedback covariate shift and assume $\tilde{P}_{X;D}$ is absolutely continuous with respect to P_X for all possible values of D. Then, for any miscoverage level, $\alpha \in (0, 1)$, the randomized full confidence set, C_α^{rand}, in Eq. [S4] satisfies the* exact coverage *property:*

$$\mathbb{P}(Y_{\text{test}} \in C_\alpha^{\text{rand}}(X_{\text{test}})) = 1 - \alpha, \quad \text{[S5]}$$

where the probability is over $Z_1, \ldots, Z_n, Z_{\text{test}}$ and the randomness in C_α^{rand}.

Proof. Denote $Z_{n+1} = Z_{\text{test}}$ and $Z = (Z_1, \ldots, Z_{n+1})$. For a vector of $n+1$ data point values, $z = (z_1, \ldots, z_{n+1})$, use the following shorthand:

$$\begin{aligned}
Q_\beta(z) &= \text{QUANTILE}_\beta(s(z), w(z)),\\
L_\beta(z) &= \text{QUANTILELB}_\beta(s(z), w(z)),\\
R_\beta(z) &= \text{RANDOMIZEDQUANTILE}_\beta(s(z), w(z)),\\
\text{QF}_\beta(z) &= \text{QF}_\beta(s(z), w(z)),\\
\text{LF}_\beta(z) &= \text{LF}_\beta(s(z), w(z)).
\end{aligned}$$

As in the proof of Lemma 1, consider the event, E_z, that $\{Z_1, \ldots, Z_{n+1}\} = \{z_1, \ldots, z_{n+1}\}$. Assuming for simplicity that the scores are distinct almost surely, by Proposition 1

$$S(Z_{n+1}, Z_{1:n}) \mid E_z \sim \sum_{i=1}^{n+1} w_i(z_1, \ldots, z_{n+1})\, \delta_{S(z_i, z_{-i})},$$

and consequently

$$\begin{aligned}
&\mathbb{P}(S(Z_{n+1}, Z_{1:n}) \le \text{R}_{1-\alpha}(z) \mid E_z)\\
&= \mathbb{P}(S(Z_{n+1}, Z_{1:n}) \le \text{R}_{1-\alpha}(z) \mid E_z, \text{R}_{1-\alpha}(z) = \text{Q}_{1-\alpha}(z)) \cdot \mathbb{P}(\text{R}_{1-\alpha}(z) = \text{Q}_{1-\alpha}(z) \mid E_z) +\\
&\quad \mathbb{P}(S(Z_{n+1}, Z_{1:n}) \le \text{R}_{1-\alpha}(z) \mid E_z, \text{R}_{1-\alpha}(z) = \text{L}_{1-\alpha}(z)) \cdot \mathbb{P}(\text{R}_{1-\alpha}(z) = \text{L}_{1-\alpha}(z) \mid E_z)\\
&= \mathbb{P}(S(Z_{n+1}, Z_{1:n}) \le \text{Q}_{1-\alpha}(z) \mid E_z) \cdot \left(1 - \frac{\text{QF}_{1-\alpha}(z) - (1-\alpha)}{\text{QF}_{1-\alpha}(z) - \text{LF}_{1-\alpha}(z)}\right) +\\
&\quad \mathbb{P}(S(Z_{n+1}, Z_{1:n}) \le \text{L}_{1-\alpha}(z) \mid E_z) \cdot \frac{\text{QF}_{1-\alpha}(z) - (1-\alpha)}{\text{QF}_{1-\alpha}(z) - \text{LF}_{1-\alpha}(z)}\\
&= \text{QF}_{1-\alpha}(z) \cdot \left(1 - \frac{\text{QF}_{1-\alpha}(z) - (1-\alpha)}{\text{QF}_{1-\alpha}(z) - \text{LF}_{1-\alpha}(z)}\right) + \text{LF}_{1-\alpha}(z) \cdot \frac{\text{QF}_{1-\alpha}(z) - (1-\alpha)}{\text{QF}_{1-\alpha}(z) - \text{LF}_{1-\alpha}(z)}\\
&= -\left(\text{QF}_{1-\alpha}(z) - \text{LF}_{1-\alpha}(z)\right) \cdot \frac{\text{QF}_{1-\alpha}(z) - (1-\alpha)}{\text{QF}_{1-\alpha}(z) - \text{LF}_{1-\alpha}(z)} + \text{QF}_{1-\alpha}(z)\\
&= -\text{QF}_{1-\alpha}(z) + (1-\alpha) + \text{QF}_{1-\alpha}(z)\\
&= 1 - \alpha.
\end{aligned}$$

Since we condition on E_z, we equivalently have

$$\mathbb{P}(S(Z_{n+1}, Z_{1:n}) \le R_{1-\alpha}(Z) \mid E_z) = 1 - \alpha,$$

and since this equality holds for all events E_z, where z is a vector of $n+1$ data point values, taking an expectation over E_z yields

$$\mathbb{P}(S(Z_{n+1}, Z_{1:n}) \le R_{1-\alpha}(Z)) = 1 - \alpha.$$

Finally, since

$$Y_{n+1} \in C_\alpha^{\text{rand}}(X_{n+1}) \iff S(Z_{n+1}, Z_{1:n}) \le R_{1-\alpha}(Z),$$

the result follows. □

Note that standard covariate shift is subsumed by feedback covariate shift, so Theorem S2 can be used to construct a randomized confidence set with exact coverage under standard covariate shift as well.

C. Data splitting. In general, computing the full conformal confidence set, $C_\alpha(x)$, using Alg. 1 requires fitting $(n+1) \times |\mathcal{Y}|$ regression models. A much more computationally attractive alternative is called a *data splitting* or *split conformal* approach (2, 3), in which we (i) randomly partition the labeled data into disjoint training and *calibration* data sets, (ii) fit a regression model to the training data, and (iii) use the scores that it provides for the calibration data (but not the training data) to construct confidence sets for test data points. Though this approach only requires fitting a single model, the trade-off is that it does not use the labeled data as efficiently: only some fraction of our labeled data can be used to train the regression model. This limitation may be inconsequential for settings with abundant data, but can be a nonstarter when labeled data is limited, such as in many protein design problems.

Here, we show how data splitting simplifies feedback covariate shift (FCS) to standard covariate shift. We then use the data splitting method from Tibshirani et al. (1) to produce confidence sets with coverage; the subsequent subsection shows how to introduce randomization to achieve exact coverage.

To begin, we recall the standard covariate shift model (4–6). The training data, $Z_1, \ldots, Z_n$ where $Z_i = (X_i, Y_i)$, are i.i.d. from some distribution: $X_i \sim P_X, Y_i \sim P_{Y|X_i}$ for $i = 1, \ldots, n$. A test data point, $Z_{\text{test}} = (X_{\text{test}}, Y_{\text{test}})$, is drawn from a different input distribution but the same conditional distribution, $X_{\text{test}} \sim \tilde{P}_X, Y_{\text{test}} \sim P_{Y|X_{\text{test}}}$, independently from the training data. In contrast to FCS, here the test input cannot be chosen in a way that depends on the training data.

