Position: Don’t Use the CLT in LLM Evals With Fewer Than a Few Hundred Datapoints

Sam Bowyer, Laurence Aitchison, and Desi R. Ivanova



June 12, 2025

Evaluations

A statistical approach to model evaluations

19 Nov 2024

Read the paper

Suppose an AI model outperforms another model on a benchmark of interest—testing its general knowledge, for example, or its ability to solve computer-coding questions. Is the difference in capabilities real, or could one model simply have gotten lucky in the choice of questions on the benchmark?

With the amount of public interest in AI model evaluations—informally called "evals"—this question remains surprisingly understudied among the AI research community. This month, we published a new research paper that attempts to answer the question rigorously. Drawing on statistical theory and the experiment design literature, the paper makes a number of recommendations to the AI research community for reporting eval results in a scientifically informative way. In this post, we briefly go over the reporting recommendations, and the logic behind them.

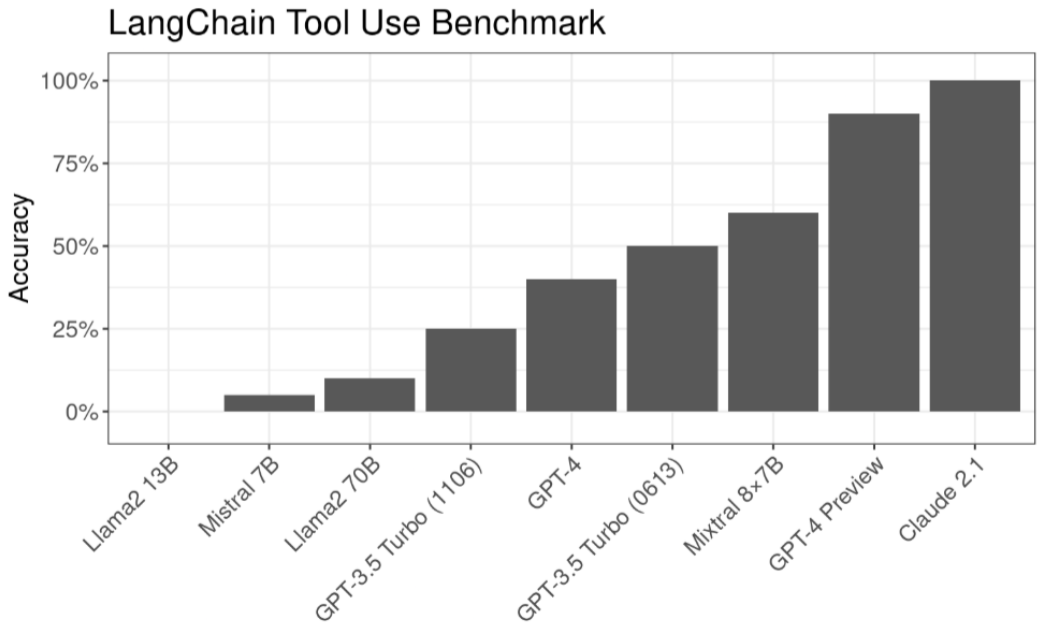
Recommendation #1: Use the Central Limit Theorem

Evals often consist of hundreds or thousands of unrelated questions. MMLU, for instance, contains questions as diverse as:

TL;DR

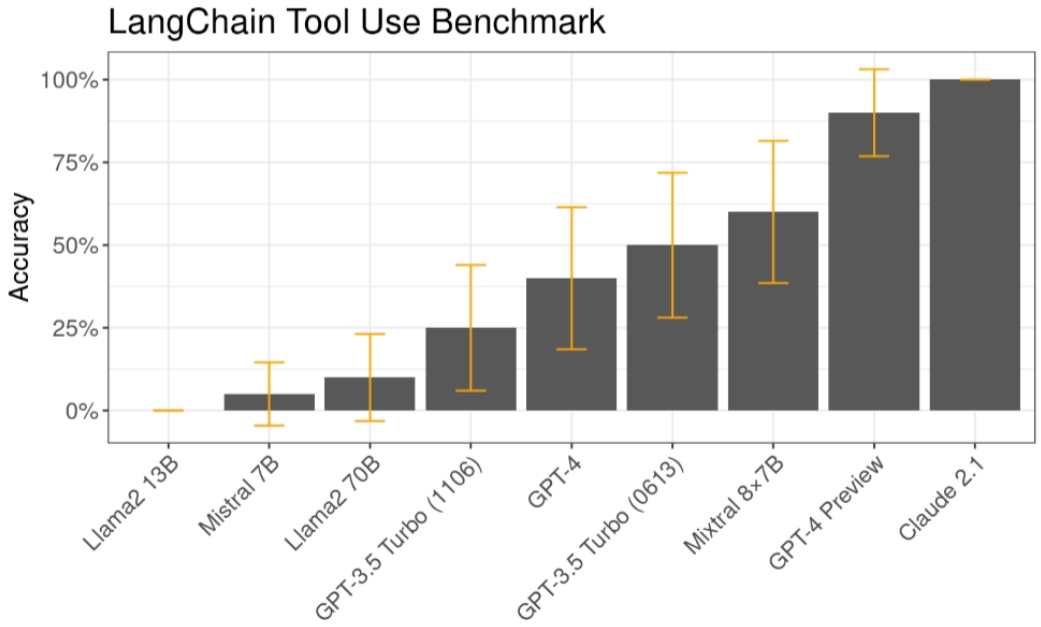
TL;DR

- Error bars are important for evals



TL;DR

- Error bars are important for evals
- CLT-based methods are (increasingly) unwise



TL;DR

- Error bars are important for evals
- CLT-based methods are (increasingly) unwise
- We can do a lot better, very easily

```python
# y is a length N binary "eval" vector
from scipy.stats import binomtest, beta

S, N = y.sum(), len(y) # total successes & questions
result = binomtest(k=S, n=N)

# 95% Wilson score and Clopper-Pearson intervals
wilson_ci = result.proportion_ci("wilson", 0.95)
cp_ci = result.proportion_ci("exact", 0.95)

# Bayesian Credible interval
posterior = beta(1+S, 1+(N-S))
bayes_ci = posterior.interval(confidence=0.95)
```

Central Limit Theorem (CLT)

If $X_1, \ldots, X_N$ are IID r.v.s with mean $\mu \in \mathbb{R}$ and finite variance σ^2, then

$$\sqrt{N}(\hat{\mu} - \mu) \xrightarrow{d} \mathcal{N}\left(0, \sigma^2\right) \text{ as } N \to \infty,$$

where $\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} X_i$ is the sample mean.

Central Limit Theorem (CLT)

If $X_1, \ldots, X_N$ are IID r.v.s with mean $\mu \in \mathbb{R}$ and finite variance σ^2, then

$$\sqrt{N}(\hat{\mu} - \mu) \xrightarrow{d} \mathcal{N}\left(0, \sigma^2\right) \text{ as } N \to \infty,$$

where $\hat{\mu} = \frac{1}{N} \sum_{i=1}^{N} X_i$ is the sample mean.

(We generally estimate $\sigma^2 \approx \hat{\sigma}^2 = \frac{1}{N-1} \sum_{i=1}^{N} (X_i - \hat{\mu})^2$.)

Central Limit Theorem (CLT) - Confidence Intervals

We construct CLT-based confidence intervals at confidence level $1 - \alpha \in [0, 1]$ as

$$\mathrm{CI}_{1-\alpha}(\mu) = \hat{\mu} \pm z_{\alpha/2}\mathrm{SE}(\hat{\mu}),$$

where $z_{\alpha/2}$ is the $100(1 - \alpha/2)$-th percentile of the standard normal distribution and

$$\mathrm{SE}(\hat{\mu}) = \sqrt{\hat{\sigma}^2/N}$$

is the standard error of the sample mean.

Central Limit Theorem (CLT) - Confidence Intervals

We construct CLT-based confidence intervals at confidence level $1-\alpha \in [0,1]$ as

$$\mathrm{CI}_{1-\alpha}(\mu) = \hat{\mu} \pm z_{\alpha/2}\mathrm{SE}(\hat{\mu}),$$

where $z_{\alpha/2}$ is the $100(1-\alpha/2)$-th percentile of the standard normal distribution and

$$\mathrm{SE}(\hat{\mu}) = \sqrt{\hat{\sigma}^2/N}$$

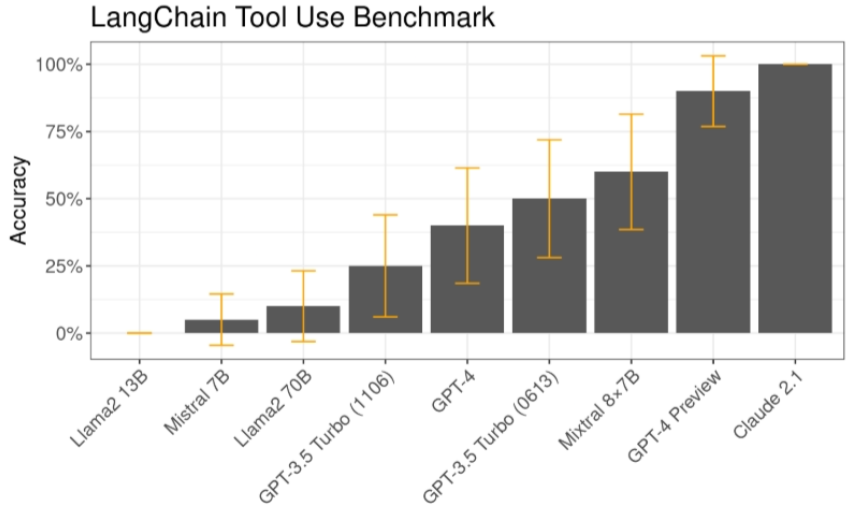
is the standard error of the sample mean.

