Prediction-Powered Inference Framework Under Covariate Shift

PHS 7065 Fall 2024 - Final Report

Instructor: Dr. Xuan Wang
Group Members: Yingjia Wei, Haojia Li

2024-12-14

Contents

1 Introduction

A substantial proportion of missing data is a common challenge in Electronic Health Records (EHR) data analysis, which has the potential to undermine the validity of research findings (Sterne et al. 2009). Machine learning algorithms can be employed to predict missing values based on observed data. However, questions remain about the validity of conclusions drawn from such predicted data. To address this concern, Angelopoulos et al. (2023) proposed Prediction-Powered Inference (PPI), a framework designed to enable provably valid statistical inference when predictions are used as data. The PPI method leverages a gold-standard dataset, consisting of features paired with observed outcomes, to quantify and correct errors made by the machine-learning algorithm on the unlabeled dataset. This allows the constructed confidence intervals to achieve the best between two extremes: using only labeled data and relying solely on predicted unlabeled data: (1) the intervals are valid, as they contain the true value of the estimand of interest, and (2) they are more efficient, with narrower widths achieved by incorporating information from the larger sample size of unlabeled data.

Despite its advantages, the PPI framework has notable limitations. When the provided predictions are inaccurate, the constructed intervals can perform worse than the "classical intervals" derived solely from labeled data. To improve the statistical efficiency of PPI, Angelopoulos, Duchi, and Zrnic (2024) developed Efficient Prediction-Powered Inference (PPI++), which incorporates a weighting parameter, λ, to minimize the asymptotic variance of the prediction-powered estimator. Additionally, Zrnic and Candès (2024) proposed Cross-Prediction-Powered Inference (Cross-PPI), an extension of PPI that ensures validity by splitting the labeled data to train the predictive model.

Our study aims to evaluate the performance of the PPI, PPI++, Cross-PPI, and the combination of PPI++ and Cross-PPI (Cross-PPI++) methods through a comprehensive simulation study. Specifically, we will examine their statistical properties, including the validity and efficiency of the constructed confidence intervals, under various simulated scenarios. Initially, we will evaluate these methods in settings where the labeled and unlabeled data are drawn from the same distribution. Furthermore, we will expand our focus to include the common challenge in EHR data analysis known as covariate shift, where the distribution of labeled data differs from that of unlabeled data. To address this, we will incorporate the density ratio into the PPI methods.

In the simulation study, we will focus on the simplest estimand, the mean outcome, and evaluate the performance of these methods across various settings, including different ratios of labeled to unlabeled data, varying levels of predictive model accuracy, diverse feature distributions, and degrees of covariate shift. The simulation will be replicated 100 times, and performance metrics will include the coverage probability of confidence intervals, the mean width of intervals. By systematically exploring these scenarios, we aim to identify the conditions under which each method performs optimally or encounters limitations, offering practical guidance for their application in real-world EHR datasets.

2 Methods

2.1 PPI

Below are the the fundamental notations used in the PPI framework:

- Let $(X, Y) \in (\mathcal{X} \times \mathcal{Y})^n$ denote the labeled or the gold-standard dataset, where $X = (X_1, \ldots, X_n)$ and $Y = (Y_1, \ldots, Y_n)$.
- Similarly, let $(\tilde{X}, \tilde{Y}) \in (\mathcal{X} \times \mathcal{Y})^N$ denote the unlabeled dataset, where the outcomes $\tilde{Y}$ are not observed.
- In this section, we assume that (X, Y) and $(\tilde{X}, \tilde{Y})$ are independently and identically distributed (i.i.d.) samples from a common distribution, $p(x, y)$.
- We have a prediction rule, $f : X \to Y$ that is independent of the observed data. Thus, $f(X_i)$ denote the predictions for the labeled data and $f(\tilde{X}_i)$ denote the predictions for the unlabeled data.

The goal of PPI is to construct a confidence interval $\mathcal{C}^{PP}$ for the estimand θ. We will focus on the mean outcome as the estimand in our study. Specifically, we are interested in the mean outcome in the unlabeled data, $E[\tilde{Y}]$. The key conceptual innovation of PPI lies in the introduction of measure of fit m_θ and rectifier Δ_θ:

- Measure of fit m_θ is computed on the predictions for the unlabeled data $(\tilde{X}, f(\tilde{X}))$ and quantifies how close the estimate of θ it to its true value.
- Rectifier Δ_θ is defined as the difference of the measure of fit m_θ computed using the labeled data and its predictions $(X, Y, f(X))$. If the predictions are perfect, the rectifier is equal to zero.

The PPI method computes a confidence interval as $\mathcal{C}_\alpha^{PP} = \{\theta \text{ such that} |m_\theta + \Delta_\theta| \leq t_\theta(\alpha)\}$, where $t_\theta(\alpha)$ is a constant depending on the error level α.

When the estimand of interest is the mean outcome in unlabeled data, the algorithm for constructing the confidence interval $\mathcal{C}_\alpha^{PP}$ at error level $\alpha \in (0, 1)$ is as follows:

1. Point estimate of θ by PPI: $\hat{\theta}^{PP} \leftarrow \tilde{\theta}^f - \hat{\Delta} := \frac{1}{N}\sum_{i=1}^N f(\tilde{X}_i) - \frac{1}{n}\sum_{i=1}^n (f(X_i) - Y_i)$
2. Empirical variance of estimate based on unlabeled data: $\hat{\sigma}_f^2 \leftarrow \frac{1}{N}\sum_{i=1}^N (f(\tilde{X}_i) - \tilde{\theta}^f)^2$
3. Empirical variance of the rectifier: $\hat{\sigma}_\Delta^2 \leftarrow \frac{1}{n}\sum_{i=1}^n (f(X_i) - Y_i - \hat{\Delta})^2$
4. Normal approximation: $t_\alpha \leftarrow z_{1-\alpha/2}\sqrt{\frac{\hat{\sigma}_\Delta^2}{n} + \frac{\hat{\sigma}_f^2}{N}}$
5. Construct prediction-powered confidence set: $\mathcal{C}_\alpha^{PP} = \left(\hat{\theta}^{PP} \pm t_\alpha\right)$

2.2 PPI++

The original PPI constructs a confidence set for θ^* by searching through all possible values of θ.

$$\begin{aligned}
\hat{\theta}^{PP} &= L_n(\theta) + \tilde{L}_N^f(\theta) - L_n^f(\theta) \\
&= \frac{1}{N}\sum_{i=1}^N f(\tilde{X}_i) + \frac{1}{n}\sum_{i=1}^n Y_i - \frac{1}{n}\sum_{i=1}^n f(X_i) \\
&= \frac{1}{n}\sum_{i=1}^n Y_i + \left(\frac{1}{N}\sum_{i=1}^N f(\tilde{X}_i) - \frac{1}{n}\sum_{i=1}^n f(X_i)\right)
\end{aligned}$$

Where $L(\theta)$ is the losses. It has a limitation: when the provided predictions are inaccurate, the intervals can be worse than the "classical intervals" using only the labeled data. To address this problem, we introduce a parameter λ:

$$\begin{aligned}
\hat{\theta}^{PP++} &= L_n(\theta) + \lambda(\tilde{L}_N^f(\theta) - L_n^f(\theta)) \\
&= \frac{1}{n}\sum_{i=1}^n Y_i + \lambda\left(\frac{1}{N}\sum_{i=1}^N f(\tilde{X}_i) - \frac{1}{n}\sum_{i=1}^n f(X_i)\right)
\end{aligned}$$

Tuning the parameter λ to minimize the asymptotic variance of the prediction-powered estimator, we have:

$$\begin{aligned}
\hat{\lambda} &= argmin_\lambda E\left\{\frac{1}{n}\sum_{i=1}^n Y_i - \theta^* + \lambda\left[\frac{1}{N}\sum_{i=1}^N f(\tilde{X}_i) - \frac{1}{n}\sum_{i=1}^n f(X_i)\right]\right\}^2 \\
&= \frac{Cov\left(\frac{1}{n}\sum_{i=1}^n Y_i, \frac{1}{N}\sum_{i=1}^N f(\tilde{X}_i) - \frac{1}{n}\sum_{i=1}^n f(X_i)\right)}{Var\left(\frac{1}{N}\sum_{i=1}^N f(\tilde{X}_i) - \frac{1}{n}\sum_{i=1}^n f(X_i)\right)} \\
&= \frac{\frac{1}{n}Cov\left(Y, f(X)\right)}{\frac{1}{N}Var\left(f(\tilde{X})\right) + \frac{1}{n}Var\left(f(X)\right)}
\end{aligned}$$

In this case, PPI++ is essentially never worse than either classical or original PPI. Certainly the objective remains unbiased for any λ, since $E[\tilde{L}_N^f(\theta) - L_n^f(\theta)] = 0$, thus $E[L_n(\theta) + \tilde{L}_N^f(\theta) - L_n^f(\theta)] = L(\theta)$.

2.3 Cross-PPI

Cross-prediction gives more stable inferences especially when the prediction model is not accurate. It begins by spliting the labeled data into K chunks: $I_1 = \{1, ..., n/K\}, I_2 = \{n/K + 1, ..., 2n/K\}, ...$ The cross-prediction estimator is defined as:

$$\hat{\theta}^{Cross-PPI} = \frac{1}{n}\sum_{j=1}^{K}\sum_{i\in I_j} Y_i + \frac{1}{KN}\sum_{j=1}^{K}\sum_{i=1}^{N} f^{(j)}(\tilde{X}_i) - \frac{1}{n}\sum_{j=1}^{K}\sum_{i\in I_j} f^{(j)}(X_i)$$

where $f^{(j)}$ is the model obtained by training on all data except the I_j. Intuitively, the second term in $\hat{\theta}^{Cross-PPI}$ is an empirical approximation of the population mean, the difference between first and third term is the rectifier which is an estimate of the bias between the predicted labels and the true labels. This cross-prediction estimator is also unbiased, as $E[f^{(j)}(\tilde{X}_{i'})] = E[f^{(j)}(X_i)]$ for all $j \in [K], i \in I_j, i' \in [N]$.

2.4 Cross-PPI++

Combining PPI++ and Cross-PPI, we can compute the cross-PPI++ estimator, defined as:

$$\hat{\theta}^{Cross-PPI++} = \frac{1}{n}\sum_{j=1}^{K}\sum_{i\in I_j} Y_i + \lambda\left[\frac{1}{KN}\sum_{j=1}^{K}\sum_{i=1}^{N} f^{(j)}(\tilde{X}_i) - \frac{1}{n}\sum_{j=1}^{K}\sum_{i\in I_j} f^{(j)}(X_i)\right]$$

Similarly, the parameter λ is tuned to minimize the asymptotic variance of the prediction-powered estimator. The optimal λ is:

$$\begin{aligned}
&\hat{\lambda} \text{ with k-fold cross-validation} \\
&= \frac{Cov\left(\frac{1}{n}\sum_{i=1}^{n} w(X_i)Y_i, \frac{1}{KN}\sum_{j=1}^{K}\sum_{i=1}^{N} f^{(j)}(\tilde{X}_i) - \frac{1}{n}\sum_{j=1}^{K}\sum_{i\in I_j} w(X_i)f^{(j)}(X_i)\right)}{Var\left(\frac{1}{KN}\sum_{j=1}^{K}\sum_{i=1}^{N} f^{(j)}(\tilde{X}_i) - \frac{1}{n}\sum_{j=1}^{K}\sum_{i\in I_j} w(X_i)f^{(j)}(X_i)\right)} \\
&= \frac{\frac{1}{n}Cov\left(w(X)Y, w(X)f^{(j)}(X)\right)}{\frac{1}{K^2N}\sum_{j=1}^{K} Var\left(f^{(j)}(\tilde{X})\right) + \frac{1}{n^2}\sum_{j=1}^{K}\sum_{i\in I_j} Var\left(w(X)f^{(j)}(X)\right)}
\end{aligned}$$

3 Extensions under Covariate Shift

3.1 Standard PPI under Covariate shift

Although the foundational PPI framework assumes that the labeled and unlabeled data are drawn from the same distribution, this assumption does not always hold in practice. One common violation is covariate shift, where the distribution of input features, $p(X)$, differs between the labeled and unlabeled dataset, while the conditional distribution of the outcome given the feature $p(Y|X)$ remains the same (Shimodaira, 2000). To address this mismatch, the density ratio $w(x) = \tilde{p}(x)/p(x)$ is introduced, where $\tilde{p}(x)$ is the distribution of the features in the unlabeled data and $p(x)$ is the distribution of the features in the gold-standard data (Sugiyama et al., 2007). Incorporating the density ratio allows for adjusting model predictions to account for the covariate shift, ensuring valid and unbiased inference in scenarios where the covariate shift occurs.