Returning to FCS, suppose we randomly partition all our labeled data into disjoint training and calibration data sets. Let μ denote the regression model fit to the training data; we henceforth consider μ as fixed and make no further use of the training data. As such, without loss of generality we will use $Z_1, \ldots, Z_m$ to refer to the calibration data. Now suppose the test input distribution is induced by the trained regression model, μ; we write this as $\tilde{P}_{X;\mu}$. Observe that, conditioned on the training data, we now have a setting where the calibration and test data are drawn from different input distributions but the same conditional distribution, $P_{Y|X}$, and are independent of each other. That is, data splitting returns us to standard covariate shift.

To construct valid confidence sets under standard covariate shift, define the following likelihood ratio function:

$$v(x) = \frac{\tilde{p}_{X;\mu}(x)}{p_X(x)}, \tag{S6}$$

where p_X and $\tilde{p}_{X;\mu}$ refer to the densities of the training and test input distributions, respectively. We restrict our attention to score functions of the following form (7):

$$S(x, y) = \frac{|y - \mu(x)|}{u(x)}. \tag{S7}$$

where u is any heuristic, nonnegative notion of uncertainty; one can also set $u(x) = 1$ to recover the residual score function. Note that, since we condition on the training data and treat the regression model as fixed, the score of a point, (x, y), is no longer also a function of other data points. Finally, for any miscoverage level, $\alpha \in (0, 1)$, and any $x \in \mathcal{X}$, define the *split conformal* confidence set as

$$\begin{aligned} C_\alpha^{\text{split}}(x) &= \mu(x) \pm q \cdot u(x), \\ q &= \text{QUANTILE}_{1-\alpha}\left(\sum_{i=1}^{m} w_i(x)\,\delta_{S_i} + w_{n+1}(x)\,\delta_\infty\right), \end{aligned} \tag{S8}$$

where $S_i = S(X_i, Y_i)$ for $i = 1, \ldots, m$ and

$$\begin{aligned} w_i(x) &= \frac{v(X_i)}{\sum_{j=1}^{m} v(X_j) + v(x)}, \quad i = 1, \ldots, m, \\ w_{m+1}(x) &= \frac{v(x)}{\sum_{j=1}^{m} v(X_j) + v(x)}. \end{aligned} \tag{S9}$$

For data under standard covariate shift, the split conformal confidence set achieves coverage, as first shown in (1).

Theorem S3 (Corollary 1 in (1))**.** *Suppose calibration and test data, $Z_1, \ldots, Z_m, Z_{\text{test}}$, are under standard covariate shift, and assume $\tilde{P}_{X;\mu}$ is absolutely continuous with respect to P_X. For score functions of the form in Eq. [S7], and any miscoverage level, $\alpha \in (0, 1)$, the split conformal confidence set, $C_\alpha^{\text{split}}(x)$, in Eq. [S8] satisfies the coverage property in Eq. [1].*

To achieve exact coverage, we can introduce randomization, as we discuss next.

D. Data splitting with randomization achieves exact coverage. Here, we stay in the setting and notation of the previous subsection and demonstrate how randomizing the β-quantile enables a data splitting approach to achieve exact coverage. For any score function of the form in Eq. [S7], any miscoverage level, $\alpha \in (0, 1)$, the *randomized split conformal* confidence set is the following random variable for $x \in \mathcal{X}$:

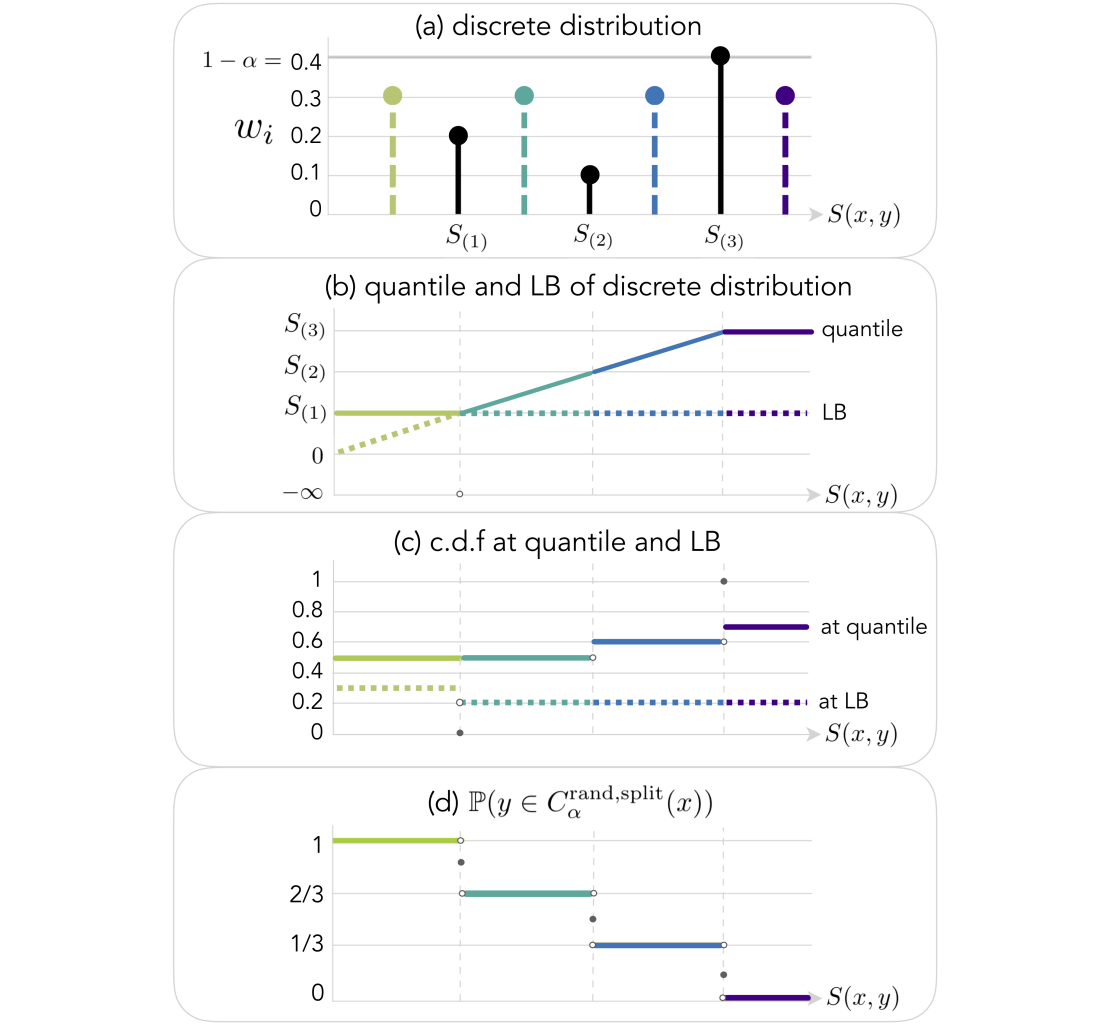
$$C_\alpha^{\text{rand,split}}(x) = \left\{y \in \mathbb{R} : S(x, y) \le \text{RANDOMIZEDQUANTILE}_{1-\alpha}\left((S_1, \ldots, S_m, S(x, y)), (w_1(x), \ldots, w_{m+1}(x))\right)\right\}, \tag{S10}$$

where the randomized β-quantile, $\text{RANDOMIZEDQUANTILE}_\beta$ is defined in Eq. [S3], $S_i = S(X_i, Y_i)$ for $i = 1, \ldots, m$, and $w_i(\cdot)$ for $i = 1, \ldots, m+1$ is defined in Eq. [S9]. Observe that for each candidate label, $y \in \mathbb{R}$, an independent randomized β-quantile is drawn, such that the scores of some values are compared to the β-quantile while the others are compared to the β-quantile lower bound. The exact coverage property of this confidence set is a consequence of Theorem S2.