For binary data (e.g. correct/incorrect), $X_i \sim \mathrm{Bernoulli}(\theta)$, we can use the Bernoulli variance formula:

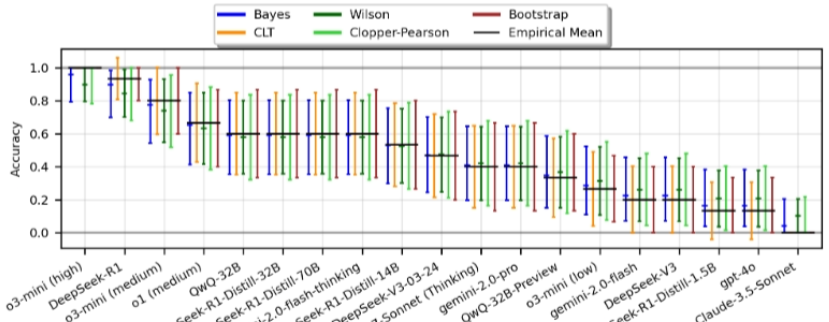
$$\mathrm{SE}(\hat{\theta}) = \sqrt{\frac{\hat{\theta}(1-\hat{\theta})}{N}}.$$

Real-world failures

As models get better (and more expensive), benchmarks get harder and smaller, posing problems for the CLT. (E.g. Math Arena's AIME II 2025 Benchmark has N=15 competition maths problems.)



Langchain Typewriter Tool Use Benchmark (N=20)

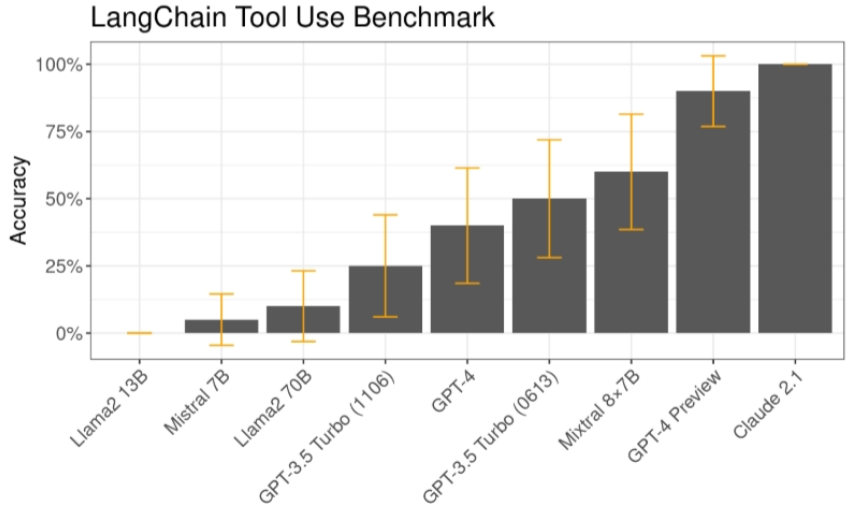


Math Arena's AIME II 2025 Benchmark (N=15)

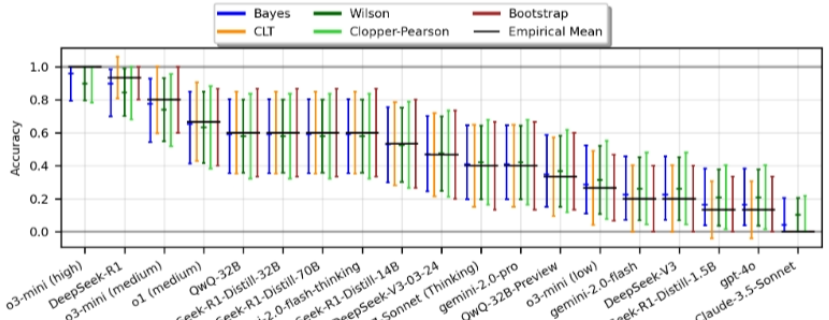
Real-world failures

As models get better (and more expensive), benchmarks get harder and smaller, posing problems for the CLT. (E.g. Math Arena's AIME II 2025 Benchmark has N=15 competition maths problems.)

- Error bars can collapse to zero-width.



Langchain Typewriter Tool Use Benchmark (N=20)

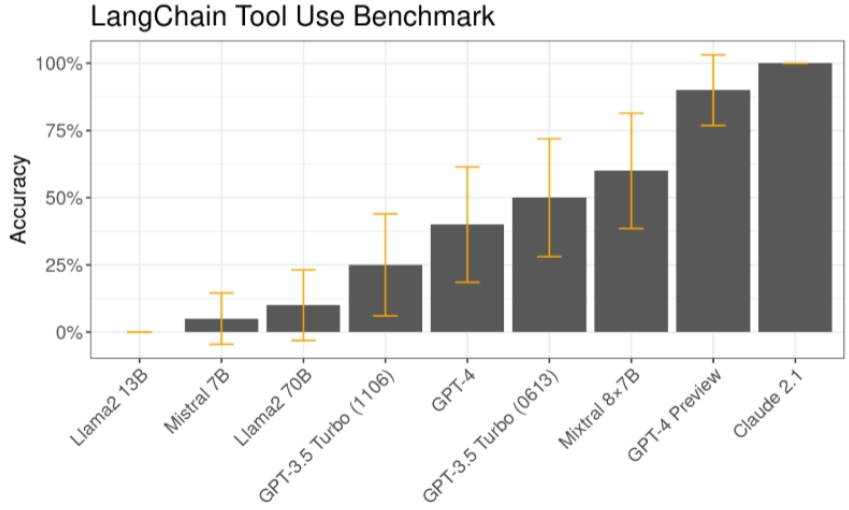


Math Arena's AIME II 2025 Benchmark (N=15)

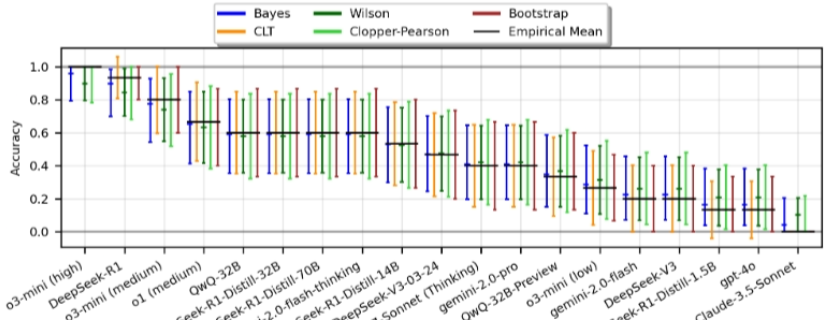
Real-world failures

As models get better (and more expensive), benchmarks get harder and smaller, posing problems for the CLT. (E.g. Math Arena's AIME II 2025 Benchmark has N=15 competition maths problems.)

- Error bars can collapse to zero-width.
- Error bars can extend past [0,1].



Langchain Typewriter Tool Use Benchmark (N=20)



Math Arena's AIME II 2025 Benchmark (N=15)

Alternative #1 – Beta-Binomial Model

Treat the data as IID Bernoulli with a uniform prior on the parameter θ.

$$\theta \sim \text{Beta}(1, 1) = \text{Uniform}[0, 1]$$
$$y_i \sim \text{Bernoulli}(\theta) \text{ for } i = 1, \ldots N$$

Alternative #1 – Beta-Binomial Model

Treat the data as IID Bernoulli with a uniform prior on the parameter θ.

$$\theta \sim \text{Beta}(1, 1) = \text{Uniform}[0, 1]$$
$$y_i \sim \text{Bernoulli}(\theta) \text{ for } i = 1, \ldots N$$

Alternative #1 – Beta-Binomial Model

Treat the data as IID Bernoulli with a uniform prior on the parameter θ.

$$\theta \sim \text{Beta}(1, 1) = \text{Uniform}[0, 1]$$
$$y_i \sim \text{Bernoulli}(\theta) \text{ for } i = 1, \ldots N$$

We say y_i is correct if $y_i = 1$ and incorrect if $y_i = 0$. (Think of θ as the probability of correctness.)

Alternative #1 – Beta-Binomial Model

Treat the data as IID Bernoulli with a uniform prior on the parameter θ.

$$\theta \sim \text{Beta}(1,1) = \text{Uniform}[0,1]$$
$$y_i \sim \text{Bernoulli}(\theta) \text{ for } i = 1, \dots N$$

We say y_i is correct if $y_i = 1$ and incorrect if $y_i = 0$. (Think of θ as the probability of correctness.)

$$\mathbb{P}(\theta | y_{1:N}) = \text{Beta}\left(1 + \sum_{i=1}^{N} y_i, 1 + \sum_{i=1}^{N} (1 - y_i)\right)$$

Alternative #1 – Beta-Binomial Model

Treat the data as IID Bernoulli with a uniform prior on the parameter θ.

$$\theta \sim \text{Beta}(1,1) = \text{Uniform}[0,1]$$
$$y_i \sim \text{Bernoulli}(\theta) \text{ for } i = 1, \dots N$$

We say y_i is correct if $y_i = 1$ and incorrect if $y_i = 0$. (Think of θ as the probability of correctness.)

$$\mathbb{P}(\theta | y_{1:N}) = \text{Beta}\left(1 + \sum_{i=1}^{N} y_i, 1 + \sum_{i=1}^{N} (1 - y_i)\right)$$

Construct a quantile-based Bayesian *credible interval* for θ from the closed form posterior.

Alternative #1 – Beta-Binomial Model

Treat the data as IID Bernoulli with a uniform prior on the parameter θ.

$$\theta \sim \text{Beta}(1,1) = \text{Uniform}[0,1]$$
$$y_i \sim \text{Bernoulli}(\theta) \text{ for } i = 1, \dots N$$

We say y_i is correct if $y_i = 1$ and incorrect if $y_i = 0$. (Think of θ as the probability of correctness.)

$$\mathbb{P}(\theta | y_{1:N}) = \text{Beta}\left(1 + \sum_{i=1}^{N} y_i, 1 + \sum_{i=1}^{N} (1 - y_i)\right)$$

Construct a quantile-based Bayesian *credible interval* for θ from the closed form posterior.

```
# y is a length N binary "eval" vector
S, N = y.sum(), len(y) # total successes & questions

# Bayesian Credible interval
posterior = scipy.stats.beta(1+S, 1+(N-S))
bayes_ci = posterior.interval(confidence=0.95)
```

Frequentist vs. Bayesian Intervals

Frequentist vs. Bayesian Intervals

- Frequentist *confidence interval:*

Frequentist vs. Bayesian Intervals

- Frequentist *confidence interval:*
  - The parameter is fixed but unknown, the interval is a random variable depending on the data.

