Algorithmic - Complexity Analysis

BETA-OPTIM : UNCERTAINTY QUANTIFICATION METHOD FOR MULTI-OUTPUT REGRESSION PROBLEMS

Compulsory Project

Jaad BELHOUARI - Fatima-Zahra HANNOU - Aymane MIMOUN

M2 Data Science : Santé, Finance et Assurance

May 21, 2025

Abstract

Multi-output regression models often require simultaneous prediction intervals to ensure valid uncertainty quantification across all output dimensions. While Conformal Prediction (CP) provides a distribution-free framework for uncertainty quantification, existing methods typically focus on one-dimensional outputs. In this paper, we introduce **Fast Beta-Optim**, a novel approach that optimizes local conformal prediction intervals to achieve valid *simultaneous* coverage. By combining conformal prediction with an optimization framework, Fast Beta-Optim ensures that the true outputs fall within the prediction intervals with high probability. We compare **Fast Beta-Optim** to **Max-Rank**, an existing method, and demonstrate that **Fast Beta-Optim** achieves similar coverage guarantees and prediction interval sizes while offering execution times comparable to **Max-Rank**, but with the added benefit of estimating the local error (via β_{optim}), a feature not captured by Max-Rank. We also present an open-source R package to facilitate the practical use of our methods.

1 Introduction

1.1 Problem Statement and Motivation

Black-box machine learning models, particularly in high-risk domains such as medical diagnostics or engineering systems, require not only accurate predictions but also reliable uncertainty quantification. Understanding *how confident* a model is about its predictions is crucial to prevent critical failures and to enable robust and trustworthy decision making.

Conformal prediction (or conformal inference) provides a statistically principled approach to this challenge. Unlike traditional methods, it makes no assumptions about the model and delivers prediction intervals with **valid coverage guarantees**—ensuring that, on average, the true value falls within the predicted range at the desired confidence level. This makes it an especially appealing tool for uncertainty quantification in complex, data-driven systems.

In the context of multi-output regression problems—where multiple, possibly dependent, outputs must be predicted simultaneously—quantifying uncertainty becomes even more challenging. One must account not only for the individual uncertainty of each output -**local uncertainty**- but also for the joint behavior of the outputs as a whole -**global or simultaneous uncertainty**-. Addressing this, our paper introduces a new method called **Beta-Optim**, specifically designed to build **simultaneous prediction sets** in such settings, mixing conformal prediction framework and an optimization problem.

Our approach is particularly suited for applications involving the prediction of high-dimensional structured outputs, such as curves, time series, or spatial profiles. These outputs can be described by a set of predicted points or by coefficients in a functional basis. Accurate joint uncertainty quantification in such cases enables various advanced tasks: performing *multiple hypothesis testing* across dimensions, detecting *outliers* or *anomalous patterns*, and supporting *reliable decision-making* under uncertainty.

1.2 Challenges and Objectives

Quantifying uncertainty in multi-output regression is not only theoretically demanding but also practically complex. The main challenge lies in designing methods that can efficiently produce valid and informative prediction sets that jointly capture uncertainty across all output dimensions—especially when those outputs are high-dimensional or structured.

To address this, we explore and compare three conformal prediction approaches specifically adapted for multi-output settings. Two of them—**Beta-Optim** and **Fast Beta-Optim**—are novel methods proposed in this work. Both extend the conformal framework by introducing optimization-based strategies to construct **simultaneous prediction sets** that are valid and efficient, even when the outputs are correlated or vary in importance. The third method, **MaxRank**, is adapted from recent literature and serves as a relevant benchmark for comparison.

In addition to methodological innovation, we aim to provide tools that are accessible and reproducible. Alongside this paper, we release an **open-source R library**—built with a performant C++ backend—offering a flexible implementation of all three methods. This package is designed to be user-friendly, extensible, and ready for integration into real-world machine learning workflows.

Our objective in this paper is to analyze the **computational complexity** of each method with respect to the **calibration set size**. This aspect is particularly important in high-dimensional settings, where scalability plays a crucial role in the method's practical applicability. By examining the evolution of execution time as the number of calibration points increases, we aim to provide **empirical validation of the theoretical complexity** and offer insights into the computational efficiency of the different approaches.