Denote $w(x) = p^*(x)/\tilde{p}(x)$ the density ratio.

$$\hat{\theta} \text{ by standard PPI under covariate shift} = \frac{1}{n}\sum_{i=1}^{n} w(X_i)Y_i + \frac{1}{N}\sum_{i=1}^{N} f(\tilde{X}_i) - \frac{1}{n}\sum_{i=1}^{n} w(X_i)f(X_i)$$

3.2 Efficient PPI under Covariate shift

Efficient PPI estimator adjusted by density ratio $w(x) = \tilde{p}(x)/p(x)$ is:

$$\hat{\theta} \text{ by Efficient PPI under covariate shift} = \frac{1}{n}\sum_{i=1}^{n} w(X_i)Y_i + \lambda\left[\frac{1}{N}\sum_{i=1}^{N} f(\tilde{X}_i) - \frac{1}{n}\sum_{i=1}^{n} w(X_i)f(X_i)\right]$$

We can find the parameter λ at:

$$\begin{aligned}
\hat{\lambda} &= argmin_{\lambda} E\left\{\frac{1}{n}\sum_{i=1}^{n} w(X_i)Y_i - \theta^* + \lambda\left[\frac{1}{N}\sum_{i=1}^{N} f(\tilde{X}_i) - \frac{1}{n}\sum_{i=1}^{n} w(X_i)f(X_i)\right]\right\}^2 \\
&= \frac{Cov\left(\frac{1}{n}\sum_{i=1}^{n} w(X_i)Y_i, \frac{1}{N}\sum_{i=1}^{N} f(\tilde{X}_i) - \frac{1}{n}\sum_{i=1}^{n} w(X_i)f(X_i)\right)}{Var\left(\frac{1}{N}\sum_{i=1}^{N} f(\tilde{X}_i) - \frac{1}{n}\sum_{i=1}^{n} w(X_i)f(X_i)\right)} \\
&= \frac{\frac{1}{n}Cov\left(w(X)Y, w(X)f(X)\right)}{\frac{1}{N}Var\left(f(\tilde{X})\right) + \frac{1}{n}Var\left(w(X)f(X)\right)}
\end{aligned}$$

3.3 Cross Efficient PPI under Covariate Shift

We can compute the cross-prediction estimator using the trained models, consider the covariate shift situation and adding density ratio, the estimator by Cross-PPI is defined as:

$$\begin{aligned}
&\hat{\theta} \text{ by Cross PPI under covariate shift} \\
=&\frac{1}{n}\sum_{j=1}^{K}\sum_{i\in I_j} w(X_i)Y_i + \frac{1}{KN}\sum_{j=1}^{K}\sum_{i=1}^{N} f^{(j)}(\tilde{X}_i) - \frac{1}{n}\sum_{j=1}^{K}\sum_{i\in I_j} w(X_i)f^{(j)}(X_i)
\end{aligned}$$

Adding the parameter λ to minimize the asymptotic variance of the prediction-powered estimator under this covariate shift situation, we have:

$$\begin{aligned}
&\hat{\theta} \text{ by Cross Efficient PPI under covariate shift} \\
=&\frac{1}{n}\sum_{j=1}^{K}\sum_{i\in I_j} w(X_i)Y_i + \lambda\left[\frac{1}{KN}\sum_{j=1}^{K}\sum_{i=1}^{N} f^{(j)}(\tilde{X}_i) - \frac{1}{n}\sum_{j=1}^{K}\sum_{i\in I_j} w(X_i)f^{(j)}(X_i)\right]
\end{aligned}$$

Thus, λ can be found at:

$$\begin{aligned}
&\hat{\lambda} \text{ with k-fold cross-validation} \\
=&\frac{Cov\left(\frac{1}{n}\sum_{i=1}^{n} w(X_i)Y_i, \frac{1}{KN}\sum_{j=1}^{K}\sum_{i=1}^{N} f^{(j)}(\tilde{X}_i) - \frac{1}{n}\sum_{j=1}^{K}\sum_{i\in I_j} w(X_i)f^{(j)}(X_i)\right)}{Var\left(\frac{1}{KN}\sum_{j=1}^{K}\sum_{i=1}^{N} f^{(j)}(\tilde{X}_i) - \frac{1}{n}\sum_{j=1}^{K}\sum_{i\in I_j} w(X_i)f^{(j)}(X_i)\right)} \\
=&\frac{\frac{1}{n}Cov\left(w(X)Y, w(X)f^{(j)}(X)\right)}{\frac{1}{K^2N}\sum_{j=1}^{K} Var\left(f^{(j)}(\tilde{X})\right) + \frac{1}{n}Var\left(w(X)f^{(\cdot)}(X)\right)}
\end{aligned}$$

4 Simulation Study

4.1 Simulation Assuming IID Data

When (X, Y) and $(\tilde{X}, \tilde{Y})$ are are independently and identically distributed (i.i.d.), the performance of PPI methods depends on two key factors:

1. The relative size of N (unlabeled data) compared to n (labeled data). The accuracy of the prediction model.
2. The accuracy of the prediction model.

To evaluate these factors, we designed a simulation study with the following settings:

- **Unlabeled data size N:** fixed at 10000.
- **Labeled data size n:** varied across 100, 200, 500, 1000, 2000, 3000, 4000, 5000.
- **Prediction model:** A linear regression model defined as:

$$Y = 1 + 2X_1 + 3X_2 + \epsilon$$

  where $X_1 \sim N(0, 1)$, $X_2 \sim Bernoulli(0.5)$, and $\epsilon \sim N(0, \sigma_e^2)$. Here, X_1 and X_2 are independent.
- **Noise level σ_e:** set to 0.1, 1, 2, 5.
- **Predicted values $\hat{Y}$:**
  - For PPI and PPI++, $\hat{Y}$ is computed from linear models trained on the labeled data.
  - For Cross-PPI and Cross-PPI++, $\hat{Y}$ obtained from cross-predictions using $K = 5$ folds.
- **Simulation replications:** each scenario is replicated 100 times, and all results are averaged over these replications.
- **Performance metrics:** we evaluate the performance of PPI methods using the following metrics:
  1. **Mean width of confidence intervals:** A measure of efficiency, where narrower intervals indicate better precision.
  2. **Coverage probability:** proportion of confidence intervals that include the true parameter value, where higher values indicate that the method produces valid intervals that reliably capture the true value.

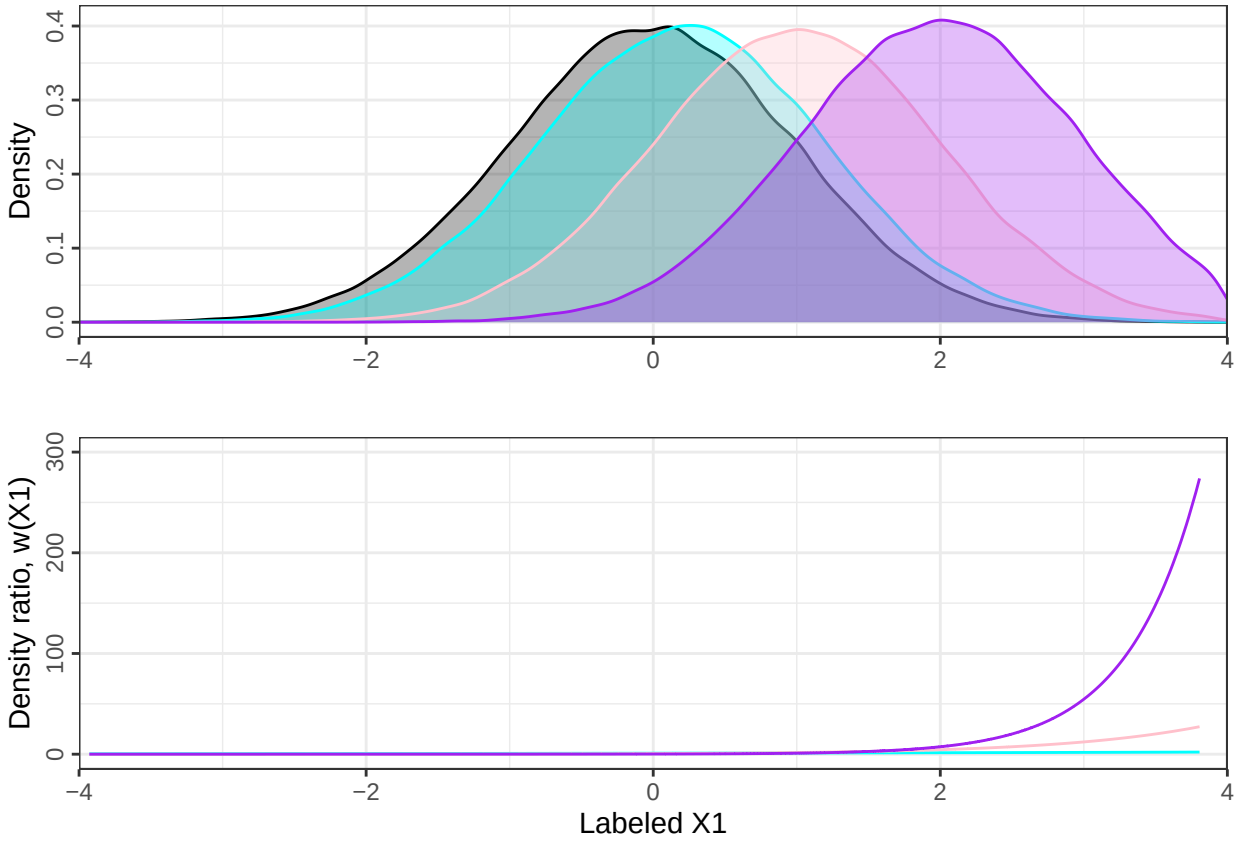
4.2 Simulation Under Covariate Shift

When the distribution of (X, Y) and $(\tilde{X}, \tilde{Y})$ is different, the performance of PPI methods can be affected by the covariate shift. We tested the performance of PPI methods incorporating the density ratio $w(X)$ under three different scenarios:

1. In the unlabeled data, $X_1 \sim N(\mu_1, \sigma_0^2)$, where $\mu_1 = 0.2, 1, 2$, and $X_2 \sim Bernoulli(p_0)$. The distribution of X_1 undergoes a location shift to the right as μ_1 increase, while its overall shape remains unchanged. In this case, the density ratio is:

$$w(x) = \frac{\tilde{p}(x)}{p(x)} = \exp\left\{\frac{1}{2\sigma_0^2}(2x_1 - \mu_1 - \mu_0)(\mu_1 - \mu_0)\right\},$$