Frequentist vs. Bayesian Intervals

- Frequentist *confidence interval:*
  - The parameter is fixed but unknown, the interval is a random variable depending on the data.
  - "*If we repeated the experiment many times,* $100 \times (1-\alpha)\%$ *of the time the interval would contain the true parameter.*"

Frequentist vs. Bayesian Intervals

- Frequentist *confidence interval:*
  - The parameter is fixed but unknown, the interval is a random variable depending on the data.
  - "*If we repeated the experiment many times,* $100 \times (1-\alpha)\%$ *of the time the interval would contain the true parameter.*"
- Bayesian *credible interval:*

Frequentist vs. Bayesian Intervals

- Frequentist *confidence interval:*
  - The parameter is fixed but unknown, the interval is a random variable depending on the data.
  - "*If we repeated the experiment many times,* $100 \times (1-\alpha)\%$ *of the time the interval would contain the true parameter.*"
- Bayesian *credible interval:*
  - The parameter is random, we infer the posterior distribution of the parameter given the data.

Frequentist vs. Bayesian Intervals

- Frequentist *confidence interval:*
  - The parameter is fixed but unknown, the interval is a random variable depending on the data.
  - "*If we repeated the experiment many times,* $100 \times (1-\alpha)\%$ *of the time the interval would contain the true parameter.*"
- Bayesian *credible interval:*
  - The parameter is random, we infer the posterior distribution of the parameter given the data.
  - "*There is a* $100 \times (1-\alpha)\%$ *probability that the interval contains the true parameter. (Under some modelling assumptions.)*"

Interval Comparison

We'll focus on two metrics for evaluating intervals:

Interval Comparison

We'll focus on two metrics for evaluating intervals:

- Coverage

Interval Comparison

We'll focus on two metrics for evaluating intervals:

- Coverage
  - What proportion of the time does a $1 - \alpha$ confidence-level interval *actually contain* the true underlying value of θ?

Interval Comparison

We'll focus on two metrics for evaluating intervals:

- Coverage
  - What proportion of the time does a $1 - \alpha$ confidence-level interval *actually contain* the true underlying value of θ?
  - Ideally: *actual* coverage $=$ *nominal* coverage (i.e. $1 - \alpha$).

Interval Comparison

We'll focus on two metrics for evaluating intervals:

- Coverage
  - What proportion of the time does a $1 - \alpha$ confidence-level interval *actually contain* the true underlying value of θ?
  - Ideally: *actual* coverage $=$ *nominal* coverage (i.e. $1 - \alpha$).
  - (This is a frequentist measure really, but still a useful one for evaluating Bayesian methods too.)

Interval Comparison

We'll focus on two metrics for evaluating intervals:

- Coverage
  - What proportion of the time does a $1 - \alpha$ confidence-level interval *actually contain* the true underlying value of θ?
  - Ideally: *actual* coverage $=$ *nominal* coverage (i.e. $1 - \alpha$).
  - (This is a frequentist measure really, but still a useful one for evaluating Bayesian methods too.)
- Width

Interval Comparison

We'll focus on two metrics for evaluating intervals:

- Coverage
  - What proportion of the time does a $1 - \alpha$ confidence-level interval *actually contain* the true underlying value of θ?
  - Ideally: *actual* coverage $=$ *nominal* coverage (i.e. $1 - \alpha$).
  - (This is a frequentist measure really, but still a useful one for evaluating Bayesian methods too.)
- Width
  - Ideally, our intervals would be as tight as possible.

Interval Comparison

We'll focus on two metrics for evaluating intervals:

- Coverage
  - What proportion of the time does a $1 - \alpha$ confidence-level interval *actually contain* the true underlying value of θ?
  - Ideally: *actual* coverage $=$ *nominal* coverage (i.e. $1 - \alpha$).
  - (This is a frequentist measure really, but still a useful one for evaluating Bayesian methods too.)
- Width
  - Ideally, our intervals would be as tight as possible.
  - *"For a certain width of interval, method X achieves the highest coverage."*

Experiment Setup

We have to rely on synthetic data so that we *know* the true parameter θ.

Experiment Setup

We have to rely on synthetic data so that we *know* the true parameter θ.

- Draw $\theta \sim \text{Uniform}[0, 1]$.

Experiment Setup

We have to rely on synthetic data so that we *know* the true parameter θ.

- Draw $\theta \sim \mathrm{Uniform}[0, 1]$.
- Draw $N \in \{3, 10, 30, 100\}$ IID Bernoulli datapoints with parameter θ.

Experiment Setup

We have to rely on synthetic data so that we *know* the true parameter θ.

- Draw $\theta \sim \mathrm{Uniform}[0, 1]$.
- Draw $N \in \{3, 10, 30, 100\}$ IID Bernoulli datapoints with parameter θ.
- Construct $1 - \alpha$ confidence-level intervals for θ using both methods with various α values.

Experiment Setup

We have to rely on synthetic data so that we *know* the true parameter θ.

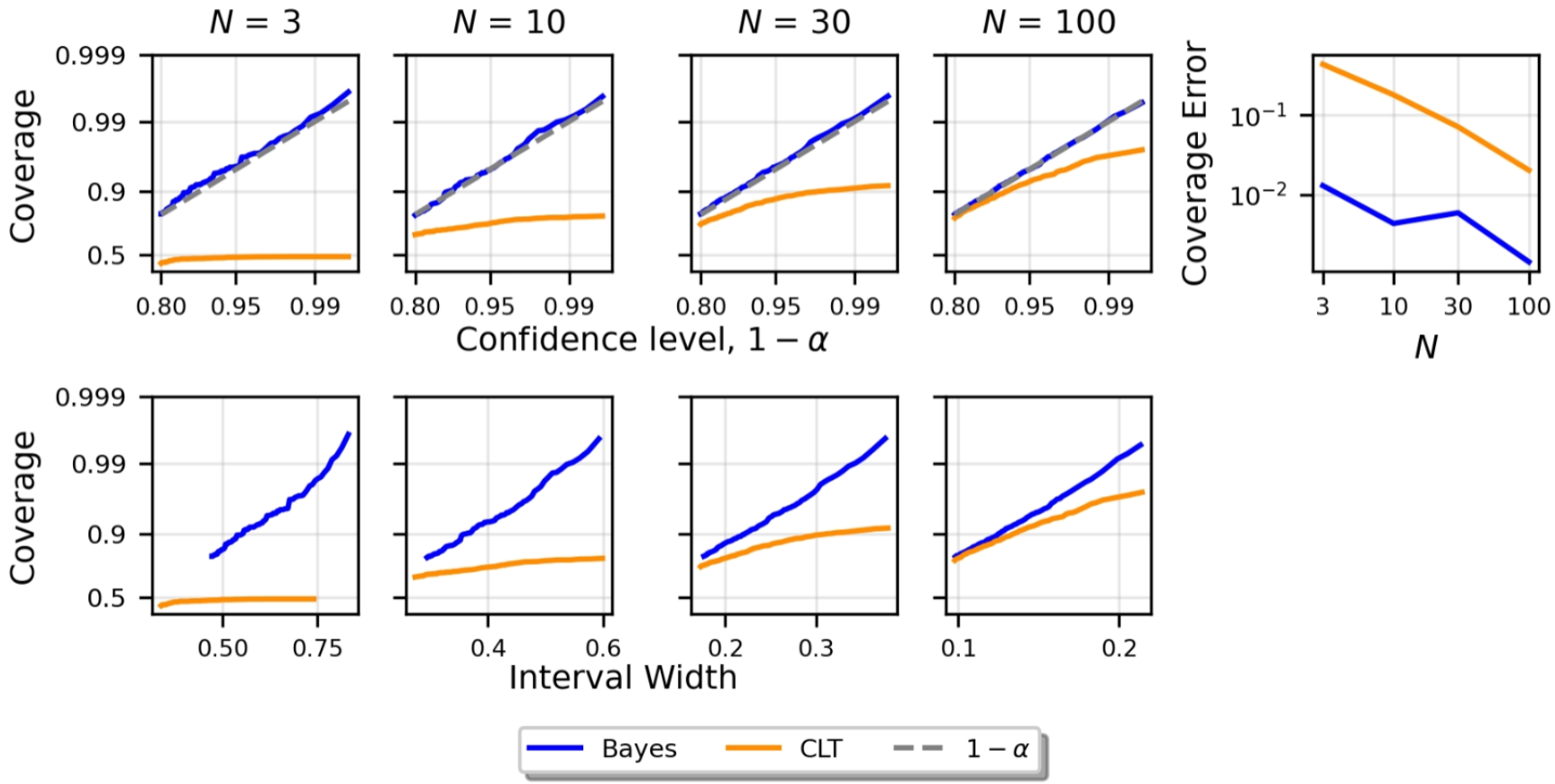
- Draw $\theta \sim \mathrm{Uniform}[0, 1]$.
- Draw $N \in \{3, 10, 30, 100\}$ IID Bernoulli datapoints with parameter θ.
- Construct $1 - \alpha$ confidence-level intervals for θ using both methods with various α values.
- Repeat for this 20,000 times for each of the 4 values of N.

Experiment Setup

We have to rely on synthetic data so that we *know* the true parameter θ.

- Draw $\theta \sim \text{Uniform}[0, 1]$.
- Draw $N \in \{3, 10, 30, 100\}$ IID Bernoulli datapoints with parameter θ.
- Construct $1 - \alpha$ confidence-level intervals for θ using both methods with various α values.
- Repeat for this 20,000 times for each of the 4 values of N.
- Compute the coverage and average width of the intervals.