Corollary 1. *Suppose calibration and test data, $Z_1, \ldots, Z_m, Z_{\text{test}}$, are under standard covariate shift, and assume $\tilde{P}_{X;\mu}$ is absolutely continuous with respect to P_X. For score functions of the form in Eq. [S7], and any miscoverage level, $\alpha \in (0, 1)$, the randomized split conformal confidence set, $C_\alpha^{\text{rand,split}}(x)$, in Eq. [S10] satisfies the exact coverage property in Eq. [S5].*

Proof. Since standard covariate shift is a special case of FCS, the calibration and test data can be described by FCS where $\tilde{P}_{X;D} = \tilde{P}_{X;\mu}$ for any multiset D. The randomized split conformal confidence set, $C_\alpha^{\text{rand,split}}$, is simply the randomized full conformal confidence set, C_α^{rand}, defined in Eq. [S4], instantiated with the scores $S((x, y), Z_{1:m}) = S(x, y)$ and $S(Z_i, Z_{-i} \cup \{(x, y)\}) = S(Z_i)$ for $i = 1, \ldots, m$, and weights resulting from $\tilde{P}_{X;D} = \tilde{P}_{X;\mu}$ for all D. The result then follows from Theorem S2. □

While we only need to fit a single regression model to compute the scores for data splitting, naively it might seem that in practice, we need to approximate $C_\alpha^{\text{rand,split}}(x)$ by introducing a discrete grid of candidate labels, $\mathcal{Y} \subset \mathbb{R}$, and computing a randomized β-quantile for $|\mathcal{Y}|$ different discrete distributions. Fortunately, we can construct an alternative confidence set that also achieves exact coverage, the *randomized staircase* confidence set, $C_\alpha^{\text{staircase}}$, which only requires sorting m scores and an additional $O(m)$ floating point operations to compute (see Alg. S1).



color legend:

$S(x,y) \in [0, S_{(1)})$	$S(x,y) \in (S_{(1)}, S_{(2)})$	$S(x,y) \in (S_{(2)}, S_{(3)})$	$S(x,y) \in (S_{(3)}, \infty]$
$\text{QUANTILE} = S_{(1)} > S(x,y)$ $\mathbb{P}(y \in C_\alpha^{\text{rand,split}}(x)) = 1$	$\text{QUANTILE} = S(x,y)$ $\text{QUANTILELB} = S_{(1)}$ $\text{QF} = 0.2 + 0.3$ $\text{LF} = 0.2$ $\mathbb{P}(y \in C_\alpha^{\text{rand,split}}(x)) = 2/3$	$\text{QUANTILE} = S(x,y)$ $\text{QUANTILELB} = S_{(2)}$ $\text{QF} = 0.2 + 0.1 + 0.3$ $\text{LF} = 0.2 + 0.1$ $\mathbb{P}(y \in C_\alpha^{\text{rand,split}}(x)) = 1/3$	$\text{QUANTILE} = S_{(3)} < S(x,y)$ $\mathbb{P}(y \in C_\alpha^{\text{rand,split}}(x)) = 0$

Fig. S1. Depiction of how the probability $\mathbb{P}(y \in C_\alpha^{\text{rand,split}}(x))$ is a piecewise constant function of y. (a) Given the values of the calibration data and test input, the scores $S_1, \ldots, S_m$ and corresponding probability masses $w_1, \ldots, w_m$ (black stems), as well as the probability mass for the test input, $w_{m+1} = 0.3$, are fixed. The only quantity that depends on y is $S(x,y)$. Four example values are shown as dashed green, teal, blue, and purple stems, representing values in $[0, S_{(1)}), (S_{(1)}, S_{(2)}), (S_{(2)}, S_{(3)})$, and $(S_{(3)}, \infty]$, respectively (see color legend). Note that in this example, $1 - \alpha = 0.4$. (b) The 0.4-quantile and 0.4-quantile lower bound of the discrete distribution in the top panel as a function of $S(x,y)$, where the colors correspond to values of $S(x,y)$ in the intervals just listed. Note the discontinuity in the 0.4-quantile lower bound at $S(x,y) = S_{(1)}$. (c) The c.d.f. of the discrete distribution at the 0.4-quantile and 0.4-quantile lower bound. Note the discontinuities when $S(x,y)$ equals a calibration score. (d) The probability $\mathbb{P}(y \in C_\alpha^{\text{rand,split}}(x))$, which equals 1 or 0 if $S(x,y) = 0.4$-quantile lower bound or $S(x,y) > 0.4$-quantile, respectively, and otherwise equals the probability in Eq. [S3] that the randomized 0.4-quantile equals the 0.4-quantile: $1 - \frac{\text{QF}-0.4}{\text{QF}-\text{LF}}$, where QF and LF denote the c.d.f. at the 0.4-quantile and 0.4-quantile lower bound, respectively. Color legend: calculations of the plotted quantities (calculations for $S(x,y) = S_{(i)}$ omitted).

At a high level, its construction is based on the observation that for any $x \in \mathcal{X}$ and $y \in \mathbb{R}$, the quantity $\mathbb{P}(y \in C_\alpha^{\text{rand,split}}(x))$, where the probability is over the randomness in $C_\alpha^{\text{rand,split}}(x)$, is a piecewise constant function of y. Instead of testing each value of $y \in \mathbb{R}$, we can then construct this piecewise constant function, and randomly include entire intervals of y values that

Algorithm S1 Randomized staircase confidence set

Input: Miscoverage level, $\alpha \in (0,1)$; calibration data, $Z_1, \ldots, Z_m$, where $Z_i = (X_i, Y_i)$; test input, X_{test}; subroutine for likelihood ratio function, $v(\cdot)$, defined in Eq. [S6]; subroutine for uncertainty heuristic, $u(\cdot)$; subroutine for regression model prediction, $\mu(\cdot)$.
Output: Randomized staircase confidence set, $C = C_\alpha^{\text{staircase}}(X_{\text{test}})$.