2 Beta-Optim Method

2.1 Brief introduction to Conformal Prediction

This paper explores Conformal Prediction's theoretical foundations, algorithmic implementations, and practical applications, enhancing prediction reliability in data-driven contexts, especially for **regression problems**. Therefore, before going through our Beta-Optim method, let's briefly introduce the conformal prediction framework, for those who are new to the domain. For readers interested in a deeper understanding of conformal prediction, we recommend the following reference: *A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification* [1] from Anastasios N. Angelopoulos and Stephen Bates.

2.1.1 Idea behind conformal prediction

In the regression case, suppose we have a trained model $\hat{f}$ that predicts a continuous output y from input x. Given a separate **calibration** dataset $(X_1, Y_1), ..., (X_n, Y_n)$ not used during training, we compute a *score function* $s(x, y)$ that reflects how well the prediction $\hat{f}(x)$ matches the true output y. A common choice is the **absolute error**: $s(x, y) = |\hat{f}(x) - y|$.

We then calculate the $(1 - \alpha)$ quantile $\hat{q}$ of the calibration scores:

$$\hat{q} = \text{Quantile}_{1-\alpha}(|\hat{f}(X_i) - Y_i|), \quad i = 1, ..., n.$$

Finally, for a new test input X_{test}, we form the prediction interval:

$$\mathcal{C}(X_{\text{test}}) = [\hat{f}(X_{\text{test}}) - \hat{q},\ \hat{f}(X_{\text{test}}) + \hat{q}].$$

This interval satisfies the following guarantee:

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

meaning the true output will fall within the interval with high probability—averaged over the randomness of the data (marginal coverage).

2.1.2 General method

Conformal prediction can be applied beyond basic regression. It only requires a pre-trained model and a score function measuring prediction fit. The general steps are:

1. Choose a score function $s(x, y)$ that reflects prediction error.
2. Compute calibration scores $s_i = s(X_i, Y_i)$ using a separate calibration set.
3. Estimate the $(1 - \alpha)$ quantile $\hat{q}$ of these scores.
4. Define prediction sets for new inputs as $\mathcal{C}(X_{\text{test}}) = \{y : s(X_{\text{test}}, y) \leq \hat{q}\}$.

The **[Figure 2]** in the Appendices illustrates this procedure.

2.1.3 Coverage Guarantee

Let (X_i, Y_i) and $(X_{\text{test}}, Y_{\text{test}})$ be exchangeable 6.2 samples, and let $\hat{q}$ and $\mathcal{C}(X_{\text{test}})$ be defined as above. Then conformal prediction guarantees:

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

as shown by Vovk, Gammerman, and G. Shafer [4]. This makes conformal prediction a powerful and versatile tool for uncertainty quantification.

2.2 Beta-optim strategy

For the remainder of this paper, let p denote the dimension of the target space $\mathcal{Y} \subset \mathbb{R}^p$. We denote by $\mathcal{C}^i_\beta(x)$ the local prediction interval at level $1-\beta$ for the i^{th} component of the output, constructed using the conformal prediction procedure.

We propose a new method for constructing *simultaneous* prediction intervals by solving an optimization problem. The approach is inspired from **Bonferroni's strategy** [2], where the goal is to identify a *local error level* β such that the resulting one-at-a-time prediction intervals collectively achieve a desired *simultaneous coverage* level of $1-\alpha$. To this end, we define the empirical simultaneous coverage function $simcov(\beta)$ over the calibration set $\mathcal{D}_{\text{cal}}$ as:

$$simcov(\beta) = \frac{\sum\limits_{(x,y)\in\mathcal{D}_{cal}} \prod\limits_{i\in\{1,\dots,p\}} \mathbb{1}\{y_i \in \mathcal{C}^i_\beta(x)\}}{|\mathcal{D}_{cal}|}$$

which estimates the proportion of calibration points for which all p output components are simultaneously covered by their respective local intervals. Then, we look for the β_{optim} value that minimizes the following loss function J :

$$\mathcal{J}(\beta) = |simcov(\beta) - (1-\alpha)|$$

In other words, the local β_{optim} error value in $[0,1]$ that leads to $1-\alpha$ global error.