IID Questions Setting - Bayes vs. CLT



Alternative #2 – Wilson Score Intervals

$$\mathrm{CI}_{1-\alpha,\mathrm{Wilson}}(\theta) = \frac{\hat{\theta} + \frac{z^2_{\alpha/2}}{2N}}{1 + \frac{z^2_{\alpha/2}}{N}} \pm \frac{\frac{z_{\alpha/2}}{2N}}{1 + \frac{z^2_{\alpha/2}}{N}} \sqrt{4N\hat{\theta}(1-\hat{\theta}) + z^2_{\alpha/2}}$$

where $z_{\alpha/2}$ is the $100(1-\alpha/2)$-th percentile of the standard normal distribution.

Alternative #2 – Wilson Score Intervals

$$\mathrm{CI}_{1-\alpha,\mathrm{Wilson}}(\theta) = \frac{\hat{\theta} + \frac{z_{\alpha/2}^2}{2N}}{1 + \frac{z_{\alpha/2}^2}{N}} \pm \frac{\frac{z_{\alpha/2}}{2N}}{1 + \frac{z_{\alpha/2}^2}{N}} \sqrt{4N\hat{\theta}(1-\hat{\theta}) + z_{\alpha/2}^2}$$

where $z_{\alpha/2}$ is the $100(1-\alpha/2)$-th percentile of the standard normal distribution.

- Not centered at $\hat{\theta}$.

Alternative #2 – Wilson Score Intervals

$$\text{CI}_{1-\alpha,\text{Wilson}}(\theta) = \frac{\hat{\theta} + \frac{z_{\alpha/2}^2}{2N}}{1 + \frac{z_{\alpha/2}^2}{N}} \pm \frac{\frac{z_{\alpha/2}}{2N}}{1 + \frac{z_{\alpha/2}^2}{N}} \sqrt{4N\hat{\theta}(1-\hat{\theta}) + z_{\alpha/2}^2}$$

where $z_{\alpha/2}$ is the $100(1-\alpha/2)$-th percentile of the standard normal distribution.

- Not centered at $\hat{\theta}$.
- Based on a normal approximation to the binomial distribution (but **not** the CLT).

Alternative #2 – Wilson Score Intervals

$$\mathrm{CI}_{1-\alpha,\mathrm{Wilson}}(\theta) = \frac{\hat{\theta} + \frac{z_{\alpha/2}^2}{2N}}{1 + \frac{z_{\alpha/2}^2}{N}} \pm \frac{\frac{z_{\alpha/2}}{2N}}{1 + \frac{z_{\alpha/2}^2}{N}} \sqrt{4N\hat{\theta}(1-\hat{\theta}) + z_{\alpha/2}^2}$$

where $z_{\alpha/2}$ is the $100(1-\alpha/2)$-th percentile of the standard normal distribution.

- Not centered at $\hat{\theta}$.
- Based on a normal approximation to the binomial distribution (but **not** the CLT).

```
# y is a length N binary "eval" vector
S, N = y.sum(), len(y) # total successes & questions
result = scipy.stats.binomtest(k=S, n=N)

# 95% Wilson score interval
wilson_ci = result.proportion_ci("wilson", 0.95)
```

Alternative #3 – Clopper-Pearson Exact Intervals

$$\mathrm{CI}_{1-\alpha,\mathrm{CP}}(\theta) = [\theta_{\text{lower}}, \theta_{\text{upper}}]$$

$$\theta_{\text{lower}} = B\left(\frac{\alpha}{2}, \sum_{i=1}^{N} y_i, 1 + \sum_{i=1}^{N}(1 - y_i)\right) \quad \text{and} \quad \theta_{\text{upper}} = B\left(1 - \frac{\alpha}{2}, 1 + \sum_{i=1}^{N} y_i, \sum_{i=1}^{N}(1 - y_i)\right)$$

where $B(\alpha, a, b)$ is the α-th quantile of the Beta(a, b) distribution.

Alternative #3 – Clopper-Pearson Exact Intervals

$$\mathrm{CI}_{1-\alpha,\mathrm{CP}}(\theta) = [\theta_{\text{lower}}, \theta_{\text{upper}}]$$

$$\theta_{\text{lower}} = B\left(\frac{\alpha}{2}, \sum_{i=1}^{N} y_i, 1 + \sum_{i=1}^{N}(1 - y_i)\right) \quad \text{and} \quad \theta_{\text{upper}} = B\left(1 - \frac{\alpha}{2}, 1 + \sum_{i=1}^{N} y_i, \sum_{i=1}^{N}(1 - y_i)\right)$$

where $B(\alpha, a, b)$ is the α-th quantile of the Beta(a, b) distribution.

- Guaranteed to never under-cover (very conservative method; 'worst-case' approach).

Alternative #3 – Clopper-Pearson Exact Intervals

$$\mathrm{CI}_{1-\alpha,\mathrm{CP}}(\theta) = [\theta_{\text{lower}}, \theta_{\text{upper}}]$$

$$\theta_{\text{lower}} = B\left(\frac{\alpha}{2}, \sum_{i=1}^{N} y_i, 1 + \sum_{i=1}^{N}(1 - y_i)\right) \quad \text{and} \quad \theta_{\text{upper}} = B\left(1 - \frac{\alpha}{2}, 1 + \sum_{i=1}^{N} y_i, \sum_{i=1}^{N}(1 - y_i)\right)$$

where $B(\alpha, a, b)$ is the α-th quantile of the Beta(a, b) distribution.

- Guaranteed to never under-cover (very conservative method; 'worst-case' approach).
  - Contains all $\theta \in [0, 1]$ that would not reject $H_0 : \theta = \hat{\theta}$ in favour of $H_1 : \theta \neq \hat{\theta}$ at confidence level α.

Alternative #3 – Clopper-Pearson Exact Intervals

$$\mathrm{CI}_{1-\alpha,\mathrm{CP}}(\theta) = [\theta_{\text{lower}}, \theta_{\text{upper}}]$$

$$\theta_{\text{lower}} = B\left(\frac{\alpha}{2}, \sum_{i=1}^{N} y_i, 1 + \sum_{i=1}^{N}(1 - y_i)\right) \quad \text{and} \quad \theta_{\text{upper}} = B\left(1 - \frac{\alpha}{2}, 1 + \sum_{i=1}^{N} y_i, \sum_{i=1}^{N}(1 - y_i)\right)$$

where $B(\alpha, a, b)$ is the α-th quantile of the Beta(a, b) distribution.

- Guaranteed to never under-cover (very conservative method; 'worst-case' approach).
  - Contains all $\theta \in [0, 1]$ that would not reject $H_0 : \theta = \hat{\theta}$ in favour of $H_1 : \theta \neq \hat{\theta}$ at confidence level α.
- Equivalent to the Bayesian interval with the uniform prior on θ removed.

Alternative #3 – Clopper-Pearson Exact Intervals

$$\mathrm{CI}_{1-\alpha,\mathrm{CP}}(\theta) = [\theta_{\text{lower}}, \theta_{\text{upper}}]$$

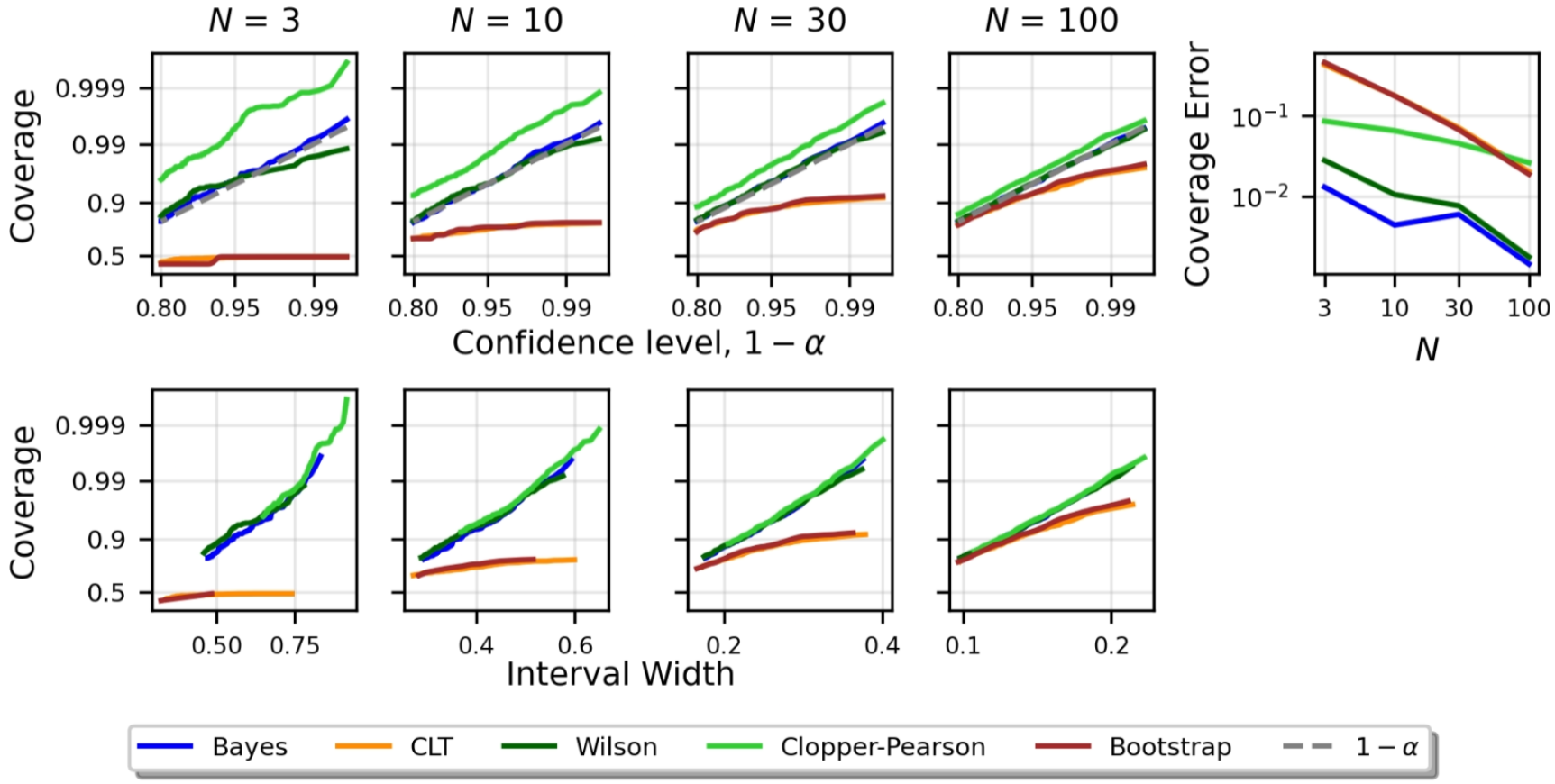
$$\theta_{\text{lower}} = B\left(\frac{\alpha}{2}, \sum_{i=1}^{N} y_i, 1 + \sum_{i=1}^{N}(1 - y_i)\right) \quad \text{and} \quad \theta_{\text{upper}} = B\left(1 - \frac{\alpha}{2}, 1 + \sum_{i=1}^{N} y_i, \sum_{i=1}^{N}(1 - y_i)\right)$$

where $B(\alpha, a, b)$ is the α-th quantile of the Beta(a, b) distribution.