for $i = 1, \ldots, m$ **do** ▷ Compute calibration scores
  $S_i \leftarrow |Y_i - \mu(X_i)|/u(X_i)$
  $v_i \leftarrow v(X_i)$
$v_{m+1} \leftarrow v(X_{\text{test}})$
for $i = 1, \ldots, m+1$ **do** ▷ Compute calibration and test weights
  $w_i \leftarrow v_i / \sum_{j=1}^{m+1} v_j$
$C \leftarrow \emptyset$
`LowerBoundIsSet` $\leftarrow$ `False`
$S_{(0)} = 0, w_0 = 0$ ▷ Dummy values so for-loop will include $[0, S_{(1)}]$
for $i = 0, \ldots, m-1$ **do**
  if $\sum_{j:S_j \le S_{(i)}} w_j + w_{m+1} < 1-\alpha$ **then** ▷ $S(x,y) \le \beta$-quantile lower bound, so include deterministically
    $C = C \cup \left[\mu(X_{\text{test}}) + S_{(i)} \cdot u(X_{\text{test}}), \mu(X_{\text{test}}) + S_{(i+1)} \cdot u(X_{\text{test}})\right] \cup \left[\mu(X_{\text{test}}) - S_{(i+1)} \cdot u(X_{\text{test}}), \mu(X_{\text{test}}) - S_{(i)} \cdot u(X_{\text{test}})\right]$
  else if $\sum_{j:S_j \le S_{(i)}} w_j + w_{m+1} \ge 1-\alpha$ and $\sum_{j:S_j \le S_{(i)}} w_j < 1-\alpha$ **then** ▷ $S(x,y) = \beta$-quantile, so randomize inclusion
    if `LowerBoundIsSet` = `False` **then**
      `LowerBoundIsSet` $\leftarrow$ `True` ▷ Set β-quantile lower bound
      $LF = \sum_{j:S_j \le S_{(i)}} w_j$
      $F \leftarrow \dfrac{\sum_{j:S_j \le S_{(i)}} w_j + w_{m+1} - (1-\alpha)}{\sum_{j:S_j \le S_{(i)}} w_j + w_{m+1} - LF}$
    $b \sim \text{Bernoulli}(1-F)$
    if b **then**
      $C = C \cup \left[\mu(X_{\text{test}}) + S_{(i)} \cdot u(X_{\text{test}}), \mu(X_{\text{test}}) + S_{(i+1)} \cdot u(X_{\text{test}})\right] \cup \left[\mu(X_{\text{test}}) - S_{(i+1)} \cdot u(X_{\text{test}}), \mu(X_{\text{test}}) - S_{(i)} \cdot u(X_{\text{test}})\right]$
if $\sum_{i=1}^{m} w_i < 1-\alpha$ **then** ▷ For $S(x,y) > S_{(m)}$, either $S(x,y) = \beta$-quantile or $S(x,y) > \beta$-quantile
  if `LowerBoundIsSet` = `False` **then**
    $LF = \sum_{i=1}^{m} w_i$
  $F \leftarrow \frac{1-(1-\alpha)}{1-LF}$
  $b \sim \text{Bernoulli}(1-F)$
  if b **then**
    $C = C \cup \left[\mu(X_{\text{test}}) + S_{(m)} \cdot u(X_{\text{test}}), \infty\right] \cup \left[-\infty, \mu(X_{\text{test}}) - S_{(m)} \cdot u(X_{\text{test}})\right]$

have the same value of $\mathbb{P}(y \in C_\alpha^{\text{rand,split}}(x))$.

Fig. S1 illustrates this observation, which we now explain. First, the discrete distribution in Eq. [S10] has probability masses $w_1(x), \ldots, w_{m+1}(x)$ at the points $S_1, \ldots, S_m, S(x,y)$, respectively. Given the values of the m calibration data points and the test input, x, all of these quantities are fixed—except for the score of the candidate test data point, $S(x,y)$. That is, the only quantity that depends on the value of y is $S(x,y)$, which is the location of the probability mass $w_{m+1}(x)$; the remaining m support points and their corresponding probability masses do not not change with y.

Now consider the calibration scores, $S_1, \ldots, S_m$, sorted in ascending order. Observe that for any pair of successive sorted scores, $S_{(i)}$ and $S_{(i+1)}$, the entire interval of y values such that $S(x,y) \in (S_{(i)}, S_{(i+1)})$ belongs to one of three categories: $S(x,y) \le \beta$-quantile lower bound (of the discrete distribution with probability masses $w_1, \ldots, w_{m+1}$ at support points $S_1, \ldots, S_m, S(x,y)$), $S(x,y) = \beta$-quantile, or $S(x,y) > \beta$-quantile. An interval of y values that belongs to the first category is deterministically included in $C_\alpha^{\text{rand,split}}(x)$, regardless of the randomness in the randomized β-quantile (color-coded green in Fig. S1), while an interval that belongs to the last category is deterministically excluded (color-coded purple in Fig. S1). The only y values whose inclusion is not deterministic are those in the second category (color-coded teal and blue), which are randomly included with the probability, given in Eq. [S3], that the randomized β-quantile equals the β-quantile. Consequently, we can identify the intervals of y values belonging to each of these categories, and for those in the second category, compute the probability that the randomized β-quantile is instantiated as the β-quantile, which is $\mathbb{P}(y \in C_\alpha^{\text{rand,split}}(x))$.

This probability turns out to be a piecewise constant function of y. Note that it is computed from two quantities: the c.d.f. at the β-quantile and the c.d.f at the β-quantile lower bound (see Eq. [S3]). As depicted in Fig. S1 (third panel from top), for any two successive sorted calibration scores, $S_{(i)}$ and $S_{(i+1)}$, both of these quantities are constant over $S(x,y) \in (S_{(i)}, S_{(i+1)})$. That is, both the c.d.f. at the β-quantile and the c.d.f. at β-quantile lower bound are piecewise constant functions of y, which only change values at the calibration scores, $S_1, \ldots, S_m$ (and can take on different values exactly at the calibration scores). Consequently, the probability $\mathbb{P}(y \in C_\alpha^{\text{rand,split}}(x))$ is also a piecewise constant function of y, which only changes values at the calibration scores. It attains its highest value at $\mu(x)$ and decreases as y moves further away from it, resembling a staircase, as depicted in Fig. S1 (fourth panel from the top).

Therefore, instead of computing a randomized β-quantile for all $y \in \mathbb{R}$, we can simply compute the value of this probability on the $m+1$ intervals between neighboring sorted calibration scores: $[0, S_{(1)}), (S_{(1)}, S_{(2)}), \ldots, (S_{(m-1)}, S_{(m)}), (S_{(m)}, \infty]$, as well as its value exactly at the m calibration scores. These probabilities may equal 1 or 0, which correspond to the two cases earlier described wherein y is deterministically included or excluded, respectively. If the probability is not 1 or 0, then we can randomly include the entire set of values of y such that $S(x,y)$ falls in the interval. Due to the form of the score in Eq. [S7], this set comprises two equal-length intervals on both sides of $\mu(x)$: $(\mu(x) - S_{(i+1)}, \mu(x) - S_{(i)}) \cup (\mu(x) + S_{(i+1)}, \mu(x) + S_{(i)})$.

Finally, if we assume that scores are distinct almost surely, then our treatment of values of y such that $S(x,y) = S_i$ for $i = 1, \ldots, m$, does not affect the exact coverage property. For simplicity, Alg. S1 therefore includes or excludes closed intervals that contain these y values as endpoints, rather than treating them separately.

More general score functions. In the reasoning above, we use the assumption that the score function takes the form in Eq. [S7] only at the end of the argument, to infer the form of the sets of y values. We can relax this assumption as follows. For any continuous score function, consider the preimage of the intervals $[0, S_{(1)}), (S_{(1)}, S_{(2)}), \ldots, (S_{(m-1)}, S_{(m)}), (S_{(m)}, \infty]$ under the function $S(x, \cdot)$ (a function of the second argument with x held fixed), rather than the intervals given explicitly in Lines 12, 20, and 27 of Alg. S1. This algorithm then gives exact coverage for any continuous score function, although it will only be computationally feasible when these preimages can be computed efficiently.