Finally, we construct the prediction intervals for a new test input X_{test} independently for each output dimension as follows:

$$\mathcal{C}^i_\beta(X_{test}) = [\hat{f}^i(X_{test}) - \hat{q}^i_{\beta_{optim}}, \hat{f}^i(X_{test}) + \hat{q}^i_{\beta_{optim}}] \quad \forall\ i \in \{1,...,p\}$$

where $\hat{f}^i(X_{\text{test}})$ is the model's prediction for the i^{th} output dimension, and $\hat{q}^i_{\beta_{\text{optim}}}$ is the calibrated quantile corresponding to the optimized local error level β_{optim}.

This construction ensures the following simultaneous coverage guarantee:

$$\mathbb{P}\left(\bigcap_{i=1}^{p} \left\{Y^i_{\text{test}} \in \mathcal{C}^i_\beta(X_{\text{test}})\right\}\right) \geq 1-\alpha,$$

which means that the entire ground truth vector $Y_{\text{test}} \in \mathbb{R}^p$ lies within the corresponding prediction intervals with probability at least $1-\alpha$.

3 Theoretical Analysis of Complexity

Before diving into the details, we choose—for simplicity—to fix the score function as the **absolute error**: $s(x, y) = |\hat{f}(x) - y|$ in our analysis. Nevertheless, the Beta-Optim procedure remains **compatible** with other score functions, which can be used to construct valid simultaneous prediction intervals.

3.1 Complexity of Beta-Optim

Let n be the number of examples in the calibration set, and p be the dimension of the target space. The time complexity of the Beta-Optim method is given by:

$$\mathcal{O}(p \cdot n \log n)$$

This complexity arises from the need to compute the local prediction intervals for each of the p output dimensions across all n calibration examples.

3.1.1 Detailed Steps

1. **Score Calculation**: For each calibration example (X_i, Y_i), compute the score $s(X_i, Y_i)$ across all p output dimensions. This results in $n \times p$ score computations.
   - *Complexity:* $\mathcal{O}(n \cdot p)$
2. **Quantile Estimation**: For each dimension $i \in \{1, \ldots, p\}$, estimate the $(1 - \beta)$ quantile $\hat{q}^i_\beta$ from the corresponding score distribution. This requires sorting the n scores for each dimension.
   - *Complexity:* $\mathcal{O}(p \cdot n \log n)$
3. **Optimization**: Solve for the optimal value of β that minimizes the loss function $\mathcal{J}(\beta)$, which involves evaluating the simultaneous coverage function $simcov(\beta)$ for several β values. For each candidate β, the following steps are performed:
   - Retrieve the quantiles at position $\lceil (1-\beta)n \rceil$ for each dimension from the sorted score matrix.
   - Compute the corresponding prediction intervals (lower and upper bounds).
   - Check whether each calibration point falls within all p intervals.

   Since the function $simcov(\beta)$ is **monotonically decreasing** (see Section **??**), a **dichotomy search** can be employed using a fixed number of iterations n_{iter}. The complexity of this optimization step becomes:
   - *Complexity:* $\mathcal{O}(n \cdot p \cdot n_{\text{iter}})$

   With a guaranteed precision of $\frac{1}{2^{n_{\text{iter}}}}$ for the final value β_{optim}.

Combining all steps, the total complexity of the `Beta-Optim` method is dominated by the quantile estimation step:

$$\mathcal{O}(n \cdot p + p \cdot n \log n + n \cdot p \cdot n_{\text{iter}}) = \mathcal{O}(p \cdot n \log n + n \cdot p \cdot n_{\text{iter}})$$

Assuming n_{iter} is small (≈ 10) and constant, the overall complexity simplifies to:

$$\mathcal{O}(p \cdot n \log n)$$

as only 10 iterations are sufficient to achieve a precision of 10^{-3} using a dichotomic search.

3.2 Complexity of Fast Beta-Optim

The *Fast Beta-Optim* method employs more efficient computational strategies to reduce runtime, while maintaining the same theoretical complexity as the standard Beta-Optim approach:

$$\mathcal{O}(p \cdot n \log n)$$

Although the asymptotic complexity remains unchanged, Fast Beta-Optim offers significant **practical improvements** by optimizing how the objective function is evaluated. Specifically, it precomputes and stores the *maximum rank values* of the score vectors across the calibration set, thereby avoiding redundant computations during optimization. This makes Fast Beta-Optim particularly well-suited for **high-dimensional output spaces**, where computational efficiency becomes critical.