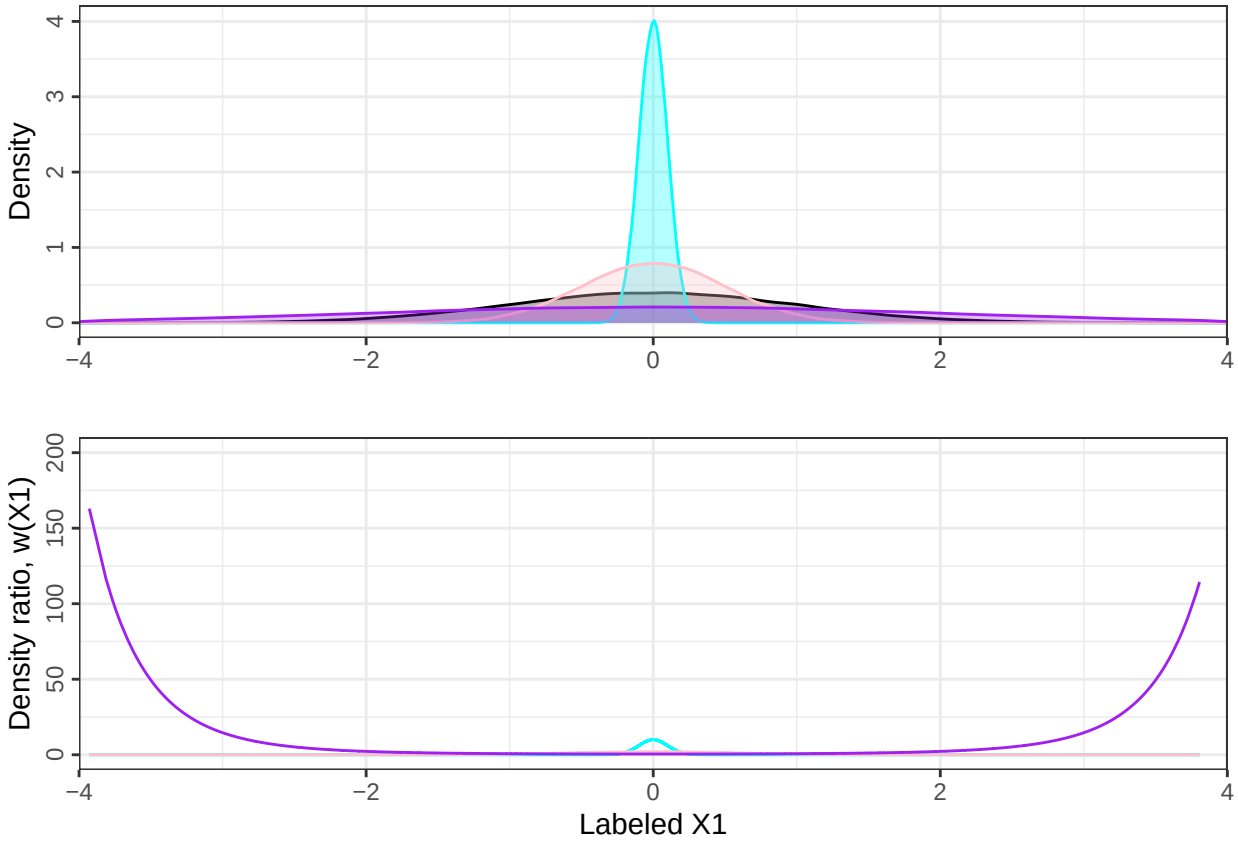
which exhibits an exponential growth pattern as X_1 increase, particularly for larger values of X_1. This behavior indicates that the discrepancy between the distributions of labeled and unlabeled data becomes increasingly pronounced in the tails.



2. $X_1 \sim N(\mu_0, \sigma_1^2)$, where $\sigma_1 = 0.1, 0.5, 2$, and $X_2 \sim Bernoulli(p_0)$. In this case, the density ratio is:

$$w(x) = \frac{\tilde{p}(x)}{p(x)} = \frac{\sigma_0}{\sigma_1}\exp\left\{-\frac{1}{2}\left(\frac{1}{\sigma_1^2} - \frac{1}{\sigma_0^2}\right)(x_1 - \mu_0)^2\right\}$$

When $\sigma_1 < \sigma_0$, the density ratio shows sharp peaks near the center and approaches zero in the tails due to the tighter concentration of the unlabeled data. Conversely, when $\sigma_1 > \sigma_0$, the density ratio flattens and increases exponentially at the tails, as the unlabeled data becomes more dispersed relative to the labeled data.



3. $X_1 \sim N(\mu_0, \sigma_0^2)$, and $X_2 \sim Bernoulli(p_1)$, where $p_1 = 0.55, 0.7, 0.85$. In this case, the density ratio is:

$$w(x) = \frac{\tilde{p}(x)}{p(x)} = \left(\frac{p_1}{p_0}\right)^{x_2} \left(\frac{1-p_1}{1-p_0}\right)^{1-x_2}$$

The density ratios remain within a limited range compared to the potentially unbounded growth of density ratios seen in normal distributions under covariate shift.

p1	w(X2=0)	w(X2=1)
0.55	0.9	1.1
0.70	0.6	1.4
0.85	0.3	1.7

5 Results

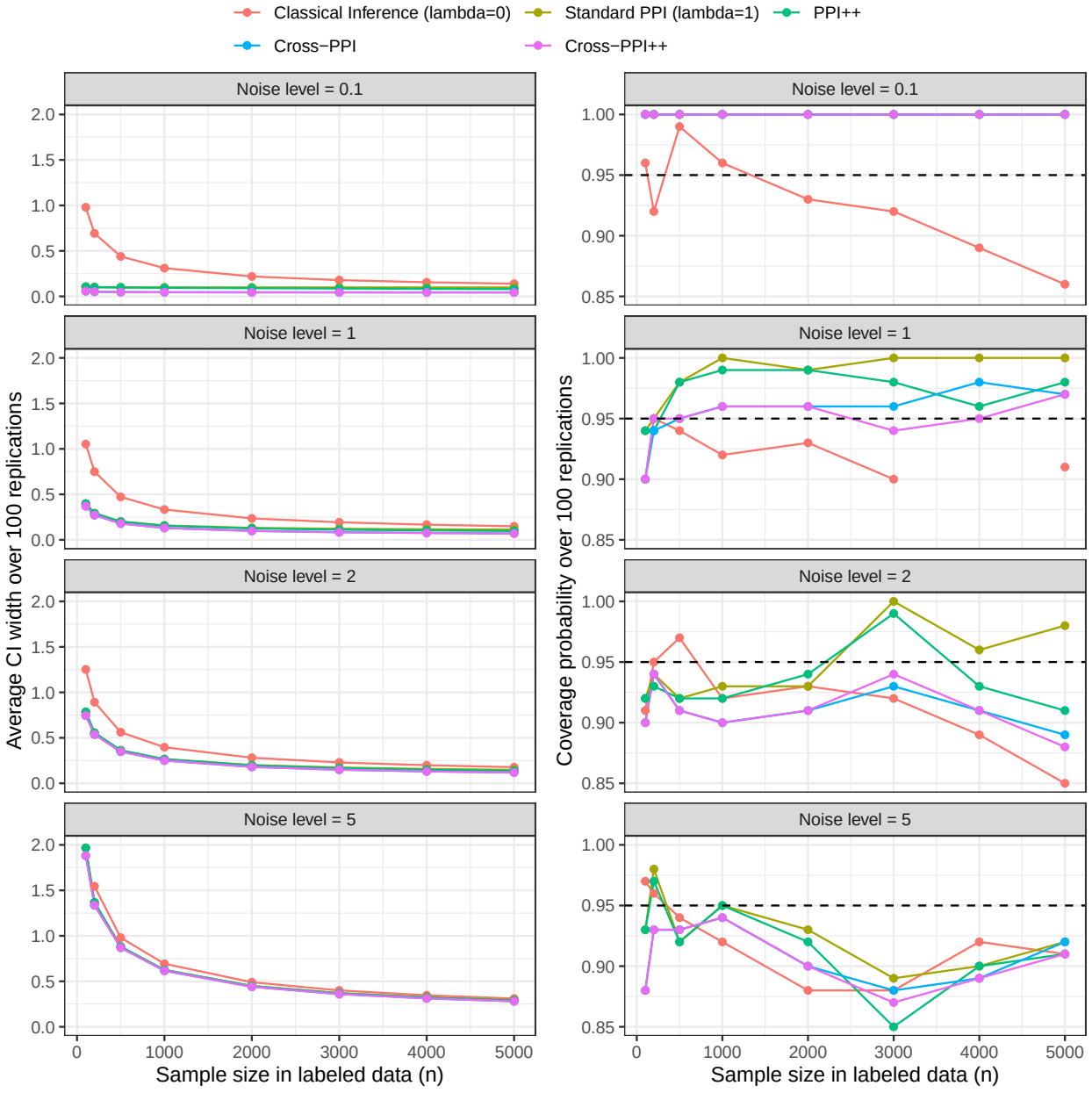
5.1 Performance of PPI Methods Without Covariate Shift

We compared five PPI methods under the assumption i.i.d. data. The methods include the following:

1. Classical inference, where $\lambda = 1$.
2. Standard PPI, where $\lambda = 1$.
3. Efficient PPI, where λ is estimated by the efficient score.

4. Cross-PPI, where $\lambda = 1$.
5. Cross-PPI++, where λ is estimated by the efficient score.

The performance of these methods was evaluated based on the mean width of the confidence interval and the coverage probability. In the plot below, we show the mean width of the confidence interval on the left side and the coverage probability on the right side by sample size in the labeled data. Difference methods are represented by different colors. The noise levels are depicted in different rows in the plot matrix.