S2. Efficient computation of the full conformal confidence set for ridge regression and Gaussian process regression

A. Ridge regression. When the likelihood of the test input is a function of the prediction from a ridge regression model, it is possible to compute the scores and weights for the full conformal confidence set by fitting $n+1$ models and $O(n \cdot p \cdot |\mathcal{Y}|)$ additional floating point operations, instead of naively fitting $(n+1) \times \mathcal{Y}$ models, as demonstrated in Alg. S2.

For the fluorescent protein design experiments, the TestInputLikelihood subroutine in Alg. S2 computed the likelihood in Eq. [6], that is,

$$\begin{aligned} \textsc{TestInputLikelihood}(a_i + b_i y) &\leftarrow \frac{\exp(\lambda \cdot (a_i + b_i y))}{\cdot \sum_{x \in \mathcal{X}} \exp(\lambda \cdot (C_i + y\mathbf{A}_{-i,n})^T x)}, \\ \textsc{TestInputLikelihood}(a_{n+1}) &\leftarrow \frac{\exp(\lambda \cdot a_{n+1})}{\cdot \sum_{x \in \mathcal{X}} \exp(\lambda \cdot \beta^T x)}, \end{aligned} \qquad \text{[S11]}$$

where the input space $\mathcal{X}$ was the combinatorially complete set of $8,192$ sequences. The TrainInputLikelihood subroutine returned the likelihood under the training input distribution, which is simply equal to to $1/8192$, since training sequences were sampled uniformly from the combinatorially complete data set. See https://github.com/clarafy/conformal-for-design for an implementation.

Computing the test input likelihoods was dominated by the $(n+1) \times |\mathcal{Y}|$ normalizing constants, which can be computed efficiently using a single tensor product between an $(n+1) \times p \times |\mathcal{Y}|$ tensor containing the model parameters, $C_i + y\mathbf{A}_{-i,n}$ and β, and an $|\mathcal{X}| \times p$ data matrix containing all inputs in $\mathcal{X}$. For domains, $\mathcal{X}$, that are too large for the normalizing constants to be computed exactly, one can turn to tractable Monte Carlo approximations.

Algorithm S2 Efficient computation of scores and weights for ridge regression-based feedback covariate shift

Input: training data, $Z_1, \ldots, Z_n$, where $Z_i = (X_i, Y_i)$; test input, X_{n+1}; grid of candidate labels, $\mathcal{Y} \subset \mathbb{R}$; subroutine for test input likelihood, TestInputLikelihood$(\cdot)$, that takes an input's predicted fitness and outputs its likelihood under the test input distribution; subroutine for training input likelihood, TrainInputLikelihood$(\cdot)$.
Output: scores $S_i(X_{n+1}, y)$ and likelihood ratios $v(X_i, Z^y_{-i})$ for $i = 1, \ldots, n+1$, $y \in \mathcal{Y}$.

for $i = 1, \ldots, n$ **do**
  $C_i \leftarrow \sum_{j=1}^{n-1} Y_{-i;j} \mathbf{A}_{-i;j}$
  $a_i \leftarrow C_i^T X_i$
  $b_i \leftarrow \mathbf{A}_{-i;n}^T X_i$
$\beta \leftarrow (\mathbf{X}^T\mathbf{X} + \gamma I)^{-1}\mathbf{X}^T Y$
$a_{n+1} \leftarrow \beta^T X_{n+1}$
for $i = 1, \ldots, n$ **do**
  for $y \in \mathcal{Y}$ **do**
    $S_i(X_{n+1}, y) \leftarrow |Y_i - (a_i + b_i y)|$ ▷ Can vectorize via outer product between $(b_1, \ldots, b_n)$ and vector of all $y \in \mathcal{Y}$.
    $v(X_i; Z_{-i,y}) \leftarrow \textsc{TestInputLikelihood}(a_i + b_i y)/\textsc{TrainInputLikelihood}(X_i)$ ▷ Can vectorize (see commentary on Eq. [S11]).
$S_{n+1}(X_{n+1}, y) \leftarrow |y - a_{n+1}|$
$v(X_{n+1}; Z_{1:n}) \leftarrow \textsc{TestInputLikelihood}(a_{n+1})/\textsc{TrainInputLikelihood}(X_{n+1})$

B. Gaussian process regression. Here we describe how the scores and weights for the confidence set in Eq. [3] can be computed efficiently, when the likelihood of the test input distribution is a function of the predictive mean and variance of a Gaussian process regression model.

For an arbitrary kernel and two data matrices, $\mathbf{V} \in \mathbb{R}^{n_1 \times p}$ and $\mathbf{V}' \in \mathbb{R}^{n_2 \times p}$, let $K(\mathbf{V}, \mathbf{V}')$ denote the $n_1 \times n_2$ matrix where the (i,j)-th entry is the covariance between the i-th row of $\mathbf{V}$ and j-th row of $\mathbf{V}'$. The mean prediction for X_i of a Gaussian process regression model fit to the i-th augmented LOO data set, $\mu^y_{-i}(X_i)$, is then given by

$$\mu^y_{-i}(X_i) = K(X_i, \mathbf{X}_{-i})[K(\mathbf{X}_{-i}, \mathbf{X}_{-i}) + \sigma^2 I]^{-1} Y^y_{-i},$$

and the model's predictive variance at X_i is

$$K(X_i, X_i) - K(X_i, \mathbf{X}_{-i})[K(\mathbf{X}_{-i}, \mathbf{X}_{-i}) + \sigma^2 I]^{-1} K(\mathbf{X}_{-i}, X_i),$$

where the rows of the matrix $\mathbf{X}_{-i} \in \mathbb{R}^{n \times p}$ are the inputs in Z^y_{-i}, $Y^y_{-i} = (Y_{-i}, y) \in \mathbb{R}^n$ is the vector of labels in Z^y_{-i}, and σ^2 is the (unknown) variance of the label noise, whose value is set as a hyperparameter. Note that the mean prediction is a linear function of the candidate value, y, which is of the same form as the ridge regression prediction in Eq. [5]; furthermore, the predictive variance is constant over y. Therefore, we can mimic Alg. S2 to efficiently compute scores and weights by training just $n + 1$ rather than $(n + 1) \times |\mathcal{Y}|$ models.

S3. Additional details and results on designing fluorescent proteins

Features Each sequence was first represented as a length-thirteen vector of signed bits (-1 or 1), each denoting which of the two wild-type parents the amino acid at a site matches. The features for the sequence consisted of these thirteen signed bits, called the first-order terms in the main text, as well as all $\binom{13}{2}$ products between pairs of these thirteen bits, called the second-order interaction terms.

Additional simulated measurement noise. Each time the i-th sequence in the combinatorially complete data set was sampled, for either training or designed data, we introduced additional simulated measurement noise using the following procedure. Poelwijk et al. (8) found that the Walsh-Hadamard transform of the brightness fitness landscape included up to seventh-order statistically significant terms. Accordingly, we fit a linear model of up to seventh-order terms for each of the combinatorially complete data sets, then estimated the standard deviation of the i-th sequence's measurement noise, σ_i, as the residual between its label and this model's prediction. Each time the i-th sequence was sampled, for either training or designed data, we also sampled zero-mean Gaussian noise with standard deviation σ_i and added it to the i-th sequence's label. This was done to simulate the fact that multiple measurements of the same sequence will yield different labels, due to measurement noise.