- Guaranteed to never under-cover (very conservative method; 'worst-case' approach).
  - Contains all $\theta \in [0, 1]$ that would not reject $H_0 : \theta = \hat{\theta}$ in favour of $H_1 : \theta \neq \hat{\theta}$ at confidence level α.
- Equivalent to the Bayesian interval with the uniform prior on θ removed.

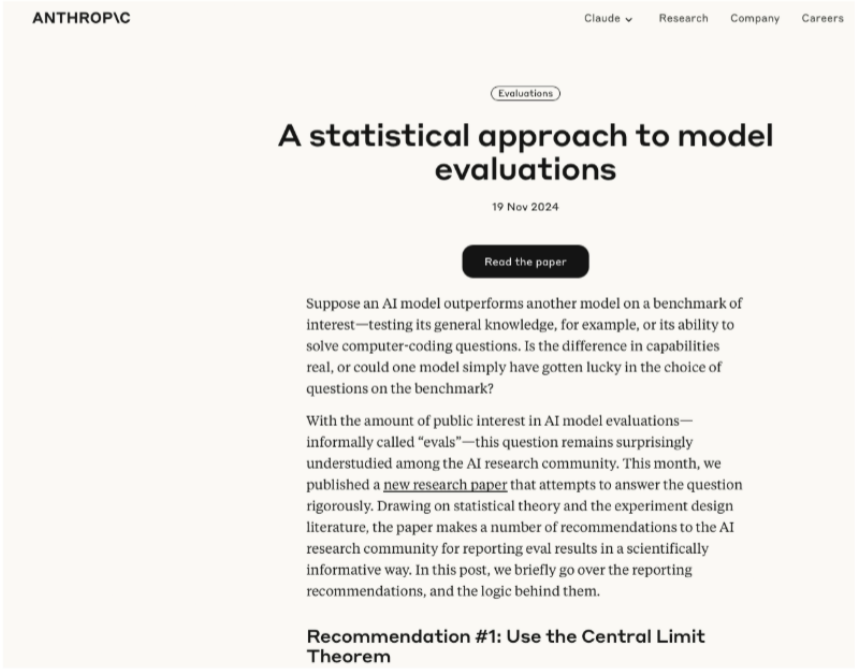
```
# y is a length N binary "eval" vector
S, N = y.sum(), len(y) # total successes & questions
result = scipy.stats.binomtest(k=S, n=N)
# 95% Clopper-Pearson exact interval
cp_ci = result.proportion_ci("exact", 0.95)
```

IID Questions Setting



Recommendation

Use Bayes or Wilson Score Intervals, not the CLT.



Other Eval Settings

Other Eval Settings

Clustered Questions

Other Eval Settings

Clustered Questions

Instead of N IID questions, we have T tasks, each with N_t IID questions.

Other Eval Settings

Clustered Questions

Instead of N IID questions, we have T tasks, each with N_t IID questions.

Independent Comparisons

Other Eval Settings

Clustered Questions

Instead of N IID questions, we have T tasks, each with N_t IID questions.

Independent Comparisons

Compare θ_A and θ_B for two different models, with access *only* to $N_A, N_B, \hat{\theta}_A$, and $\hat{\theta}_B$.

Other Eval Settings

Clustered Questions

Instead of N IID questions, we have T tasks, each with N_t IID questions.

Independent Comparisons

Compare θ_A and θ_B for two different models, with access *only* to $N_A, N_B, \hat{\theta}_A$, and $\hat{\theta}_B$.

Paired Comparisons

Other Eval Settings

Clustered Questions

Instead of N IID questions, we have T tasks, each with N_t IID questions.

Independent Comparisons

Compare θ_A and θ_B for two different models, with access *only* to $N_A, N_B, \hat{\theta}_A$, and $\hat{\theta}_B$.

Paired Comparisons

Compare θ_A and θ_B for two different models, each with the same N IID questions and access to question-level successes $\{y_{A;i}\}_{i=1}^N$ and $\{y_{B;i}\}_{i=1}^N$.

Other Eval Settings

Clustered Questions

Instead of N IID questions, we have T tasks, each with N_t IID questions.

Independent Comparisons

Compare θ_A and θ_B for two different models, with access *only* to $N_A, N_B, \hat{\theta}_A$, and $\hat{\theta}_B$.

Paired Comparisons

Compare θ_A and θ_B for two different models, each with the same N IID questions and access to question-level successes $\{y_{A;i}\}_{i=1}^{N}$ and $\{y_{B;i}\}_{i=1}^{N}$.

Metrics that aren't simple averages of binary results (e.g. F1 score).

Clustered Questions Setting

(Instead of N IID questions, we have T tasks, each with N_t IID questions.)

Clustered Questions Setting

(Instead of N IID questions, we have T tasks, each with N_t IID questions.)

Some tasks are easier than others. (E.g. each task asks K questions about a single piece of input text.)

Clustered Questions Setting

(Instead of N IID questions, we have T tasks, each with N_t IID questions.)

Some tasks are easier than others. (E.g. each task asks K questions about a single piece of input text.)

- 'Dispersion' parameter d controls the range of difficulty across tasks.

Clustered Questions Setting

(Instead of N IID questions, we have T tasks, each with N_t IID questions.)

Some tasks are easier than others. (E.g. each task asks K questions about a single piece of input text.)

- 'Dispersion' parameter d controls the range of difficulty across tasks.

$$d \sim \text{Gamma}(1, 1)$$

Clustered Questions Setting

(Instead of N IID questions, we have T tasks, each with N_t IID questions.)

Some tasks are easier than others. (E.g. each task asks K questions about a single piece of input text.)

- 'Dispersion' parameter d controls the range of difficulty across tasks.

$$d \sim \text{Gamma}(1, 1)$$

- θ is the mean difficulty of the questions across tasks.

Clustered Questions Setting

(Instead of N IID questions, we have T tasks, each with N_t IID questions.)

Some tasks are easier than others. (E.g. each task asks K questions about a single piece of input text.)

- 'Dispersion' parameter d controls the range of difficulty across tasks.

$$d \sim \text{Gamma}(1, 1)$$

- θ is the mean difficulty of the questions across tasks.

$$\theta \sim \text{Beta}(1, 1) = \text{Uniform}[0, 1]$$

Clustered Questions Setting

(Instead of N IID questions, we have T tasks, each with N_t IID questions.)

Some tasks are easier than others. (E.g. each task asks K questions about a single piece of input text.)

- 'Dispersion' parameter d controls the range of difficulty across tasks.

$$d \sim \text{Gamma}(1, 1)$$

- θ is the mean difficulty of the questions across tasks.

$$\theta \sim \text{Beta}(1, 1) = \text{Uniform}[0, 1]$$

- θ_t is the difficulty of the questions in task t (we ensure that $\mathbb{E}[\theta_t] = \theta$).

Clustered Questions Setting

(Instead of N IID questions, we have T tasks, each with N_t IID questions.)

Some tasks are easier than others. (E.g. each task asks K questions about a single piece of input text.)

- 'Dispersion' parameter d controls the range of difficulty across tasks.

$$d \sim \text{Gamma}(1, 1)$$

- θ is the mean difficulty of the questions across tasks.

$$\theta \sim \text{Beta}(1, 1) = \text{Uniform}[0, 1]$$

- θ_t is the difficulty of the questions in task t (we ensure that $\mathbb{E}[\theta_t] = \theta$).

$$\theta_t \sim \text{Beta}(d\theta, d(1-\theta))$$

Clustered Questions Setting

(Instead of N IID questions, we have T tasks, each with N_t IID questions.)

Some tasks are easier than others. (E.g. each task asks K questions about a single piece of input text.)

- 'Dispersion' parameter d controls the range of difficulty across tasks.

$$d \sim \text{Gamma}(1, 1)$$

- θ is the mean difficulty of the questions across tasks.

$$\theta \sim \text{Beta}(1, 1) = \text{Uniform}[0, 1]$$

- θ_t is the difficulty of the questions in task t (we ensure that $\mathbb{E}[\theta_t] = \theta$).

$$\theta_t \sim \text{Beta}(d\theta, d(1-\theta))$$

- If d is small, then θ_t is close to θ for all tasks.
- If d is large, then θ_t is more variable across tasks.

Clustered Questions Setting

(Instead of N IID questions, we have T tasks, each with N_t IID questions.)

Generative Model

Clustered Questions Setting

(Instead of N IID questions, we have T tasks, each with N_t IID questions.)

Generative Model

- 'Dispersion' parameter d controls the range of difficulty of the questions, whilst maintaining $\mathbb{E}[\theta_t] = \theta$.

Clustered Questions Setting

(Instead of N IID questions, we have T tasks, each with N_t IID questions.)