3.2.1 Detailed Steps

1. **Score Calculation**: As in the standard Beta-Optim, compute the score $s(X_i, Y_i)$ for each of the p output dimensions across all n calibration examples.
   - Complexity: $\mathcal{O}(n \cdot p)$
2. **Quantile Estimation and Max-Rank Preprocessing**: For each output dimension, sort the scores to estimate the quantiles. Then, for each calibration point i, compute the maximum rank value across all dimensions. This involves:
   - Sorting scores for each dimension: $\mathcal{O}(p \cdot n \log n)$
   - Computing ranks: $\mathcal{O}(p \cdot n \log n)$ (can be done during sorting)
   - Extracting maximum rank per example: $\mathcal{O}(p \cdot n)$
   - **Total**: $\mathcal{O}(p \cdot n \log n)$
3. **Optimization**: As in Beta-Optim, solve the optimization problem to find the optimal β. However, thanks to the precomputed maximum ranks, the evaluation of the simultaneous coverage function $simcov(\beta)$ becomes significantly faster:
   - For a candidate β, check whether $r^i_{\max} \leq \lceil (1-\beta)n \rceil$ for each calibration point i

   The *simcov* function is thus computed as:

$$simcov(\beta) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}\left\{ r^i_{\max} \leq \lceil (1-\beta)n \rceil \right\}$$

   - Complexity: $\mathcal{O}(n_{\text{iter}} \cdot n)$, where n_{iter} is the number of iterations in the optimization procedure (e.g., dichotomy).

3.2.2 Overall Complexity:

Combining all steps, the total complexity of Fast Beta-Optim is:

$$\mathcal{O}(n \cdot p + p \cdot n \log n + n_{\text{iter}} \cdot n) \approx \mathcal{O}(p \cdot n \log n)$$

where the optimization step is relatively lightweight compared to the sorting steps.

3.3 Complexity of Max Rank

According to the paper *"Max-Rank: Efficient Multiple Testing for Conformal Prediction"* by Timans et al. (2025) [3], the algorithm used to compute the prediction intervals is detailed in Appendix 6.3.

As in previous methods, the time complexity of the Max Rank procedure is:

$$\mathcal{O}(p \cdot n \log n)$$

We invite the reader to consult the algorithm in the appendix for a detailed step-by-step explanation and complexity justification.

3.4 Conclusion

While all three methods—`Beta-Optim`, `Fast Beta-Optim`, and `Max Rank`—share the same theoretical time complexity of $\mathcal{O}(p \cdot n \log n)$, their **practical performance** can vary significantly. This discrepancy arises from algorithmic optimizations that affect constant factors hidden in the asymptotic notation.

In particular, **Fast Beta-Optim** improves runtime by precomputing maximum rank values, which streamlines the optimization process. This makes it especially effective in high-dimensional settings, where reducing redundant operations is crucial.

However, theoretical complexity analysis alone is not sufficient to fully distinguish between the methods. All methods present the same worst-case scaling with respect to n and p, but their **empirical behavior** reveals valuable differences. For example, while Max Rank avoids optimizing β, its implementation details may still affect runtime performance compared to Fast Beta-Optim.

To draw meaningful conclusions about the practical usability of these methods, especially in large-scale or real-time settings, we must complement the theoretical study with an in-depth empirical evaluation of runtime behavior, as presented in the next section. Such an evaluation allows us to quantify constant factors, identify implementation bottlenecks, and better assess the true computational efficiency of each method under realistic conditions.