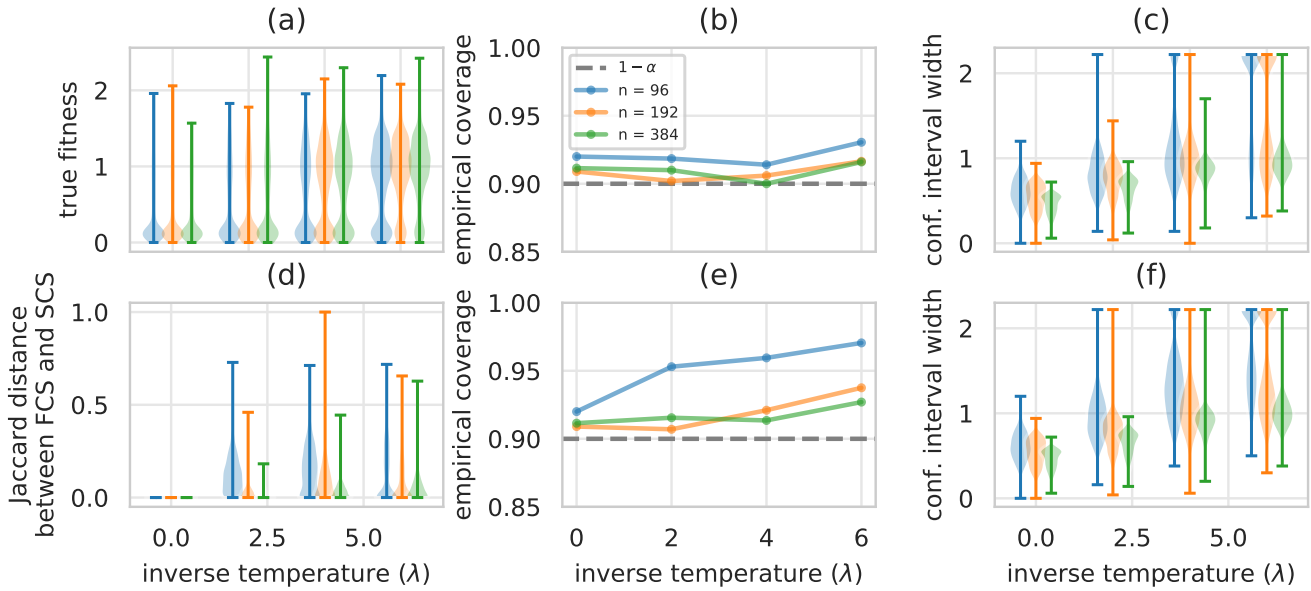


Fig. S2. Quantifying predictive uncertainty for designed proteins, using the red fluorescence data set. (a) Distributions of labels of designed proteins, for different values of the inverse temperature, λ, and different amounts of training data, n. Labels surpass the fitness range observed in the combinatorially complete data set, $[0.025, 1.692]$, due to additional simulated measurement noise. (b) Empirical coverage, compared to the theoretical lower bound of $1 - \alpha = 0.9$ (dashed gray line), and (c) distributions of confidence interval widths achieved by full conformal prediction for feedback covariate shift (our method) over $T = 2000$ trials. (d) Distributions of Jaccard distances between the confidence intervals produced by full conformal prediction for feedback covariate shift and standard covariate shift (1). (e, f) Same as (b, c) but using full conformal prediction for standard covariate shift. In (a), (c), (d), and (f), the whiskers signify the minimum and maximum observed values.

S4. Additional details on AAV experiments

NNK sequence distribution. The NNK sequence distribution is parameterized by independent categorical distributions over the four nucleotides, where the probabilities of the nucleotides are intended to result in a high diversity of amino acids while avoiding stop codons. Specifically, for three contiguous nucleotides corresponding to a codon, the first two nucleotides are sampled uniformly at random from $\{\mathtt{A}, \mathtt{C}, \mathtt{T}, \mathtt{G}\}$, while the last nucleotide is sampled uniformly at random from only $\{\mathtt{T}, \mathtt{G}\}$.

Additional simulated measurement noise. Following Zhu & Brookes et al. (9), the fitness assigned to the i-th sequence was an enrichment score based on its counts before and after a selection experiment, $n_{i,\text{pre}}$ and $n_{i,\text{post}}$, respectively. The variance of this enrichment score for the i-th sequence was estimated as

$$\sigma_i^2 = \frac{1}{n_{i,\text{post}}} \left(1 - \frac{n_{i,\text{post}}}{N_{\text{post}}}\right) + \frac{1}{n_{i,\text{pre}}} \left(1 - \frac{n_{i,\text{pre}}}{N_{\text{pre}}}\right)$$

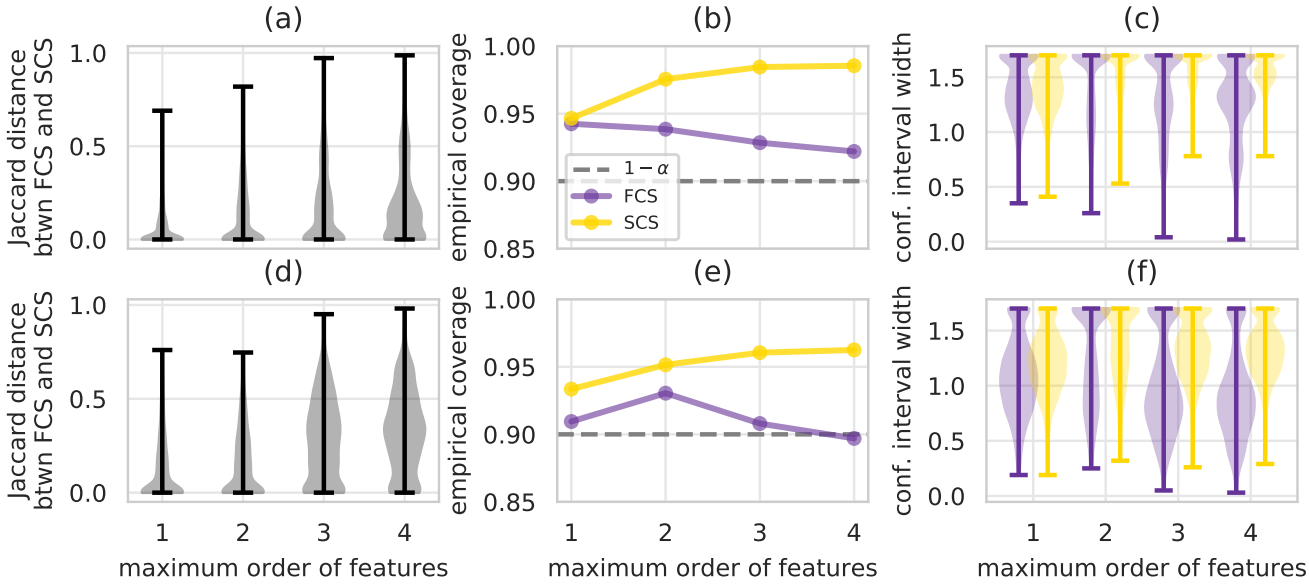


Fig. S3. Quantifying predictive uncertainty for designed proteins using the blue and red fluorescence data sets, for $n = 48$ training data points, $\lambda = 6$, and ridge regression models with features of varying complexity. In particular, the features consist of all interaction terms up to order d between the thirteen sequence sites, where the maximum order, d, is the x-axis of the following subplots. (a) Distributions of Jaccard distances between the confidence intervals produced by conformal prediction for feedback covariate shift (FCS, our method) and standard covariate shift (SCS) (1) for the blue data set over $T = 2000$ trials. (b) Empirical coverage, compared to the theoretical lower bound of $1 - \alpha = 0.9$ (dashed gray line), achieved by conformal prediction for FCS and SCS over those trials. (c) Distributions of confidence interval widths using conformal prediction for FCS and SCS. (d-f) Same as (a-c) but for the red fluorescence data set. In (a), (c), (d), and (f), whiskers signify the minimum and maximum observed values.

where N_{pre} and N_{post} denote the total counts of all the sequences before and after the selection experiment, respectively. Using this estimate, we introduced additional simulated measurement noise to the label of the i-th sequence by adding zero-mean Gaussian noise with a variance of $0.1 \cdot \sigma_i^2$.