Generative Model

- 'Dispersion' parameter d controls the range of difficulty of the questions, whilst maintaining $\mathbb{E}[\theta_t] = \theta$.

$$d \sim \text{Gamma}(1,1), \quad \theta \sim \text{Beta}(1,1), \quad \theta_t \quad \sim \text{Beta}(d\theta, d(1-\theta)), \quad y_{i,t} \sim \text{Bernoulli}(\theta_t)$$

Clustered Questions Setting

(Instead of N IID questions, we have T tasks, each with N_t IID questions.)

Generative Model

- 'Dispersion' parameter d controls the range of difficulty of the questions, whilst maintaining $\mathbb{E}[\theta_t] = \theta$.

$$d \sim \text{Gamma}(1,1), \quad \theta \sim \text{Beta}(1,1), \quad \theta_t \sim \text{Beta}(d\theta, d(1-\theta)), \quad y_{i,t} \sim \text{Bernoulli}(\theta_t)$$

Bayesian Inference

- Can we integrate out the per-task difficulty parameters θ_t?

Clustered Questions Setting

(Instead of N IID questions, we have T tasks, each with N_t IID questions.)

Generative Model

- 'Dispersion' parameter d controls the range of difficulty of the questions, whilst maintaining $\mathbb{E}[\theta_t] = \theta$.

$$d \sim \text{Gamma}(1,1), \quad \theta \sim \text{Beta}(1,1), \quad \theta_t \sim \text{Beta}(d\theta, d(1-\theta)), \quad y_{i,t} \sim \text{Bernoulli}(\theta_t)$$

Bayesian Inference

- Can we integrate out the per-task difficulty parameters θ_t?

- Yes! The number of successes per task is Beta-Binomial distributed:

$$\sum_{i=1}^{N_t} y_{i,t} = Y_t \sim \text{BetaBinomial}(N_t, d\theta, d(1-\theta))$$

Clustered Questions Setting

(Instead of N IID questions, we have T tasks, each with N_t IID questions.)

Generative Model

- 'Dispersion' parameter d controls the range of difficulty of the questions, whilst maintaining $\mathbb{E}[\theta_t] = \theta$.

$$d \sim \text{Gamma}(1,1), \quad \theta \sim \text{Beta}(1,1), \quad \theta_t \sim \text{Beta}(d\theta, d(1-\theta)), \quad y_{i,t} \sim \text{Bernoulli}(\theta_t)$$

Bayesian Inference

- Can we integrate out the per-task difficulty parameters θ_t?

- Yes! The number of successes per task is Beta-Binomial distributed:

$$\sum_{i=1}^{N_t} y_{i,t} = Y_t \sim \text{BetaBinomial}(N_t, d\theta, d(1-\theta))$$

Get an importance-weighted posterior for θ: draw prior samples $\{(\theta^{(k)}, d^{(k)})\}_{k=1}^{K}$, then compute weights

$$w^{(k)} = \prod_{t=1}^{T} p(Y_t | \theta^{(k)}, d^{(k)})$$

Clustered Questions Setting

(Instead of N IID questions, we have T tasks, each with N_t IID questions.)

Generative Model

- 'Dispersion' parameter d controls the range of difficulty of the questions, whilst maintaining $\mathbb{E}[\theta_t] = \theta$.

$$d \sim \text{Gamma}(1,1), \quad \theta \sim \text{Beta}(1,1), \quad \theta_t \sim \text{Beta}(d\theta, d(1-\theta)), \quad y_{i,t} \sim \text{Bernoulli}(\theta_t)$$

Clustered Standard Error (CLT-based Approach)

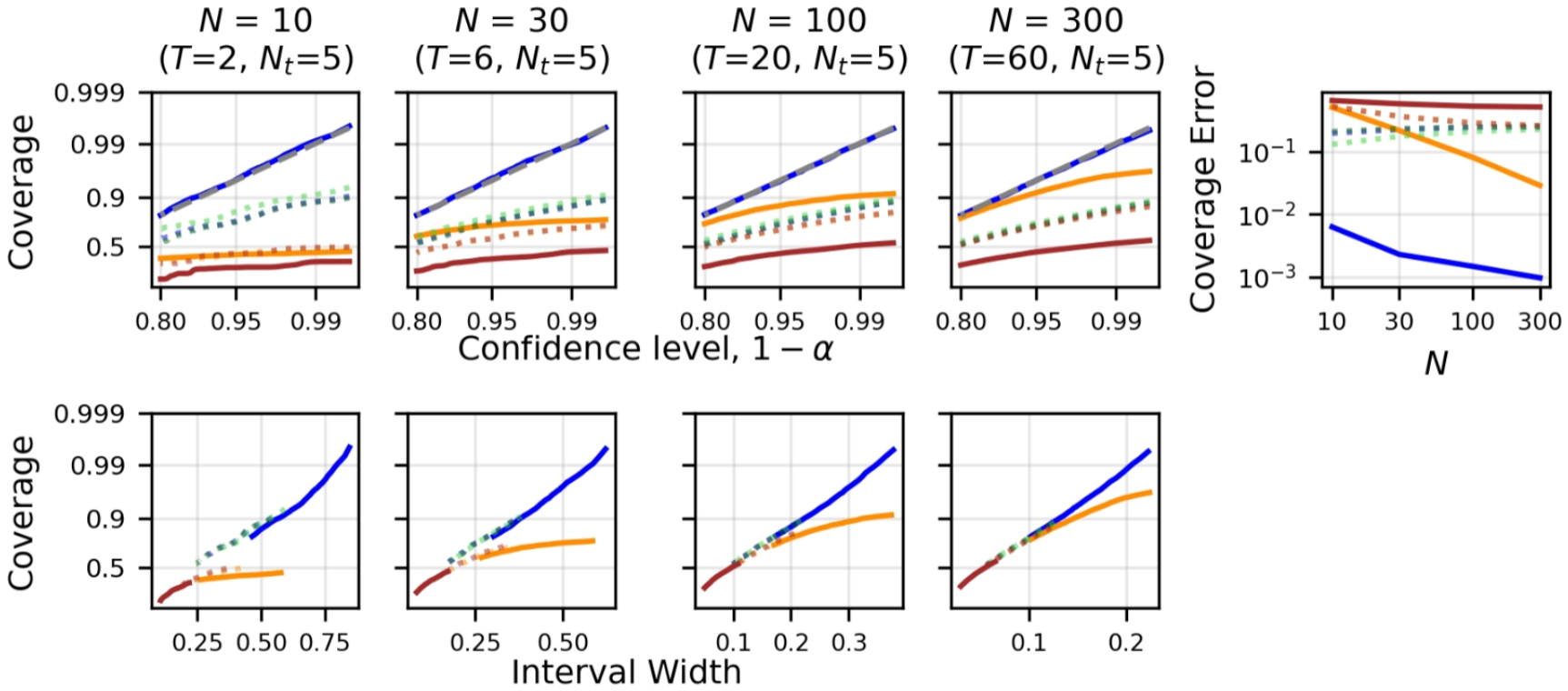
Update the standard error to account for the clustering:

$$\text{SE}_{\text{clust.}} = \sqrt{\text{SE}_{\text{CLT}}^2 + \frac{1}{N^2} \sum_{t=1}^{T} \sum_{i=1}^{N_t} \sum_{j \neq i} (y_{i,t} - \bar{y})(y_{j,t} - \bar{y})}$$

$$\text{CI}_{1-\alpha}(\theta) = \hat{\theta} \pm z_{\alpha/2} \text{SE}_{\text{clust.}}$$

Clustered Questions Setting

(Dotted lines are IID-assuming methods, solid lines are clustered methods.)



Model Comparison (Unpaired)

Compare θ_A and θ_B for two different models, with access *only* to $N = N_A = N_B, \hat{\theta}_A$, and $\hat{\theta}_B$.

Model Comparison (Unpaired)

Compare θ_A and θ_B for two different models, with access *only* to $N = N_A = N_B, \hat{\theta}_A$, and $\hat{\theta}_B$.

- Compute an interval over the difference $\theta_A - \theta_B$ check if it's positive.

Model Comparison (Unpaired)

Compare θ_A and θ_B for two different models, with access *only* to $N = N_A = N_B, \hat{\theta}_A$, and $\hat{\theta}_B$.

- Compute an interval over the difference $\theta_A - \theta_B$ check if it's positive.
- Compute an interval over the odds ratio $\frac{\theta_A/(1-\theta_A)}{\theta_B/(1-\theta_B)}$, check if it's greater than 1.

Model Comparison (Unpaired)

Compare θ_A and θ_B for two different models, with access *only* to $N = N_A = N_B, \hat{\theta}_A$, and $\hat{\theta}_B$.

- Compute an interval over the difference $\theta_A - \theta_B$ check if it's positive.
- Compute an interval over the odds ratio $\frac{\theta_A/(1-\theta_A)}{\theta_B/(1-\theta_B)}$, check if it's greater than 1.

Bayesian Approach

Obtain a posterior for model A and a posterior for model B, using the earlier Beta-Binomial model.

```
# y_A and y_B are vectors of evals for two models
import numpy as np

S_A, S_B = y_A.sum(), y_B.sum()
# draw posterior samples (ps)
ps_A = np.random.beta(1 + S_A, 1 + (N - S_A), size=2000)
ps_B = np.random.beta(1 + S_B, 1 + (N - S_B), size=2000)
# posterior difference and 95% QBI
ps_diff = ps_A - ps_B
bayes_diff = np.percentile(ps_diff, [2.5, 97.5])
# posterior odds ratio and 95% QBI
ps_or = (ps_A / (1 - ps_A)) / (ps_B / (1 - ps_B))
bayes_or = np.percentile(ps_or, [2.5, 97.5])
```

Model Comparison (Unpaired)

Compare θ_A and θ_B for two different models, with access *only* to $N = N_A = N_B, \hat{\theta}_A$, and $\hat{\theta}_B$.

- Compute an interval over the difference $\theta_A - \theta_B$ check if it's positive.
- Compute an interval over the odds ratio $\frac{\theta_A/(1-\theta_A)}{\theta_B/(1-\theta_B)}$, check if it's greater than 1.