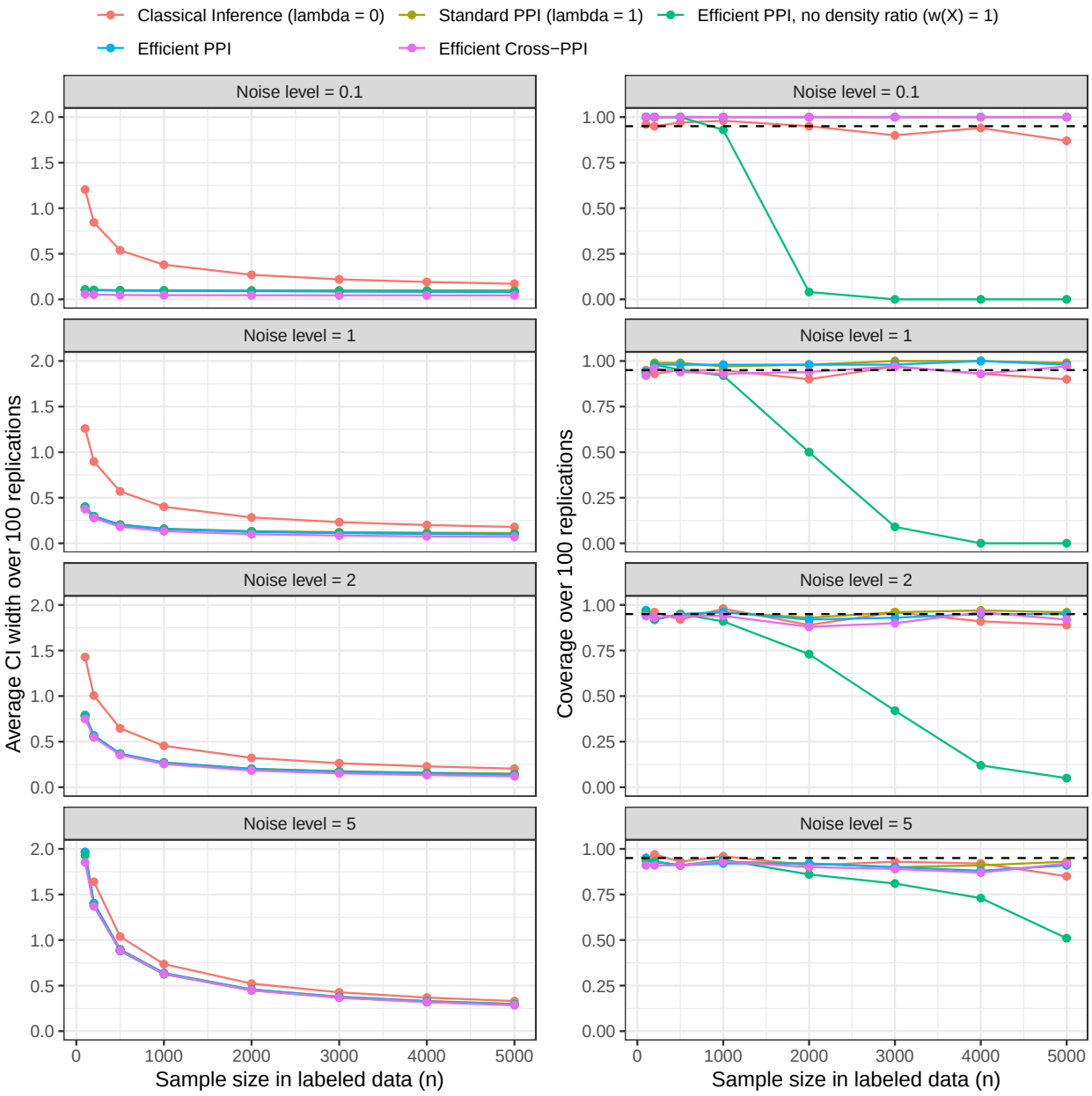
5.2 Performance of PPI Methods Under Covariate Shift

We compared five PPI methods under the assumption i.i.d. data. The methods include the following:

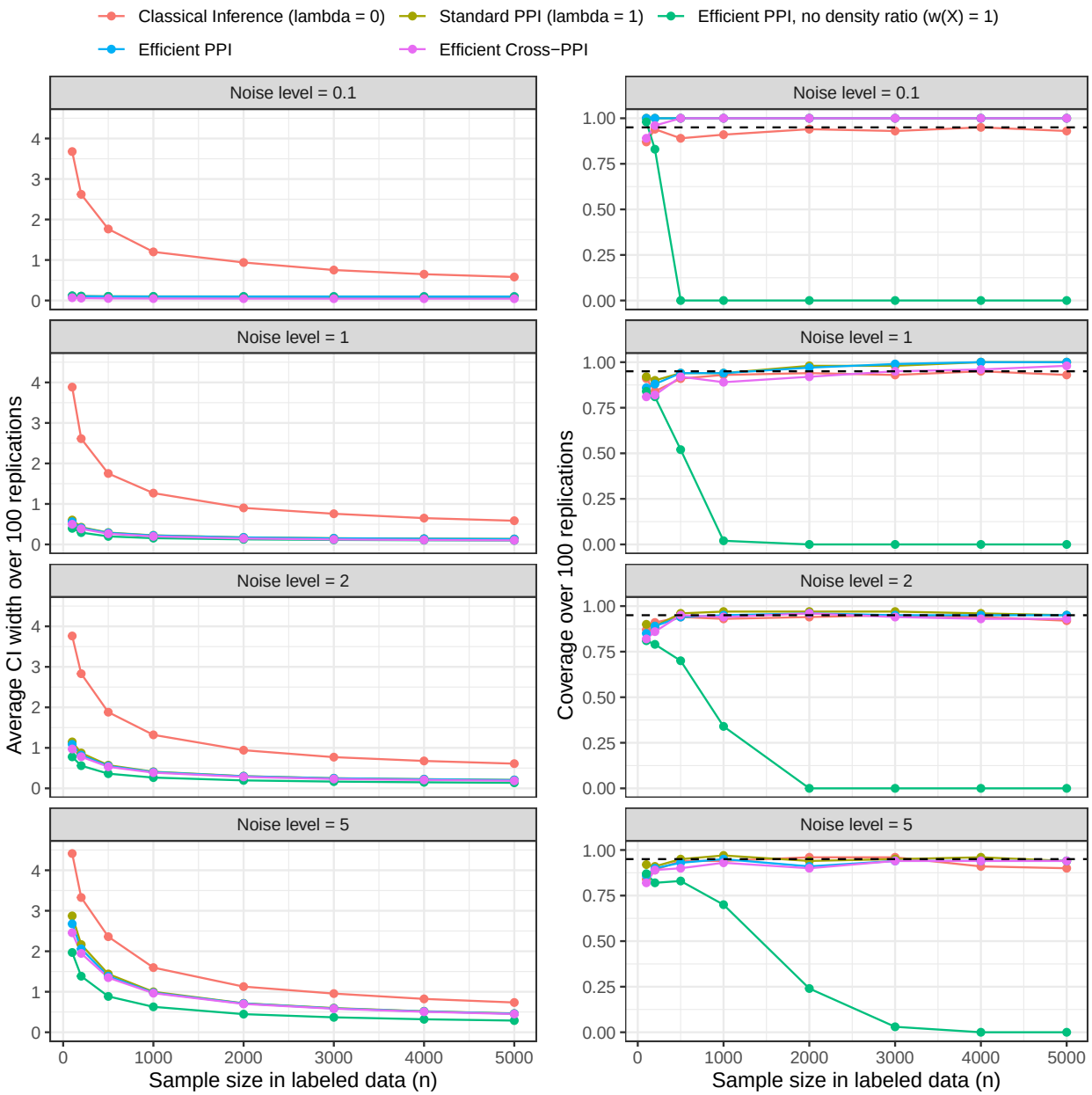
1. Classical inference, where $\lambda = 1$.

2. Standard PPI, where $\lambda = 1$.
3. Efficient PPI, without adjusting for the covariate shift.
4. Efficient PPI, where λ is estimated by the efficient score.
5. Cross-PPI++, where λ is estimated by the efficient score.

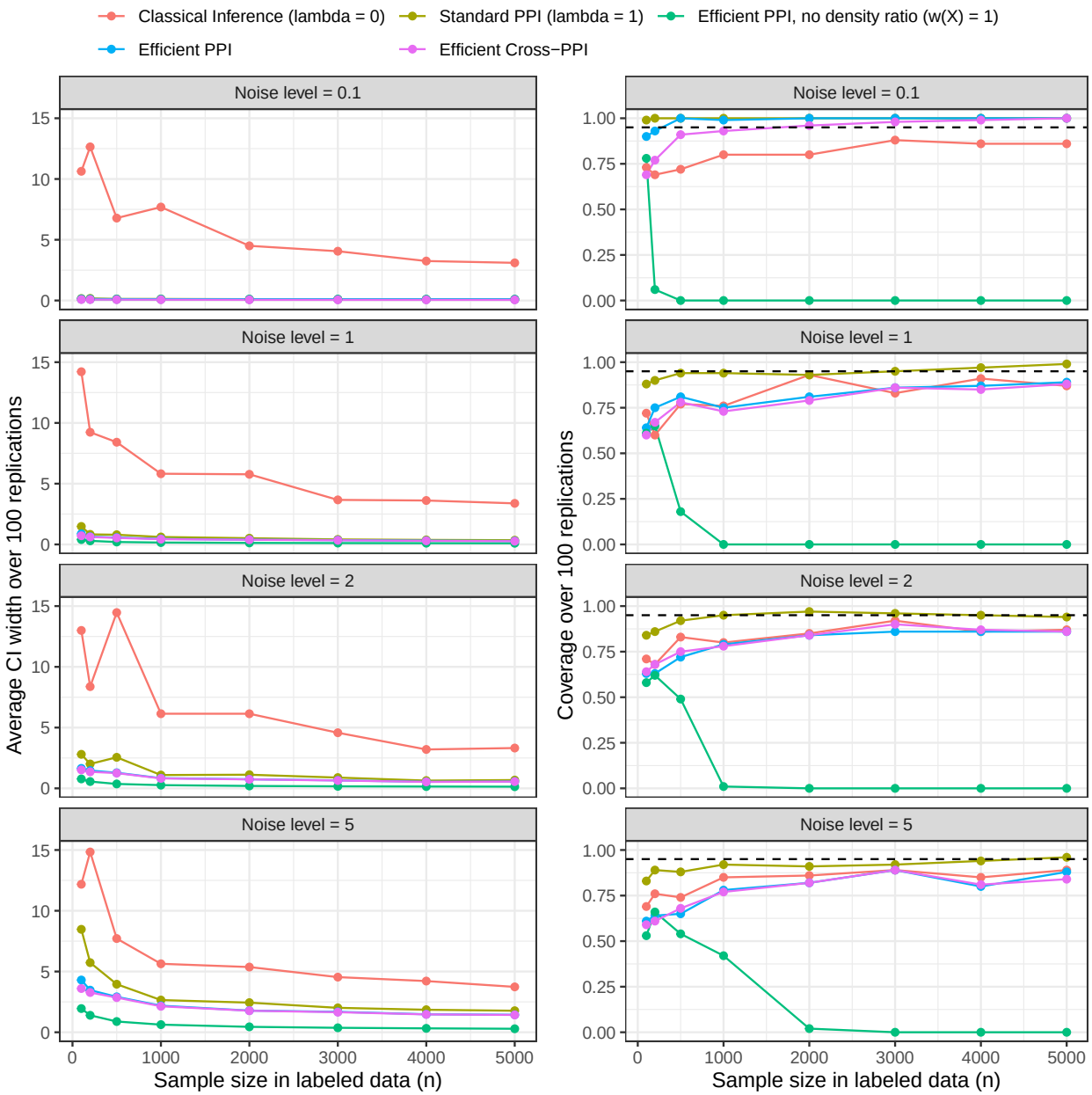
5.2.1 Scenario 1, $\mu_0 = 0, \mu_1 = 0.2$