Neural network details. As in (9), the neural network took one-hot-encoded sequences as inputs and had an architecture consisting of two fully connected hidden layers, with 100 units each and `tanh` activation functions. It was fit to the $7,552,729$ training data points with the built-in implementation of the Adam algorithm in Tensorflow, using the default hyperparameters and a batch size of 64 for 10 epochs, where each training data point was weighted according to its estimated variance as in (9).

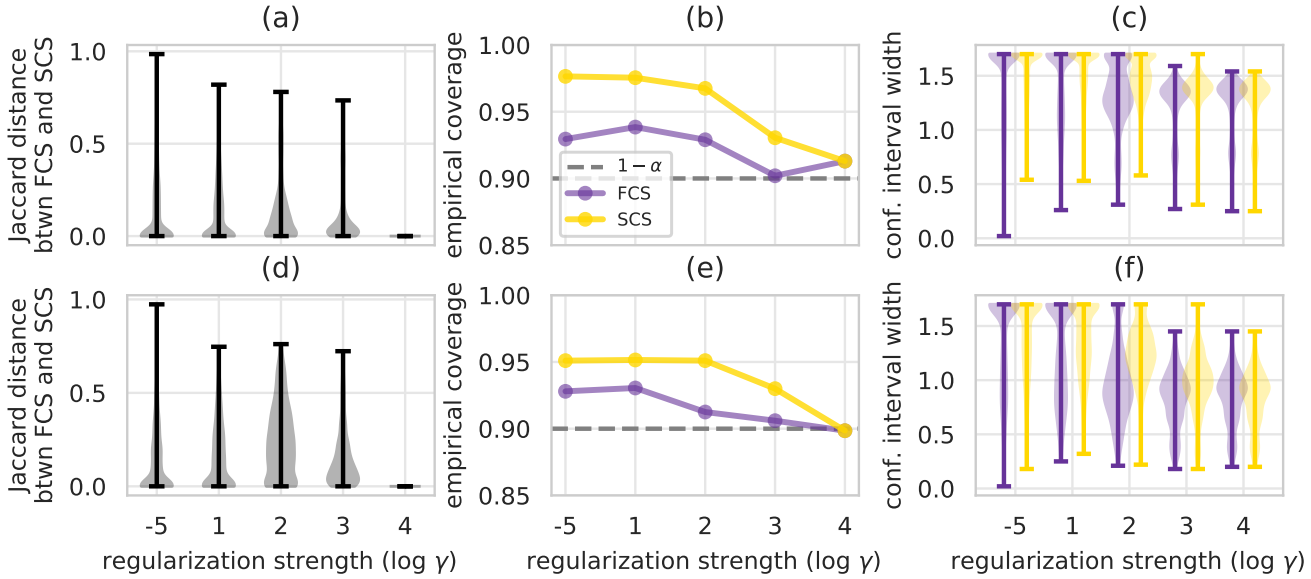


Fig. S4. Quantifying predictive uncertainty for designed proteins using the blue and red fluorescence data sets, for $n = 48$ training data points, $\lambda = 6$, and varying ridge regularization strength, γ. (a) Distributions of Jaccard distances between the confidence intervals produced by conformal prediction for feedback covariate shift (FCS, our method) and standard covariate shift (SCS) (1) for the blue data set over $T = 2000$ trials. (b) Empirical coverage, compared to the theoretical lower bound of $1 - \alpha = 0.9$ (dashed gray line), achieved by conformal prediction for FCS and SCS over those trials. (c) Distributions of confidence interval widths using conformal prediction for FCS and SCS. (d-f) Same as (a-c) but for the red fluorescence data set. In (a), (c), (d), and (f), whiskers signify the minimum and maximum observed values.