4 Practical evaluation of the complexity

4.1 Methodology

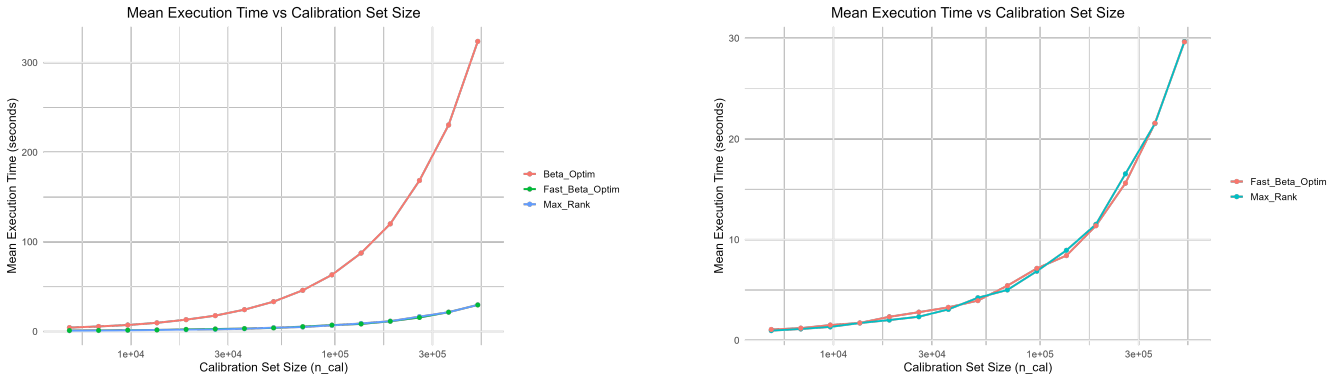
To evaluate the practical complexity of our uncertainty quantification methods, we simulate the execution time under varying calibration set sizes. Specifically, we consider a multi-output regression setting with a fixed underlying model and generate synthetic datasets that mimic real-world calibration scenarios, with a fixed value of $p = 24$ for the number of dimensions of our target.

For each experiment, we vary the calibration set size n_{cal} across a predefined range (e.g., from $2 \cdot 10^3$ to $9 \cdot 10^5$), while keeping the training and test sets **fixed**. For each value of n_{cal}, we perform 10 repetitions to account for the variability in runtime due to stochasticity in data splitting or internal method randomness.

Each method—**Beta-Optim**, **Fast Beta-Optim**, and **Max-Rank**—is applied independently on the same calibration data to ensure fairness. We record the execution time required to fit the uncertainty model (i.e., to perform the calibration step) and compute both the mean and variance of the execution time over repetitions.

4.2 Simulation Results

We compare the computational efficiency of the three methods: **Beta-Optim**, **Fast Beta-Optim**, and **Max-Rank**. The results, shown in Figure 1, illustrate the evolution of execution time with respect to the calibration set size.



(a) Execution time comparison for all methods.

(b) Zoom on Fast Beta-Optim vs Max-Rank

Figure 1: Comparison of execution time as a function of calibration set size.

From these plots, we clearly observe a significant computational gain when switching from the standard **Beta-Optim** approach to its more efficient variants. While all methods maintain similar theoretical complexity, the runtime reduction is substantial.

Notably, **Fast Beta-Optim** achieves up to an order of magnitude improvement in execution time compared to the baseline method **Beta-Optim**, especially as the calibration set size increases. This highlights the practical advantages of exploiting algorithmic shortcuts—such as avoiding redundant score evaluations and reusing rank information—to reduce unnecessary computations.

These results validate the design choices behind the fast variants, confirming that they are particularly well-suited for large-scale or high-dimensional applications.

4.3 Estimating the Exponent of Computational Complexity

To better understand how the execution time scales with the calibration set size n_{cal}, we aim to estimate the exponent k in the following relationship:

$$\text{Execution Time} \approx C \cdot n_{\text{cal}}^{k},$$

where C is a constant and k characterizes the complexity.

By taking the logarithm on both sides, we linearize the expression:

$$\log(\text{Execution Time}) = \log(C) + k \cdot \log(n_{\text{cal}}).$$

This transformation allows us to estimate the exponent k by fitting a simple **linear regression** model:

$$\log(\text{Execution Time}) \sim \log(n_{\text{cal}}).$$

We perform this analysis separately for each method to compare their respective computational behaviors. The slope of the fitted regression line corresponds to the estimated exponent k for each method. A log-log plot **[Figure 3]** is also provided to visually validate the linear trend and confirm the polynomial relationship between execution time and calibration set size.

The resulting exponents provide insight into the scalability of each approach. For instance, a value of $k \approx 1$ would indicate linear scaling, while $k \approx \log n$ would suggest quasi-linear complexity. The obtained coefficients are presented in **[Table 1]**.