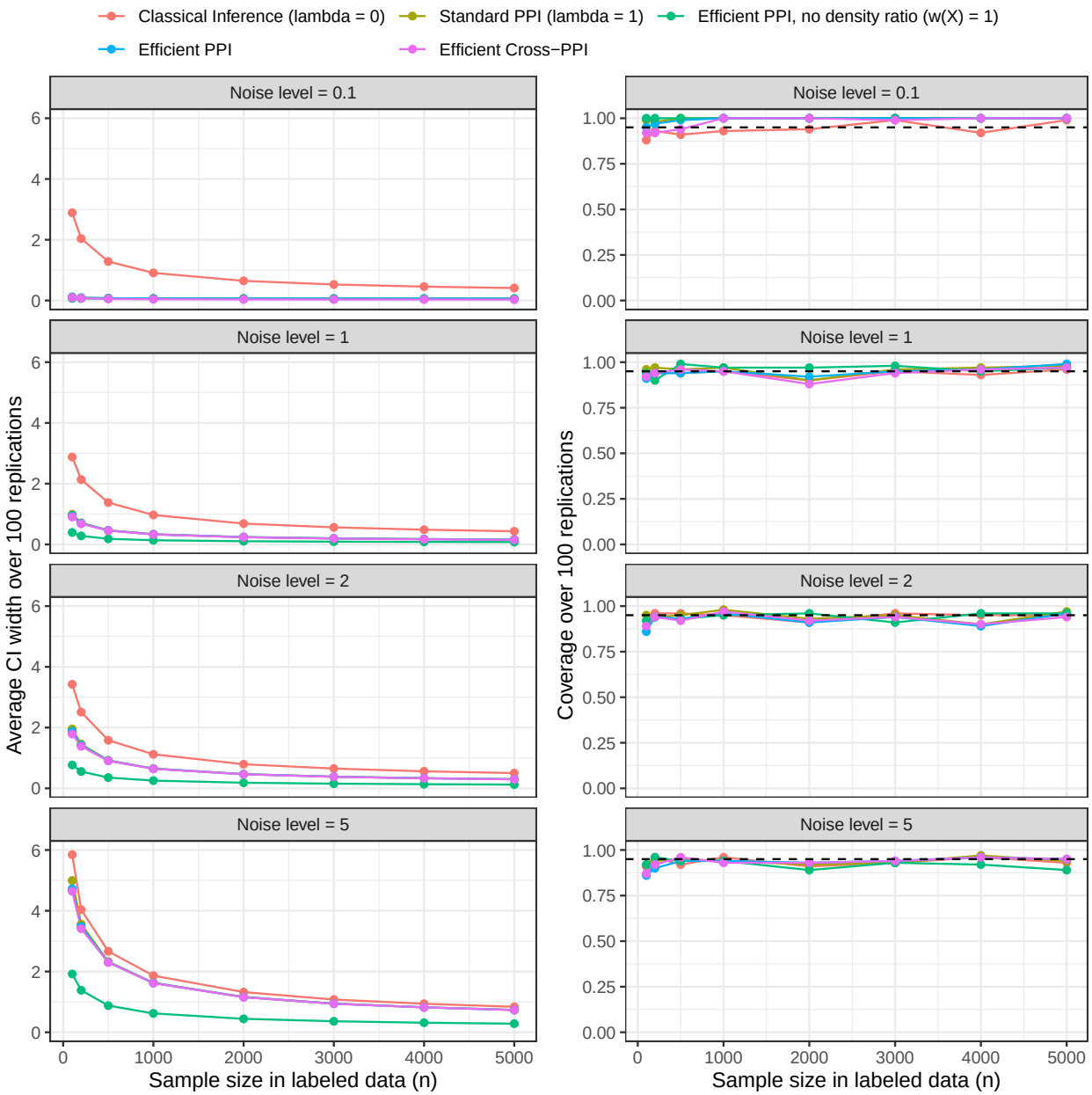
5.2.2 Scenario 1, $\mu_0 = 0, \mu_1 = 1$



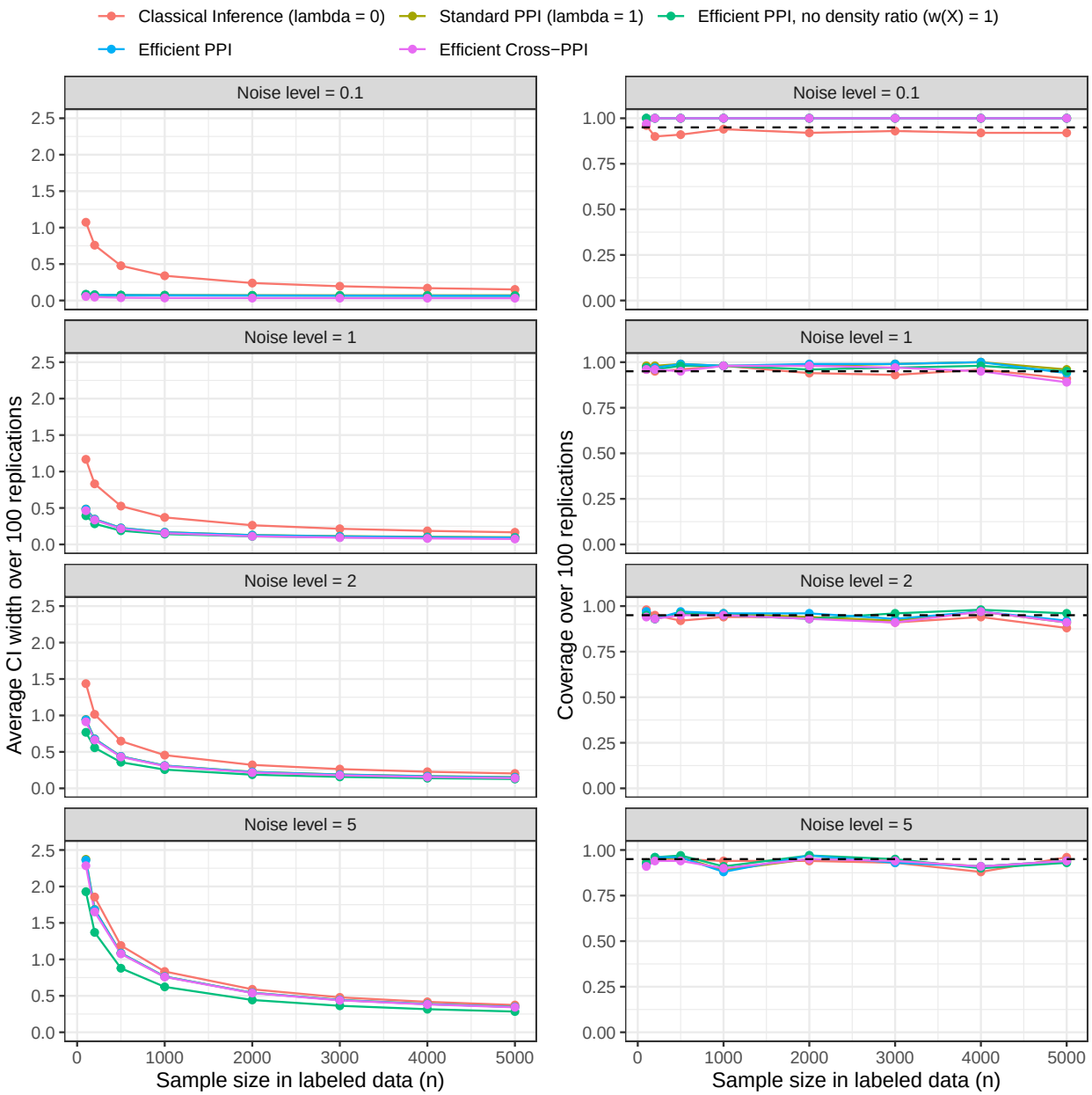
5.2.3 Scenario 1, $\mu_0 = 0, \mu_1 = 2$



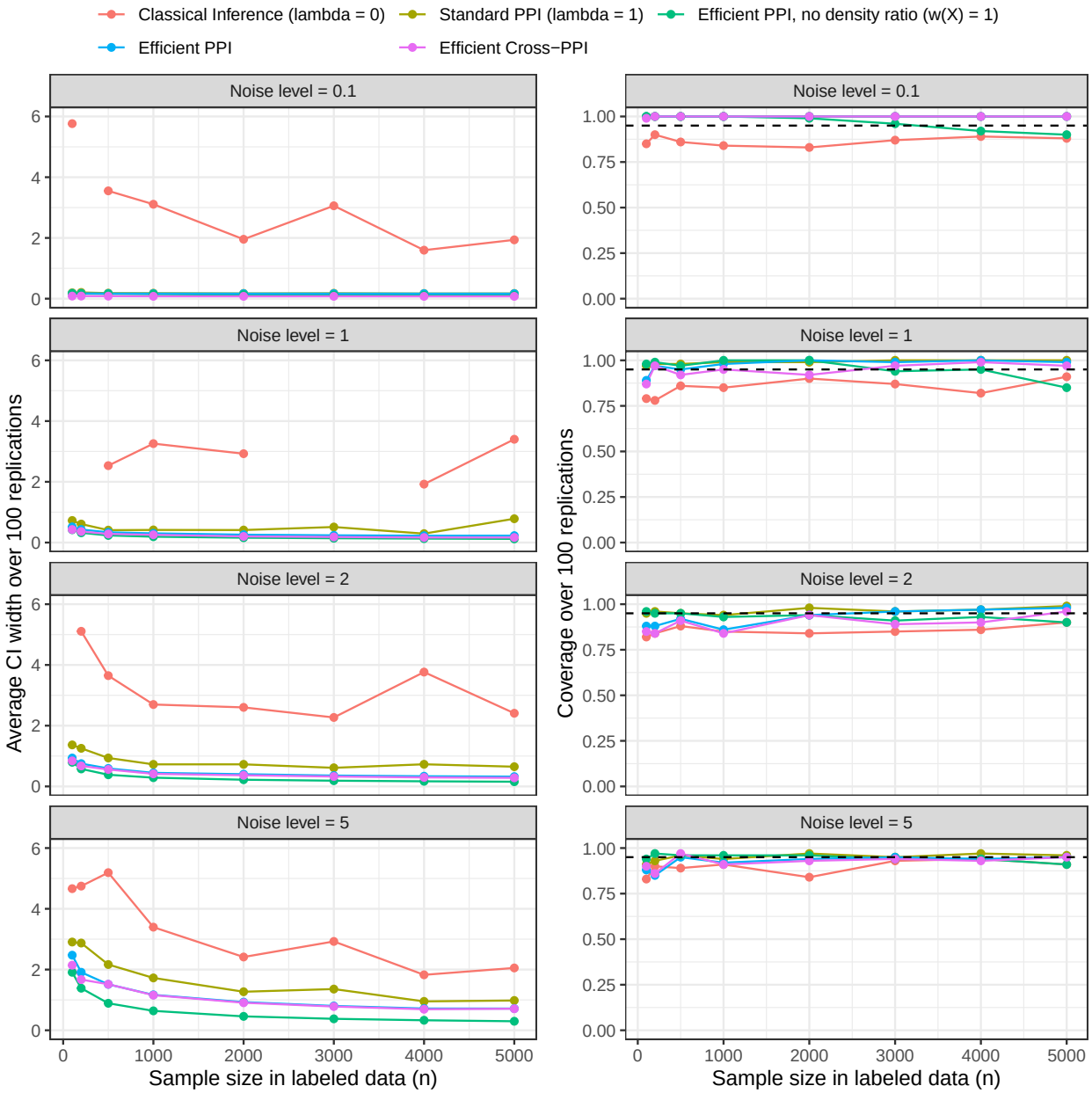
5.2.4 Scenario 2, $\sigma_0 = 1, \sigma_1 = 0.1$