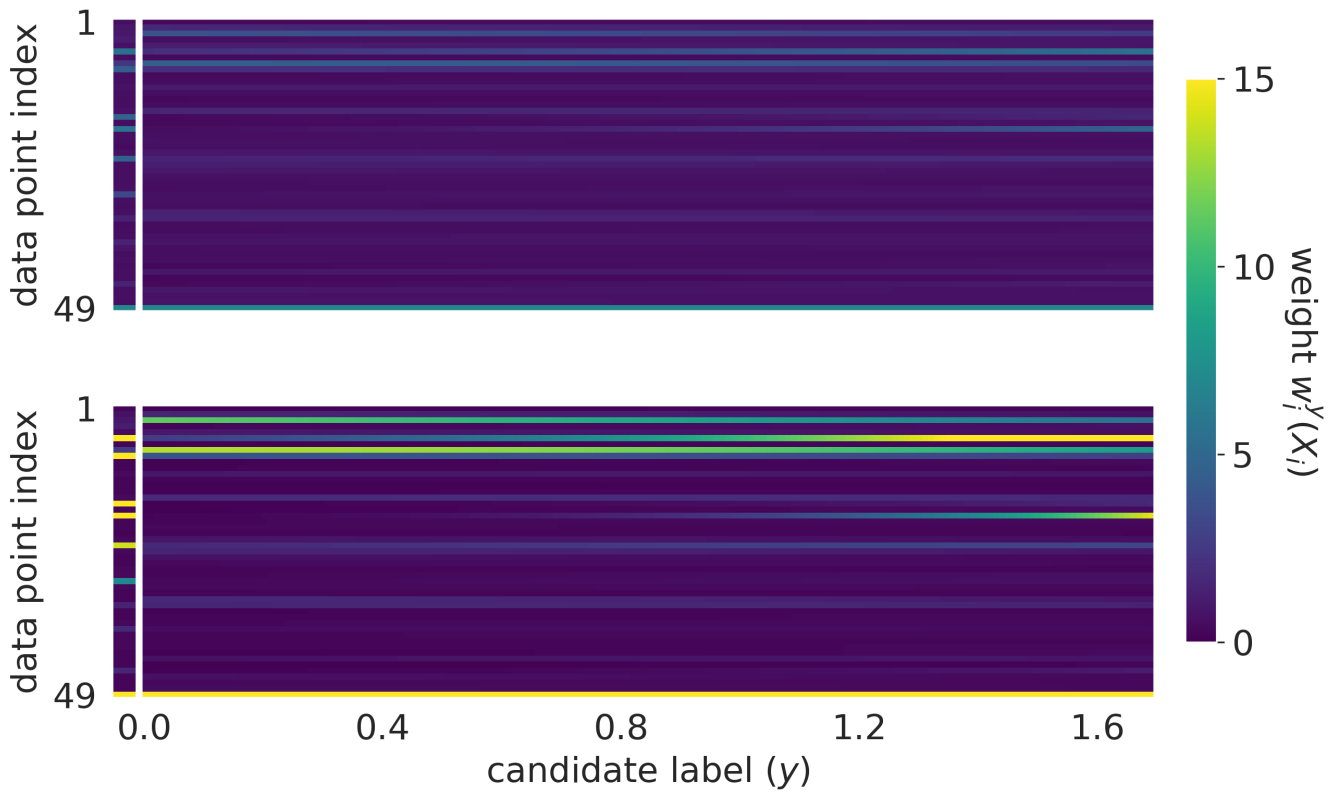


Fig. S5. Comparison between the weights constructed by conformal prediction for feedback covariate shift (FSC, our method) and standard covariate shift (SCS) (1) for one example training data set and resulting designed sequence, for $n = 48$ with the blue fluorescence data set and two different settings of the inverse temperature, λ. Top: For $\lambda = 2$, vector of the $n + 1$ weights prescribed under SCS for the n training data points (data point indices 1 through 48) and the candidate test data points (data point index 49), alongside $(n + 1) \times |\mathcal{Y}|$ matrix of the weights prescribed under FCS for those same $n + 1$ training and candidate test data points. The weight for each of these data points depends on the candidate label, y (x-axis of heatmap), through a linear relationship with y (see Section D). Bottom: same as top but for $\lambda = 6$.

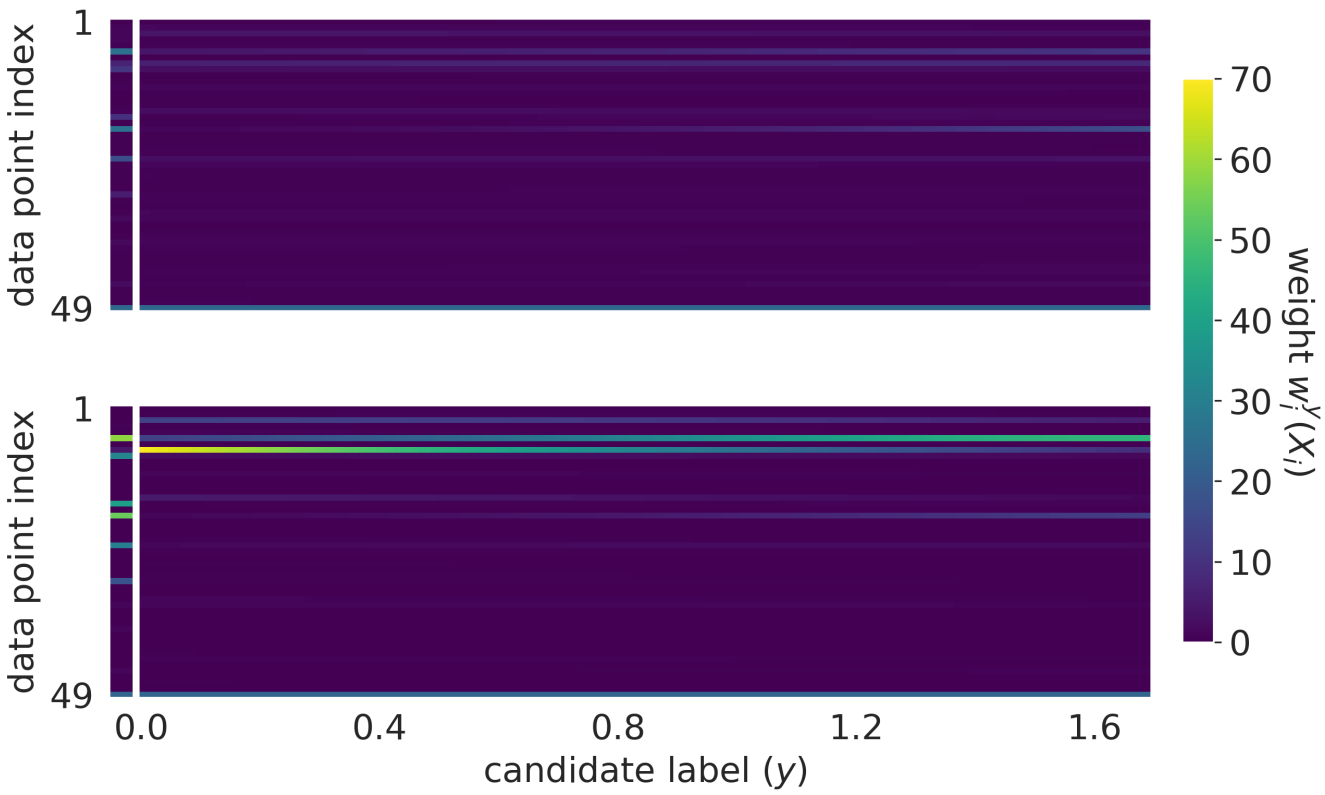


Fig. S6. Comparison between the weights constructed by conformal prediction for feedback covariate shift (FSC, our method) and standard covariate shift (SCS) (1) for one example training data set and resulting designed sequence, for $n = 48$ with the blue fluorescence data set and two different settings of the ridge regularization strength, γ. Top: For $\gamma = 100$, vector of the $n + 1$ weights prescribed under SCS for the n training data points (data point indices 1 through 48) and the candidate test data points (data point index 49), alongside $(n + 1) \times |\mathcal{Y}|$ matrix of the weights prescribed under FCS for those same $n + 1$ training and candidate test data points. The weight for each of these data points depends on the candidate label, y (x-axis of heatmap), through a linear relationship with y (see Section D). Bottom: same as top but for $\gamma = 10$.

References

1. RJ Tibshirani, R Foygel Barber, E Candes, A Ramdas, Conformal prediction under covariate shift in *Advances in Neural Information Processing Systems*. Vol. 32, pp. 2530–2540 (2019).
2. H Papadopoulos, K Proedrou, V Vovk, A Gammerman, Inductive confidence machines for regression in *Machine Learning: European Conference on Machine Learning*. pp. 345–356 (2002).
3. J Lei, M G'Sell, A Rinaldo, RJ Tibshirani, L Wasserman, Distribution-free predictive inference for regression. *Journal of the American Statistical Association* **113**, 1094–1111 (2018).
4. H Shimodaira, Improving predictive inference under covariate shift by weighting the log-likelihood function. *J. Stat. Plan. Inference* **90**, 227–244 (2000).
5. M Sugiyama, KR Müller, Input-dependent estimation of generalization error under covariate shift. *Stat. & Decis.* **23**, 249–279 (2005).
6. M Sugiyama, M Krauledat, KR Müller, Covariate shift adaptation by importance weighted cross validation. *J. Mach. Learn. Res.* **8**, 985–1005 (2007).
7. AN Angelopoulos, S Bates, A gentle introduction to conformal prediction and distribution-free uncertainty quantification. arXiv preprint 2107.07511 (2021).
8. FJ Poelwijk, M Socolich, R Ranganathan, Learning the pattern of epistasis linking genotype and phenotype in a protein. *Nat. Commun.* **10**, 4213 (2019).
9. D Zhu, et al., Optimal trade-off control in machine learning-based library design, with application to adeno-associated virus (aav) for gene therapy. bioRxiv preprint 2021.11.02.467003 (2021).