Frequentist Approach

- Use the CLT for the **difference** and add A and B's squared standard errors:

$$\text{CI}_{1-\alpha}(\theta_A - \theta_B) = (\hat{\theta}_A - \hat{\theta}_B) \pm z_{\alpha/2} \sqrt{S_A^2/N_A + S_B^2/N_B}.$$

Model Comparison (Unpaired)

Compare θ_A and θ_B for two different models, with access *only* to $N = N_A = N_B, \hat{\theta}_A$, and $\hat{\theta}_B$.

- Compute an interval over the difference $\theta_A - \theta_B$ check if it's positive.
- Compute an interval over the odds ratio $\frac{\theta_A/(1-\theta_A)}{\theta_B/(1-\theta_B)}$, check if it's greater than 1.

Frequentist Approach

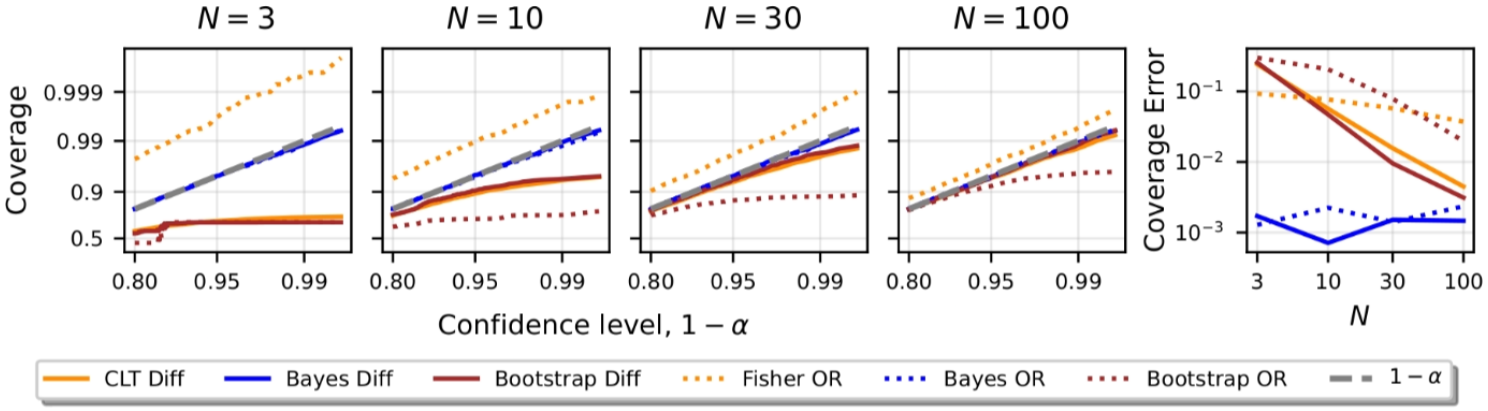
- Use the CLT for the **difference** and add A and B's squared standard errors:

$$\mathrm{CI}_{1-\alpha}(\theta_A - \theta_B) = (\hat{\theta}_A - \hat{\theta}_B) \pm z_{\alpha/2} \sqrt{S_A^2/N_A + S_B^2/N_B}.$$

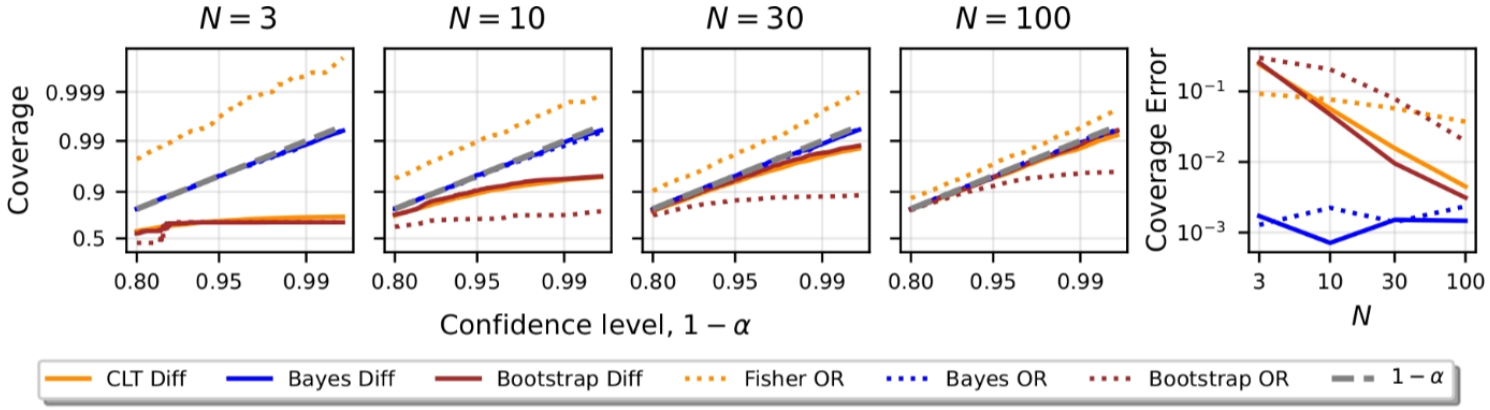
- Use an inverted Fisher's Exact Test for the **odds ratio** (this will never under-cover):
  - Equivalent to Bayesian intervals with strong priors: $\theta_A \sim \mathrm{Beta}(1, 0)$ and $\theta_B \sim \mathrm{Beta}(0, 1)$.

```
# y_A and y_B are vectors of evals for two models
from scipy.stats.contingency import odds_ratio
S_A, S_B = y_A.sum(), y_B.sum()
result = odds_ratio([[S_A, N_A - S_A], [S_B, N_B - S_B]])
ci_or = result.confidence_interval(confidence_level=0.95, alternative='two-sided')
```

Model Comparison (Unpaired)



Model Comparison (Unpaired)



Bayesian Bonus: we can easily compute probabilities of one model being better than the other:

$$\mathbb{P}(\theta_A > \theta_B | y_{A;1:N}, y_{B;1:N}) = \frac{1}{K} \sum_{k=1}^{K} 1[\theta_A^{(k)} > \theta_B^{(k)}].$$

where $\theta_m^{(k)} \sim p(\theta_m | y_{m,1:N})$ are posterior samples for models $m \in \{A, B\}$.

Model Comparison (Paired)

Compute intervals over the difference $\theta_A - \theta_B$, where we have access to the same N (IID) questions for both models: $\{y_{A;i}\}_{i=1}^N$ and $\{y_{B;i}\}_{i=1}^N$.

Model Comparison (Paired)

Compute intervals over the difference $\theta_A - \theta_B$, where we have access to the same N (IID) questions for both models: $\{y_{A;i}\}_{i=1}^N$ and $\{y_{B;i}\}_{i=1}^N$.

Frequentist Approach

Use the CLT directly for the difference $D_i = y_{A;i} - y_{B;i}$:

$$\hat{\theta}_D = \frac{1}{N}\sum_{i=1}^{N} D_i,$$

$$\mathrm{CI}_{1-\alpha}(\theta_A - \theta_B) = \hat{\theta}_D \pm z_{\alpha/2}\,\mathrm{SE}(\hat{\theta}_D).$$

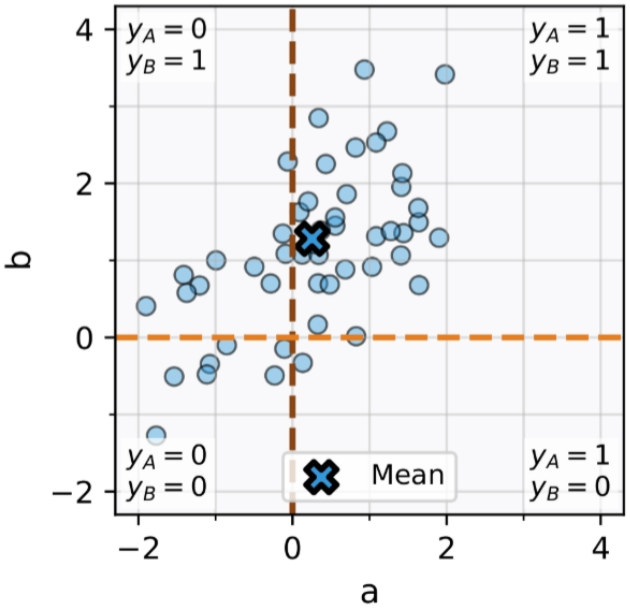
Model Comparison (Paired)

Compute intervals over the difference $\theta_A - \theta_B$, where we have access to the same N (IID) questions for both models: $\{y_{A;i}\}_{i=1}^N$ and $\{y_{B;i}\}_{i=1}^N$.

Bayesian Approach (Importance Sampling)

$$
\begin{aligned}
\theta_A, \theta_B &\sim \text{Beta}(1,1) = \text{Uniform}[0,1], \\
\hat{\rho} &\sim \text{Beta}(4,2), \\
\rho &= 2\hat{\rho} - 1, \\
(a_i, b_i) &\sim \mathcal{N}\left(\begin{pmatrix}\Phi^{-1}(\theta_A) \\ \Phi^{-1}(\theta_B)\end{pmatrix}, \begin{pmatrix}1 & \rho \\ \rho & 1\end{pmatrix}\right), \\
y_{A;i} &= 1[a_i > 0], \\
y_{B;i} &= 1[b_i > 0].
\end{aligned}
$$

where Φ is the standard normal CDF.