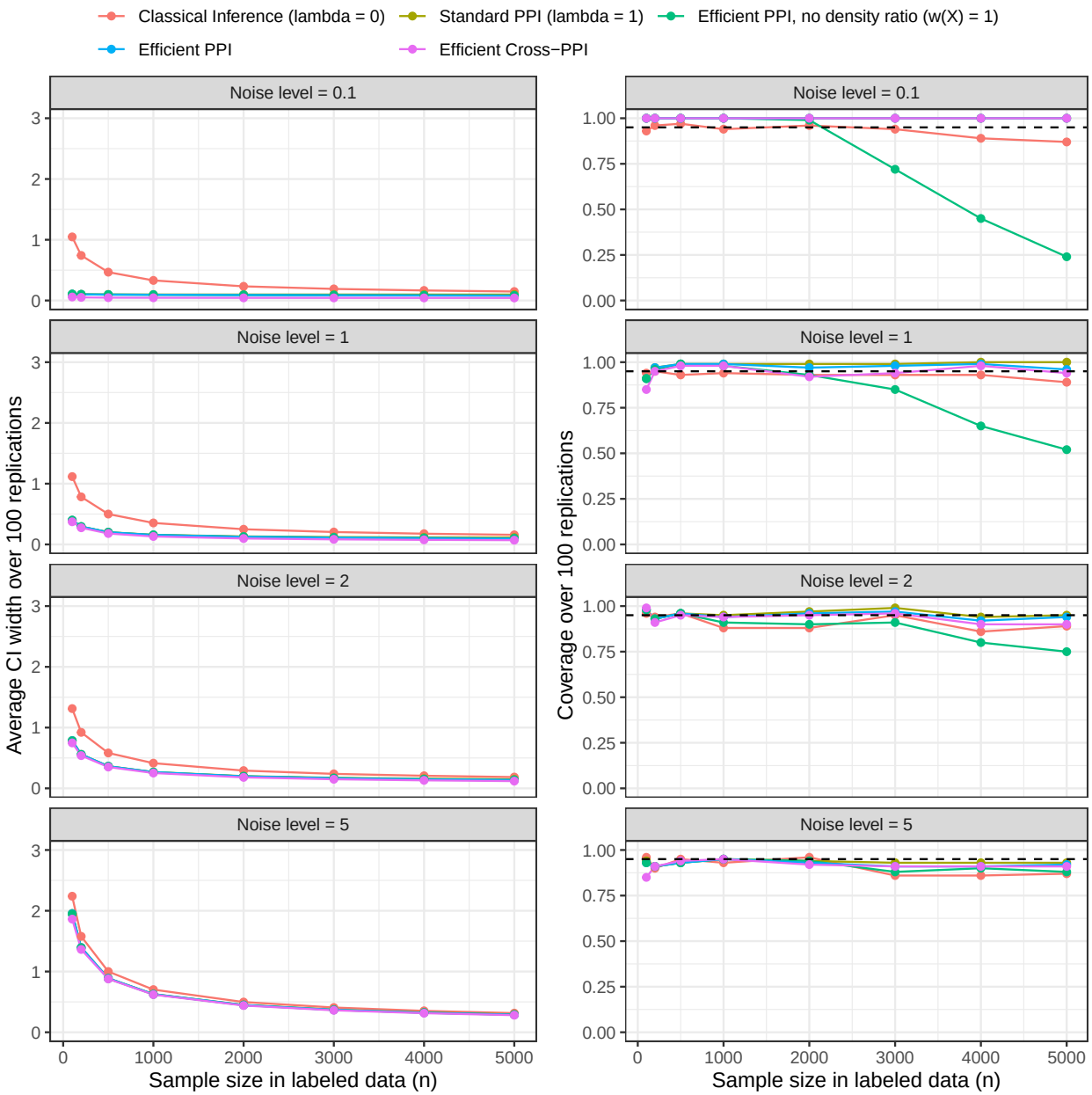
5.2.5 Scenario 2, $\sigma_0 = 1, \sigma_1 = 0.5$



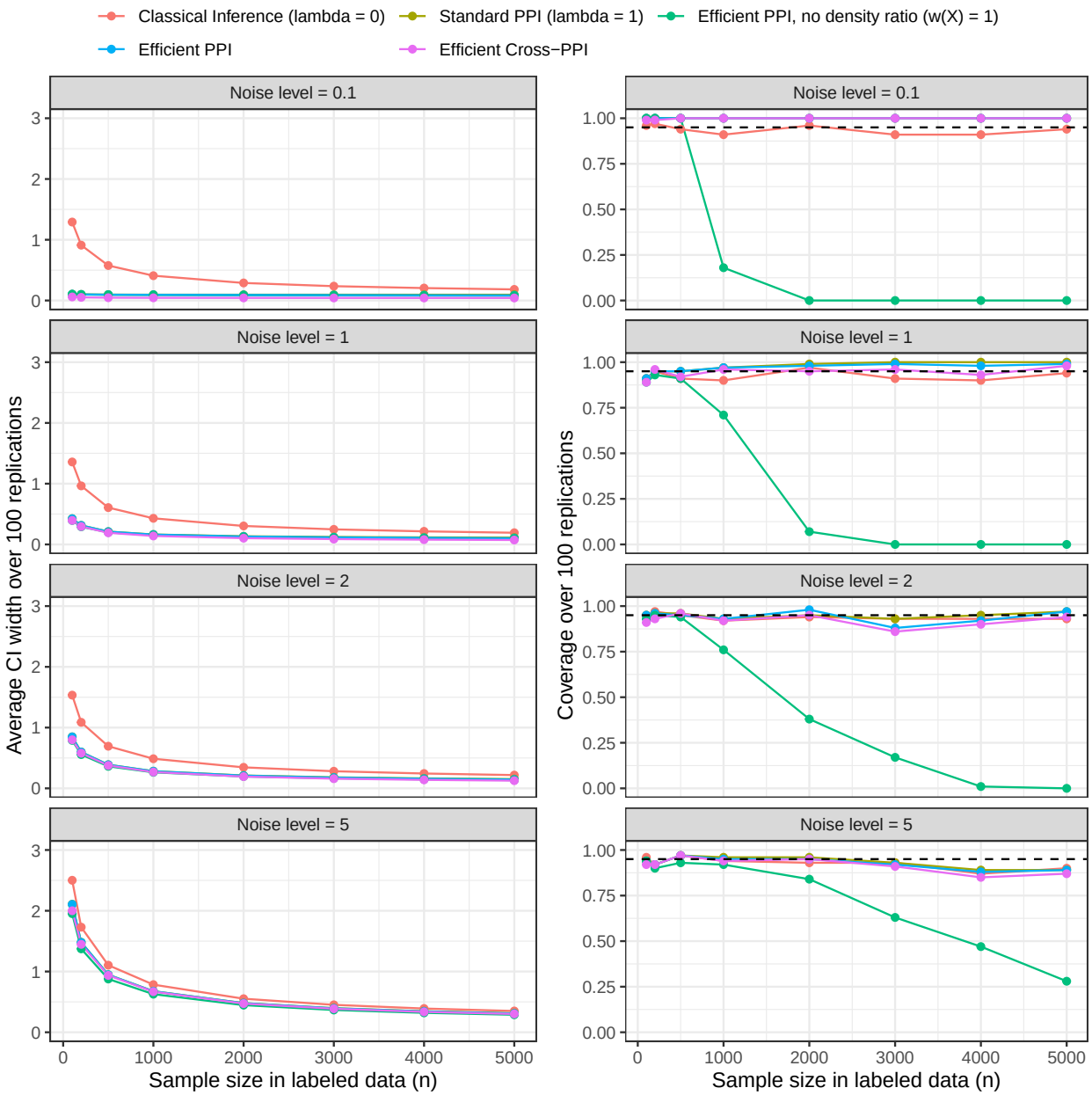
5.2.6 Scenario 2, $\sigma_0 = 1, \sigma_1 = 2$



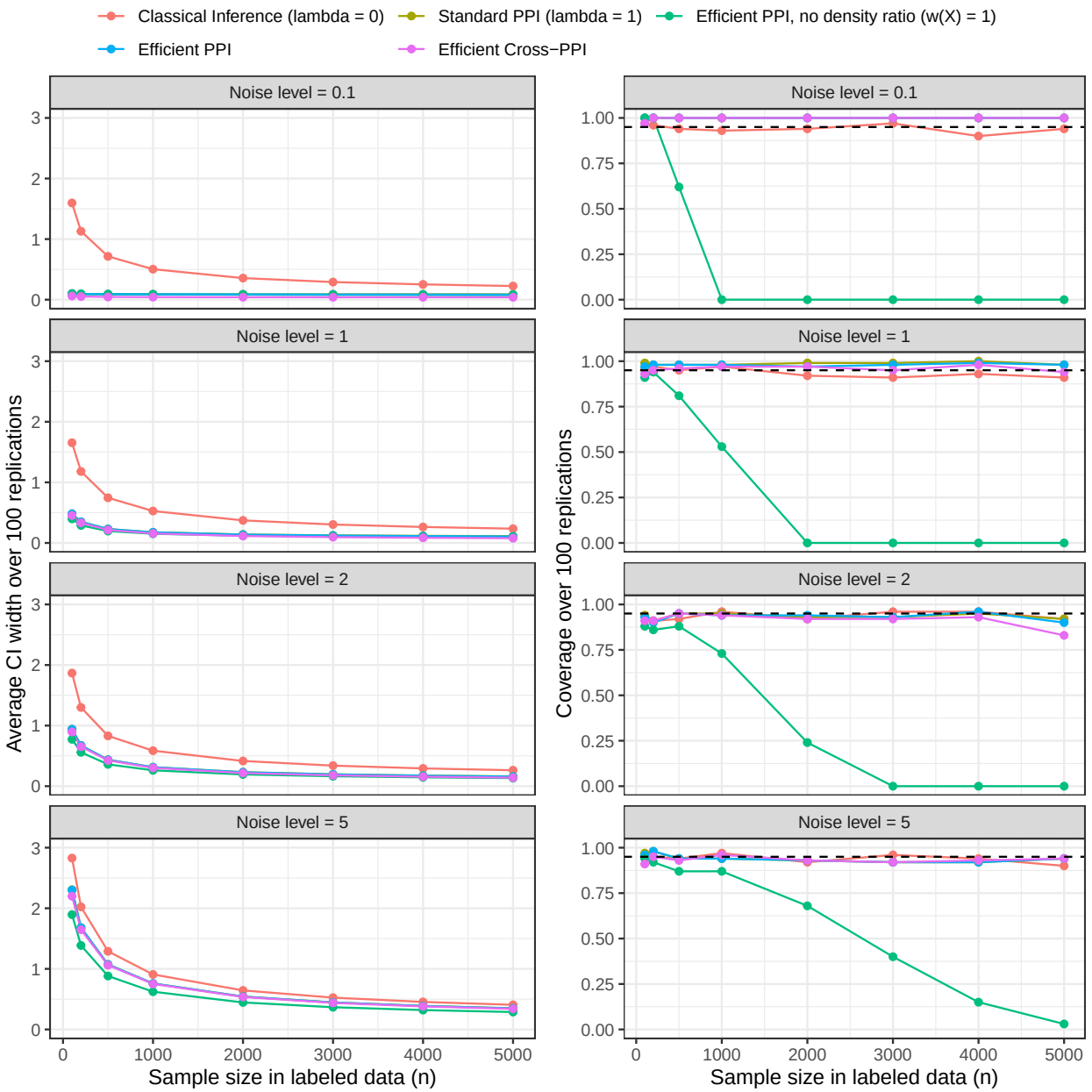
5.2.7 Scenario 3, $p_0 = 0.5, p_1 = 0.55$



5.2.8 Scenario 3, $p_0 = 0.5, p_1 = 0.7$



5.2.9 Scenario 3, $p_0 = 0.5, p_1 = 0.85$



6 Conclusion

We compared the performance of different PPI frameworks under different scenarios of covariate shifts. Efficient PPI and Efficient cross-PPI have relatively narrower CI widths compared to standard PPI and CLT. When the covariate shift is mild and moderate, most of the PPI frameworks can maintain valid, with coverage over or close to 95%. However, when the difference between two distributions are large, only standard PPI has coverage validity, especially when labeled sample size is small. Another situation that PPI will be under powered is when the prediction model is not accurate, which is simulated as higher noise level in our analysis. The coverage of all PPI frameworks decrease when noise level increases, but standard PPI is again the best across all frameworks. Thus, Efficient PPI frameworks provide more precise parameter estimates

while maintaining robustness under mild to moderate covariate shifts and noise level.