Method	**exponent k**
Beta-Optim	1.00
Fast Beta-Optim	**0.95**
Max Rank	**0.94**

Table 1: Comparison of the log coefficients

However, during our analysis, we encountered a notable issue when estimating the exponent k: the presence of **very small** calibration set sizes (e.g., $n_{\text{cal}} = 2000$) introduced **significant bias** in the regression model. These points exhibited disproportionately low execution times, likely due to implementation-level optimizations (such as caching, memory access efficiency, or negligible overhead in small loops), which do not accurately reflect the true **asymptotic behavior** of the algorithms.

To mitigate this, we **restricted** the regression analysis to calibration sizes **greater than** 50,000, ensuring that the runtime measurements better capture the dominant complexity terms of each method. After filtering, we observed that the estimated exponent k increased for all methods and exceeded 1—confirming the expected super-linear growth (i.e., $k > 1$) consistent with the theoretical complexity of $\mathcal{O}(p \cdot n \log n)$.

This highlights a key practical consideration: accurately estimating the computational complexity exponent from empirical data requires **sufficiently large** inputs to avoid **underestimating** the true growth rate. Thus, when comparing methods, it is essential to consider runtime trends at scale in order to draw meaningful conclusions about their computational efficiency.

5 Conclusion and Perspectives

In this work, we presented a detailed analysis and comparison of multiple uncertainty quantification methods within the conformal prediction framework, with a focus on multi-output regression tasks. Our main contributions and findings can be summarized as follows:

- We evaluated and compared the performance of three key methods—**Beta-Optim**, **Fast Beta-Optim**, and **Max-Rank**—in terms of predictive coverage, interval width, and computational efficiency.
- Among these, **Fast Beta-Optim** emerges as a particularly effective compromise. It achieves predictive performance comparable to both Beta-Optim and Max-Rank, offering **similar coverage guarantees and interval sizes**. However, it also **matches the execution time** of the highly efficient Max-Rank method, while providing an **additional benefit**: the estimation of the **optimal local error level**, β_{opt}, which is not available with Max-Rank.
- While the original **Beta-Optim** method remains more computationally demanding, it offers greater modeling flexibility. In particular, it opens promising directions for future work by allowing us to search for an optimal β not in $\mathbb{R}$ but in a higher-dimensional space $\mathbb{R}^d$, where $1 \leq d \leq p$. This would enable the construction of more informative and refined prediction sets, accounting for interdependencies among output dimensions.

Perspectives. One particularly appealing extension would involve developing a vector-valued generalization of Beta-Optim, where $\beta \in \mathbb{R}^d$ is optimized over multiple dimensions. This extension could yield prediction regions that are not only sharper but also tailored to the underlying correlation structure of the outputs. However, such an approach would require a deeper theoretical and empirical analysis of the objective function, particularly in the context of multidimensional optimization.

Overall, this study underscores the practicality of Fast Beta-Optim for large-scale problems, while also highlighting the potential of more expressive optimization strategies for uncertainty quantification in high-dimensional settings.

References

[1] Anastasios N. Angelopoulos and Stephen Bates. *A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification*. 2022. arXiv: 2107.07511 [cs.LG]. URL: https://arxiv.org/abs/2107.07511.

[2] "Bonferroni correction". In: *Wikipedia* (). URL: https://en.wikipedia.org/wiki/Bonferroni_correction.

[3] Alexander Timans et al. *Max-Rank: Efficient Multiple Testing for Conformal Prediction*. 2025. arXiv: 2311.10900 [stat.ME]. URL: https://arxiv.org/abs/2311.10900.

[4] A. Gammerman V. Vovk and G. Shafer. "Algorithmic Learning in a Random World." In: *Springer* (2005).