7 Discussion

Density ratio weighting performs well when the labeled and unlabeled data have relatively similar distributions. In such cases, PPI frameworks with density ratio weighting can achieve narrow and valid confidence interval estimation. However, estimating density ratio in real-world data could be challenging, where the data-generating process is complex and the true underlying distributions are unknown. Future work could explore the use of more robust techniques for density ratio estimation, such as kernel-based methods or machine learning approaches, to improve reliability in diverse real-world scenarios.

Our study also highlights the trade-off between efficiency and validity across different PPI frameworks. Both adding parameter λ and involving cross-prediction can improve efficiency, however, they may loss validity when covariate shift is severe and noise level in data is high. Standard PPI always maintain the highest overall coverage rate in different scenarios, and this robustness is particularly important in real-world applications. Therefore, the choice of PPI framework depends on the nature of application and data, balancing the efficiency and validity for specific scientific questions.

8 References

Angelopoulos, Anastasios N., Stephen Bates, Clara Fannjiang, Michael I. Jordan, and Tijana Zrnic. 2023. "Prediction-Powered Inference." *Science* 382 (6671): 669–74. https://doi.org/10.1126/science.adi6000.

Angelopoulos, Anastasios N., John C. Duchi, and Tijana Zrnic. 2024. "PPI++: Efficient Prediction-Powered Inference." arXiv. https://doi.org/10.48550/arXiv.2311.01453.

Sterne, Jonathan A. C., Ian R. White, John B. Carlin, Michael Spratt, Patrick Royston, Michael G. Kenward, Angela M. Wood, and James R. Carpenter. 2009. "Multiple Imputation for Missing Data in Epidemiological and Clinical Research: Potential and Pitfalls." *BMJ* 338 (June): b2393. https://doi.org/10.1136/bmj.b2393.

Zrnic, Tijana, and Emmanuel J. Candès. 2024. "Cross-Prediction-Powered Inference." *Proceedings of the National Academy of Sciences* 121 (15): e2322083121. https://doi.org/10.1073/pnas.2322083121.