6 Appendices

6.1 Conformal Prediction Steps

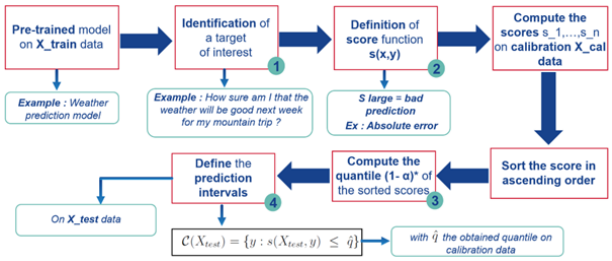
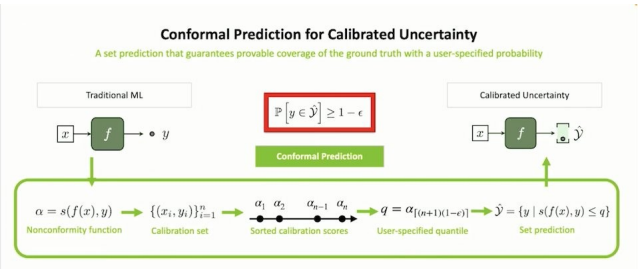


Figure 2: General steps for conformal prediction framework

6.2 Exchangeability property

Let $s(X_{n+1}, Y_{n+1})$ be the computed score for an element of our prediction set. The exchangeability property states that $s(X_{n+1}, Y_{n+1})$ has a probability $\frac{1}{n+1}$ of falling between any pairs of successive scores $s_i = s(X_i, Y_i)$ and $s_{i+1} = s(X_{i+1}, Y_{i+1})$ for all $i \in \{1, ..., n-1\}$ from the calibration set.



6.3 Max Rank Algorithm

Algorithm 1 Multiple Testing Correction via Max-Rank

1: **Input:** Hypothesis tests $k = 1, \ldots, m$ and associated empirical nulls $s_1, \ldots, s_m$ of equal size n.
2: **Output:** Adjusted quantiles with FWER control at level α as $(\hat{Q}_{1-\alpha,1}, \ldots, \hat{Q}_{1-\alpha,m}) \in \mathbb{R}^m$.
3: **Procedure:**
4: Collect empirical nulls in a matrix of test statistics $S \in \mathbb{R}^{n \times m}$, and their respective column-wise ranks in a matrix of rank statistics $R \in \mathbb{N}^{n \times m}$.
5: Apply the l_∞-norm row-wise to the ranks. Let $r_i = (r_{i,1}, \ldots, r_{i,m})$ be the i-th row of R, then $r_{\max} = (\|r_1\|_\infty, \ldots, \|r_n\|_\infty)^T = (\max_{1 \le k \le m} |r_{1,k}|, \ldots, \max_{1 \le k \le m} |r_{n,k}|)^T \in \mathbb{N}^n$.
6: Compute the rank $r_{\max} = \hat{Q}(1-\alpha; r_{\max})$ as the $\lceil (n+1)(1-\alpha)/n \rceil$-th empirical quantile of $r_{\max}$.
7: Sort S column-wise ascending such that $s_{1,k} < \ldots < s_{n,k}$ for every $k \in [m]$.
8: Select the test statistic $s_{r_{\max},k}$ at rank $r_{\max}$ column-wise in S as the adjusted quantile $\hat{Q}_{1-\alpha,k}$ for every test dimension $k \in [m]$.
9: **End Procedure**

6.4 Coefficient k for log log plot

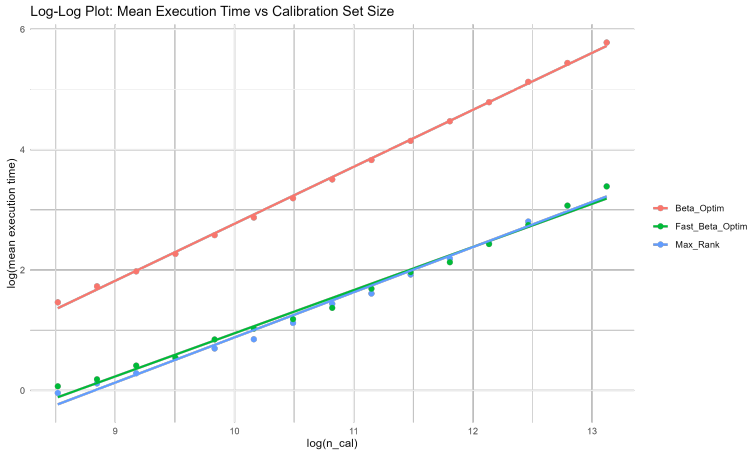


Figure 3: Log-Log Plot: Mean Execution Time vs Calibration Set Size