9 Appendix

```r
# simulation settings

# basic settings in iid simulation
# unlabeled data size
N <- 10000
# labeled data size
n <- c(100, 200, 500, 1000, 2000, 3000, 4000, 5000)
# distribution parameters for X1 and X2 in labeled data
mu_0 <- 0
sigma_0 <- 1
p_0 <- 0.5
# noise level
sigma_e <- c(0.1, 1, 2, 5)
# number of folds for cross-fit
K <- 5
# number of replications
```

```r
B <- 100
# combine iid settings
iid_settings <- expand.grid(n = n, N = N, sigma_e = sigma_e)

# additional settings under covariate shift
# distribution parameters for X1 and X2 in unlabeled data
mu_1 <- c(0.2, 1, 2)
sigma_1 <- c(0.1, 0.5, 2)
p_1 <- c(0.55, 0.7, 0.85)

# parameter combinations
param_comb <- rbind(
  # scenario 1: covariate shift in X1 with mu changed
  data.frame(mu = mu_1, sigma = sigma_0, p = p_0, scenario = "Scenario 1"),
  # scenario 2: covariate shift in X1 with sigma changed
  data.frame(mu = mu_0, sigma = sigma_1, p = p_0, scenario = "Scenario 2"),
  # scenario 3: covariate shift in X2 with p changed
  data.frame(mu = mu_0, sigma = sigma_0, p = p_1, scenario = "Scenario 3")
)
covshift_settings <- merge(expand.grid(n = n, N = N, sigma_e = sigma_e), param_comb, all = T)
```

```r
# functions used in simulation

# define function to generate a single dataset
gen_data <- function(
    size,       # sample size
    mu, sigma,  # for X1
    p,          # for X2
    sigma_e # for epsilon
){
  X1 <- rnorm(size, mu, sigma)
  X2 <- rbinom(size, 1, p)
  Y <- 1 + 2*X1 + 3*X2 + rnorm(size, 0, sigma_e)
  return(data.frame(
    X1 = X1,
    X2 = X2,
    Y = Y
  ))
}

# density functions
w1 <- function(x1, mu_1, mu_0, sigma_0, truncation = NULL){
  w <- exp((1/(2*sigma_0^2))*(2*x1-mu_1-mu_0)*(mu_1-mu_0))
  if(!is.null(truncation)) w[w > truncation] <- truncation
  return(w)
}
w2 <- function(x1, sigma_1, mu_0, sigma_0, truncation = NULL){
  w <- (sigma_0/sigma_1)*exp(-0.5*((1/sigma_1^2)-(1/sigma_0^2))*(x1-mu_0)^2)
  if(!is.null(truncation)) w[w > truncation] <- truncation
  return(w)
}
w3 <- function(x2, p_1, p_0, truncation = NULL){
  w <- (p_1/p_0)^x2*((1-p_1)/(1-p_0))^(1-x2)
```

```r
  if(!is.null(truncation)) w[w > truncation] <- truncation
  return(w)
}


# define function to combine labeled and unlabeled data
comb_data <- function(
    n, N,                # sample sizes
    mu_0, sigma_0, p_0, # X distribution in labeled data
    mu_1, sigma_1, p_1, # X distribution in unlabeled data
    sigma_e,             # noise level
    truncation = NULL    # truncation for weights
){
  # labeled data
  labeled_data <- gen_data(n, mu_0, sigma_0, p_0, sigma_e)
  # unlabeled data
  unlabeled_data <- gen_data(N, mu_1, sigma_1, p_1, sigma_e)

  # fit model
  fit <- lm(Y ~ X1 + X2, data = labeled_data)
  labeled_data$Y_hat <- predict(fit, labeled_data[,1:2])
  unlabeled_data$Y_hat <- predict(fit, unlabeled_data[,1:2])

  labeled_data$w <- 1
  # add weight
  if(mu_0 != mu_1){
    labeled_data$w <- w1(
      labeled_data$X1,
      mu_1 = mu_1,
      mu_0 = mu_0,
      sigma_0 = sigma_0,
      truncation = truncation
      )
  }
  if(sigma_0 != sigma_1){
    labeled_data$w <- w2(
      labeled_data$X1,
      sigma_1 = sigma_1,
      mu_0 = mu_0,
      sigma_0 = sigma_0,
      truncation = truncation
      )
  }
  if(p_0 != p_1){
    labeled_data$w <- w3(
      labeled_data$X2,
      p_1 = p_1,
      p_0 = p_0,
      truncation = truncation
      )
  }

  # generate output
```

```r
  return(list(
    settings = c(
      n = n, N = N,
      mu_0 = mu_0, sigma_0 = sigma_0, p_0 = p_0,
      mu_1 = mu_1, sigma_1 = sigma_1, p_1 = p_1,
      sigma_e = sigma_e
    ),
    labeled = labeled_data,
    unlabeled = unlabeled_data
  ))
}

# ---- ppi++ ----
lambda_ppi <- function(n, N, Y_n, Y_hat_n, Y_hat_N, w){
  cov <- cov(w*Y_n, w*Y_hat_n)
  var_N <- var(Y_hat_N)
  var_n <- var(w*Y_hat_n)
  lam <- cov/n/(var_N/N+var_n/n)
  return(lam)
}
pointest_ppi <- function(lam, n, N, Y_n, Y_hat_n, Y_hat_N, w){
  mean(w*Y_n) + lam*mean(unlist(Y_hat_N)) - lam*mean(Y_hat_n*w)
}
std_ppi <- function(lam, n, N, Y_n, Y_hat_n, Y_hat_N, w){
  Y_hat_N_var <- var(lam*Y_hat_N)/N
  rectifier_var <- var(w*(Y_n-lam*Y_hat_n))/n
  std <- sqrt(Y_hat_N_var+rectifier_var)
  return(std)
}
ci_ppi <- function(pointest, std, alpha) {
  z <- qnorm(1-alpha/2)
  ll <- pointest - z*std
  ul <- pointest + z*std
  return(c(ll, ul))
}
gen_res <- function(n, N, Y_n, Y_hat_n, Y_N, Y_hat_N, w, lam = NULL, alpha = 0.5) {
  if(is.null(lam)) {
    lam <- lambda_ppi(n, N, Y_n, Y_hat_n, Y_hat_N, w)
  }
  pointest <- pointest_ppi(lam, n, N, Y_n, Y_hat_n, Y_hat_N, w)
  std <- std_ppi(lam, n, N, Y_n, Y_hat_n, Y_hat_N, w)
  ci <- ci_ppi(pointest, std, alpha)
  return(c(
    lam = lam,
    truemean = mean(Y_N),
    pointest = pointest,
    std = std,
    ll = ci[1],
    ul = ci[2],
    ci_width = ci[2] - ci[1],
    coverage = (ci[1] <= mean(Y_N) & mean(Y_N) <= ci[2])
  ))
}
```

```r
# ---- cross ppi++ ----
lambda_ppi_cross <- function(n, N, Y_n, Y_hat_n, Y_hat_N, w, K, fold){
  cov <-
    sum(sapply(1:K, \(j) cov(w[fold == j]*Y_n[fold == j], w[fold == j]*Y_hat_n[fold == j])) *
          tapply(fold, fold, length)) / n
  var_N <- sum(apply(Y_hat_N, 2, var)) / (K^2)
  var_n <- sum(sapply(1:K, \(j) var(w[fold == j]*Y_hat_n[fold == j])) *
                 tapply(fold, fold, length)) / n
  lam <- cov/n/(var_N/N+var_n/n)
  return(lam)
}
std_ppi_cross <- function(lam, n, N, Y_n, Y_hat_n, Y_hat_N, w, K, fold){
  Y_hat_N_var <- sum(apply(Y_hat_N, 2, var)) *lam^2 / (K^2*N)
  rectifier_var <-
    sum(sapply(1:K, \(j) var(w[fold == j]*(Y_n[fold == j] - lam*Y_hat_n[fold == j]))) *
          tapply(fold, fold, length)) / (n^2)
  std <- sqrt(Y_hat_N_var+rectifier_var)
  return(std)
}
gen_res_cross <- function(n, N, Y_n, Y_hat_n, Y_N, Y_hat_N, w, K, fold, lam = NULL, alpha = 0.5) {
  if(is.null(lam)) {
    lam <- lambda_ppi_cross(n, N, Y_n, Y_hat_n, Y_hat_N, w, K, fold)
  }
  pointest <- pointest_ppi(lam, n, N, Y_n, Y_hat_n, Y_hat_N, w)
  std <- std_ppi_cross(lam, n, N, Y_n, Y_hat_n, Y_hat_N, w, K, fold)
  ci <- ci_ppi(pointest, std, alpha)
  return(c(
    lam = lam,
    truemean = mean(Y_N),
    pointest = pointest,
    std = std,
    ll = ci[1],
    ul = ci[2],
    ci_width = ci[2] - ci[1],
    coverage = (ci[1] <= mean(Y_N) & mean(Y_N) <= ci[2])
  ))
}
```

```r
library(parallel)
# number of cores
num_cores <- detectCores()-1
# create cluster
cl <- makeCluster(num_cores)

# export data and functions to cluster
clusterExport(cl, c(
  "iid_settings", "B", "K", "mu_0", "sigma_0", "p_0", "K", "B",
  "gen_data", "comb_data",
  "lambda_ppi", "std_ppi", "gen_res",
  "lambda_ppi_cross", "std_ppi_cross", "gen_res_cross",
  "pointest_ppi", "ci_ppi"
))
```

```r
set.seed(7065)
# run simulation in parallel
iid_simulation <- parLapply(cl, 1:B, function(i){
  combined_data_list <- mapply(
    comb_data,
    iid_settings$n,
    iid_settings$N,
    mu_0, sigma_0, p_0,
    mu_0, sigma_0, p_0,
    iid_settings$sigma_e,
    SIMPLIFY = F
  )

  # ---- PPI ----
  lam0_ppi_res <- sapply(
    combined_data_list,
    \(x) gen_res(
      n = x$settings["n"],
      N = x$settings["N"],
      Y_n = x$labeled$Y,
      Y_hat_n = x$labeled$Y_hat,
      Y_N = x$unlabeled$Y,
      Y_hat_N = x$unlabeled$Y_hat,
      w = x$labeled$w,
      lam = 0,
      alpha = 0.05
    )
  )
  lam1_ppi_res <- sapply(
    combined_data_list,
    \(x) gen_res(
      n = x$settings["n"],
      N = x$settings["N"],
      Y_n = x$labeled$Y,
      Y_hat_n = x$labeled$Y_hat,
      Y_N = x$unlabeled$Y,
      Y_hat_N = x$unlabeled$Y_hat,
      w = x$labeled$w,
      lam = 1,
      alpha = 0.05
    )
  )
  lamopt_ppi_res <- sapply(
    combined_data_list,
    \(x) gen_res(
      n = x$settings["n"],
      N = x$settings["N"],
      Y_n = x$labeled$Y,
      Y_hat_n = x$labeled$Y_hat,
      Y_N = x$unlabeled$Y,
      Y_hat_N = x$unlabeled$Y_hat,
      w = x$labeled$w,
      alpha = 0.05
```

```
  )
)

# ---- Cross-PPI ----
# add K to labeled data
combined_data_list_cross <- lapply(
  combined_data_list,
  \(x) {
    x$labeled$fold <- sample(1:K, nrow(x$labeled), replace = TRUE)
    return(x)
  }
)
# predict Y_hat_N by K models
combined_data_list_cross <- lapply(
  combined_data_list_cross,
  \(x) {
    labeled <- x$labeled
    fit_list <- lapply(
      1:K,
      \(j) lm(Y ~ X1+X2, data = labeled, subset = fold == j)
    )
    k_coef <- sapply(fit_list, unlist(coef)) |> t() |> as.data.frame()
    labeled$Y_hat <- apply(labeled, 1, \(obs) {
      coef <- k_coef[obs["fold"],] |> unlist()
      coef[1] + coef[2]*obs["X1"] + coef[3]*obs["X2"]
    })

    unlabeled <- x$unlabeled
    Y_hat_N <- data.frame(lapply(
      fit_list,
      \(fit) predict(fit, newdata = unlabeled[,1:2])
    ))

    x <- list(
      settings = x[["settings"]],
      labeled = labeled,
      unlabeled = x[["unlabeled"]],
      coef = k_coef,
      Y_hat_N = Y_hat_N
    )
    return(x)
  }
)

lam1_ppi_res_cross <- sapply(
  combined_data_list_cross,
  \(x) gen_res_cross(
    n = x$settings["n"],
    N = x$settings["N"],
    Y_n = x$labeled$Y,
    Y_hat_n = x$labeled$Y_hat,
    Y_N = x$unlabeled$Y,
    Y_hat_N = x$Y_hat_N,
```

```
    w = x$labeled$w,
    K = K,
    fold = x$labeled$fold,
    lam = 1,
    alpha = 0.05
  )
)
lamopt_ppi_res_cross <- sapply(
  combined_data_list_cross,
  \(x) gen_res_cross(
    n = x$settings["n"],
    N = x$settings["N"],
    Y_n = x$labeled$Y,
    Y_hat_n = x$labeled$Y_hat,
    Y_N = x$unlabeled$Y,
    Y_hat_N = x$Y_hat_N,
    w = x$labeled$w,
    K = K,
    fold = x$labeled$fold,
    alpha = 0.05
  )
)

# ---- combine results ----
# CLT
res_summary_lam0 <- data.frame(
  iid_settings,
  t(lam0_ppi_res) |> as.data.frame()
)
# standard PPI
res_summary_lam1 <- data.frame(
  iid_settings,
  t(lam1_ppi_res) |> as.data.frame()
)
# PPI++
res_summary_lamopt <- data.frame(
  iid_settings,
  t(lamopt_ppi_res) |> as.data.frame()
)
# Cross-PPI
res_summary_lam1_cross <- data.frame(
  iid_settings,
  t(lam1_ppi_res_cross) |> as.data.frame()
)
# Cross-PPI++
res_summary_lamopt_cross <- data.frame(
  iid_settings,
  t(lamopt_ppi_res_cross) |> as.data.frame()
)

colnames(res_summary_lam0) <-
  colnames(res_summary_lam1) <-
  colnames(res_summary_lamopt) <-
```

```
    colnames(res_summary_lam1_cross) <-
    colnames(res_summary_lamopt_cross) <-
    c("n", "N", "noise_level", "lam",
      "truemean", "pointest", "std", "ll", "ul",
      "ci_width", "coverage")
  res_summary_all <- rbind(
    cbind(res_summary_lam0, method = "classical CLT"),
    cbind(res_summary_lam1, method = "standard PPI"),
    cbind(res_summary_lamopt, method = "PPI++"),
    cbind(res_summary_lam1_cross, method = "Cross PPI"),
    cbind(res_summary_lamopt_cross, method = "Cross PPI++")
  )
  return(res_summary_all)
})

saveRDS(iid_simulation, "iid_simulation.rds")
stopCluster(cl)
```

```
library(parallel)
# number of cores
num_cores <- detectCores()-1
# create cluster
cl <- makeCluster(num_cores)

# export data and functions to cluster
clusterExport(cl, c(
  "covshift_settings", "B", "K", "mu_0", "sigma_0", "p_0", "K", "B",
  "gen_data", "comb_data", "w1", "w2", "w3",
  "lambda_ppi", "std_ppi", "gen_res",
  "lambda_ppi_cross", "std_ppi_cross", "gen_res_cross",
  "pointest_ppi", "ci_ppi"
))

set.seed(7065)
# run simulation in parallel
covshift_simulation <- parLapply(cl, 1:B, function(i){
  combined_data_list <- mapply(
    comb_data,
    covshift_settings$n,
    covshift_settings$N,
    mu_0, sigma_0, p_0,
    covshift_settings$mu,
    covshift_settings$sigma,
    covshift_settings$p,
    covshift_settings$sigma_e,
    SIMPLIFY = F
  )

  # ---- PPI ----
  lam0_ppi_res <- sapply(
    combined_data_list,
    \(x) gen_res(
      n = x$settings["n"],
```

```
    N = x$settings["N"],
    Y_n = x$labeled$Y,
    Y_hat_n = x$labeled$Y_hat,
    Y_N = x$unlabeled$Y,
    Y_hat_N = x$unlabeled$Y_hat,
    w = x$labeled$w,
    lam = 0,
    alpha = 0.05
  )
)
lam1_ppi_res <- sapply(
  combined_data_list,
  \(x) gen_res(
    n = x$settings["n"],
    N = x$settings["N"],
    Y_n = x$labeled$Y,
    Y_hat_n = x$labeled$Y_hat,
    Y_N = x$unlabeled$Y,
    Y_hat_N = x$unlabeled$Y_hat,
    w = x$labeled$w,
    lam = 1,
    alpha = 0.05
  )
)
lamopt_ppi_res <- sapply(
  combined_data_list,
  \(x) gen_res(
    n = x$settings["n"],
    N = x$settings["N"],
    Y_n = x$labeled$Y,
    Y_hat_n = x$labeled$Y_hat,
    Y_N = x$unlabeled$Y,
    Y_hat_N = x$unlabeled$Y_hat,
    w = x$labeled$w,
    alpha = 0.05
  )
)
lamopt_nodensratio_ppi_res <- sapply(
  combined_data_list,
  \(x) gen_res(
      n = x$settings["n"],
      N = x$settings["N"],
      Y_n = x$labeled$Y,
      Y_hat_n = x$labeled$Y_hat,
      Y_N = x$unlabeled$Y,
      Y_hat_N = x$unlabeled$Y_hat,
      w = 1,
      alpha = 0.05
    )
)

# ---- Cross-PPI ----
# add K to labeled data
```

```r
combined_data_list_cross <- lapply(
  combined_data_list,
  \(x) {
    x$labeled$fold <- sample(1:K, nrow(x$labeled), replace = TRUE)
    return(x)
  }
)
# predict Y_hat_N by K models
combined_data_list_cross <- lapply(
  combined_data_list_cross,
  \(x) {
    labeled <- x$labeled
    fit_list <- lapply(
      1:K,
      \(j) lm(Y ~ X1+X2, data = labeled, subset = fold == j)
    )
    k_coef <- sapply(fit_list, unlist(coef)) |> t() |> as.data.frame()
    labeled$Y_hat <- apply(labeled, 1, \(obs) {
      coef <- k_coef[obs["fold"],] |> unlist()
      coef[1] + coef[2]*obs["X1"] + coef[3]*obs["X2"]
    })

    unlabeled <- x$unlabeled
    Y_hat_N <- data.frame(lapply(
      fit_list,
      \(fit) predict(fit, newdata = unlabeled[,1:2])
    ))

    x <- list(
      settings = x[["settings"]],
      labeled = labeled,
      unlabeled = x[["unlabeled"]],
      coef = k_coef,
      Y_hat_N = Y_hat_N
    )
    return(x)
  }
)

lamopt_ppi_res_cross <- sapply(
  combined_data_list_cross,
  \(x) gen_res_cross(
    n = x$settings["n"],
    N = x$settings["N"],
    Y_n = x$labeled$Y,
    Y_hat_n = x$labeled$Y_hat,
    Y_N = x$unlabeled$Y,
    Y_hat_N = x$Y_hat_N,
    w = x$labeled$w,
    K = K,
    fold = x$labeled$fold,
    alpha = 0.05
  )
```

```
  )

  # ---- combine results ----
  # CLT
  res_summary_lam0 <- data.frame(
    covshift_settings,
    t(lam0_ppi_res) |> as.data.frame()
  )
  # standard PPI
  res_summary_lam1 <- data.frame(
    covshift_settings,
    t(lam1_ppi_res) |> as.data.frame()
  )
  # PPI++
  res_summary_lamopt <- data.frame(
    covshift_settings,
    t(lamopt_ppi_res) |> as.data.frame()
  )
  # PPI++, no density ratio
  res_summary_lamopt_nodensratio <- data.frame(
    covshift_settings,
    t(lamopt_nodensratio_ppi_res) |> as.data.frame()
  )
  # Cross-PPI++
  res_summary_lamopt_cross <- data.frame(
    covshift_settings,
    t(lamopt_ppi_res_cross) |> as.data.frame()
  )

  colnames(res_summary_lam0) <-
    colnames(res_summary_lam1) <-
    colnames(res_summary_lamopt) <-
    colnames(res_summary_lamopt_nodensratio) <-
    colnames(res_summary_lamopt_cross) <-
    c("n", "N", "noise_level", "mu", "sigma", "p", "scenario", "lam",
      "truemean", "pointest", "std", "ll", "ul",
      "ci_width", "coverage")
  res_summary_all <- rbind(
    cbind(res_summary_lam0, method = "classical CLT"),
    cbind(res_summary_lam1, method = "standard PPI"),
    cbind(res_summary_lamopt, method = "PPI++"),
    cbind(res_summary_lamopt_nodensratio, method = "PPI++, no density ratio"),
    cbind(res_summary_lamopt_cross, method = "Cross PPI++")
  )
  return(res_summary_all)
})

saveRDS(covshift_simulation, "covshift_simulation.rds")
stopCluster(cl)
```