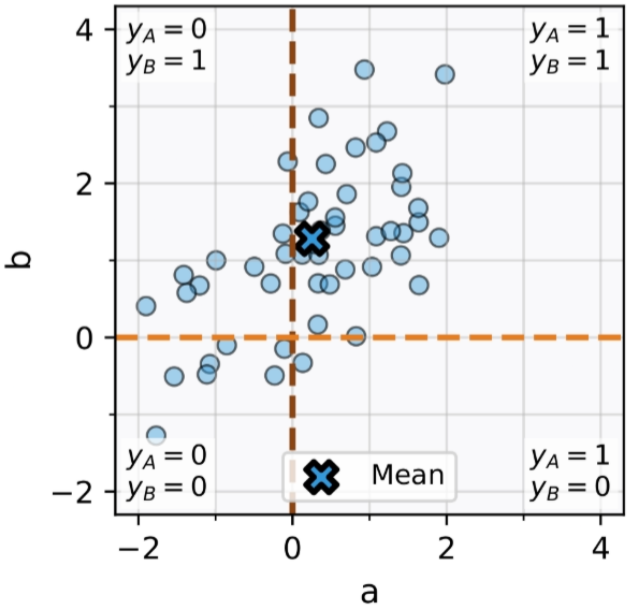
Model Comparison (Paired)

Compute intervals over the difference $\theta_A - \theta_B$, where we have access to the same N (IID) questions for both models: $\{y_{A;i}\}_{i=1}^N$ and $\{y_{B;i}\}_{i=1}^N$.

Bayesian Approach (Importance Sampling)

$$\theta_A, \theta_B \sim \text{Beta}(1,1) = \text{Uniform}[0,1],$$
$$\hat{\rho} \sim \text{Beta}(4,2),$$
$$\rho = 2\hat{\rho} - 1,$$
$$(a_i, b_i) \sim \mathcal{N}\left(\begin{pmatrix}\Phi^{-1}(\theta_A)\\ \Phi^{-1}(\theta_B)\end{pmatrix}, \begin{pmatrix}1 & \rho\\ \rho & 1\end{pmatrix}\right),$$
$$y_{A;i} = 1[a_i > 0],$$
$$y_{B;i} = 1[b_i > 0].$$

where Φ is the standard normal CDF.



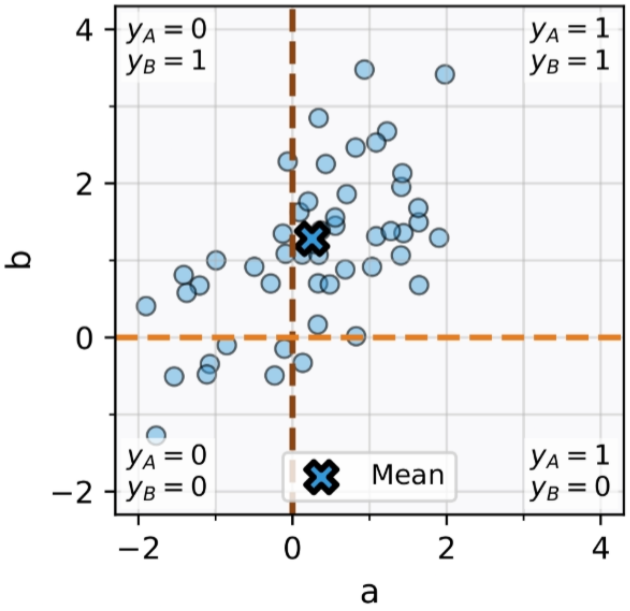
Model Comparison (Paired)

Compute intervals over the difference $\theta_A - \theta_B$, where we have access to the same N (IID) questions for both models: $\{y_{A;i}\}_{i=1}^N$ and $\{y_{B;i}\}_{i=1}^N$.

Bayesian Approach (Importance Sampling)

$$
\begin{aligned}
\theta_A, \theta_B &\sim \text{Beta}(1,1) = \text{Uniform}[0,1], \\
\hat{\rho} &\sim \text{Beta}(4,2), \\
\rho &= 2\hat{\rho} - 1, \\
(a_i, b_i) &\sim \mathcal{N}\left(\begin{pmatrix} \Phi^{-1}(\theta_A) \\ \Phi^{-1}(\theta_B) \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right), \\
y_{A;i} &= 1[a_i > 0], \\
y_{B;i} &= 1[b_i > 0].
\end{aligned}
$$

where Φ is the standard normal CDF.



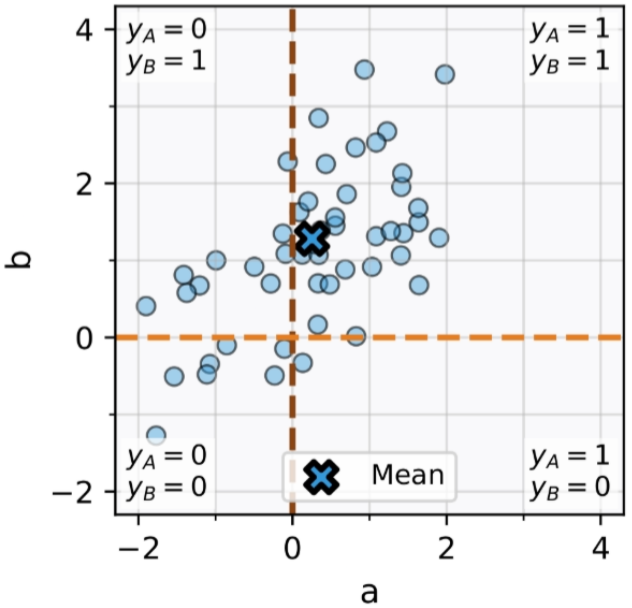
Model Comparison (Paired)

Compute intervals over the difference $\theta_A - \theta_B$, where we have access to the same N (IID) questions for both models: $\{y_{A;i}\}_{i=1}^N$ and $\{y_{B;i}\}_{i=1}^N$.

Bayesian Approach (Importance Sampling)

$$\theta_A, \theta_B \sim \text{Beta}(1,1) = \text{Uniform}[0,1],$$
$$\hat{\rho} \sim \text{Beta}(4,2),$$
$$\rho = 2\hat{\rho} - 1,$$
$$(a_i, b_i) \sim \mathcal{N}\left(\begin{pmatrix}\Phi^{-1}(\theta_A)\\ \Phi^{-1}(\theta_B)\end{pmatrix}, \begin{pmatrix}1 & \rho\\ \rho & 1\end{pmatrix}\right),$$
$$y_{A;i} = 1[a_i > 0],$$
$$y_{B;i} = 1[b_i > 0].$$

where Φ is the standard normal CDF.



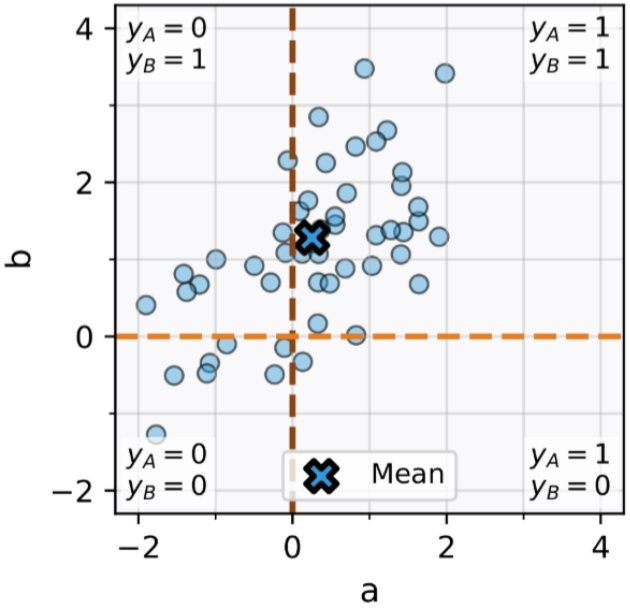
Model Comparison (Paired)

Compute intervals over the difference $\theta_A - \theta_B$, where we have access to the same N (IID) questions for both models: $\{y_{A;i}\}_{i=1}^N$ and $\{y_{B;i}\}_{i=1}^N$.

Bayesian Approach (Importance Sampling)

$$
\begin{aligned}
\theta_A, \theta_B &\sim \text{Beta}(1,1) = \text{Uniform}[0,1], \\
\hat{\rho} &\sim \text{Beta}(4,2), \\
\rho &= 2\hat{\rho} - 1, \\
(a_i, b_i) &\sim \mathcal{N}\left(\begin{pmatrix} \Phi^{-1}(\theta_A) \\ \Phi^{-1}(\theta_B) \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right), \\
y_{A;i} &= 1[a_i > 0], \\
y_{B;i} &= 1[b_i > 0].
\end{aligned}
$$

where Φ is the standard normal CDF.

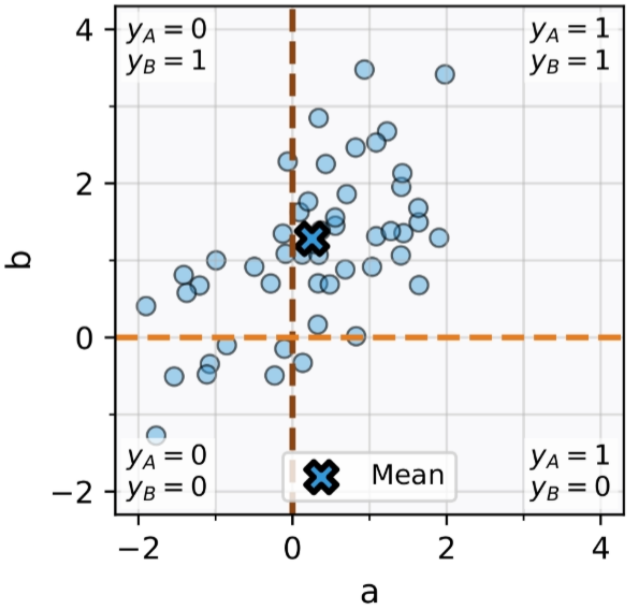


Model Comparison (Paired)

Compute intervals over the difference $\theta_A - \theta_B$, where we have access to the same N (IID) questions for both models: $\{y_{A;i}\}_{i=1}^N$ and $\{y_{B;i}\}_{i=1}^N$.

Bayesian Approach (Importance Sampling)

$$
\begin{aligned}
\theta_A, \theta_B &\sim \text{Beta}(1,1) = \text{Uniform}[0,1], \\
\hat{\rho} &\sim \text{Beta}(4,2), \\
\rho &= 2\hat{\rho} - 1, \\
(a_i, b_i) &\sim \mathcal{N}\left(\begin{pmatrix} \Phi^{-1}(\theta_A) \\ \Phi^{-1}(\theta_B) \end{pmatrix}, \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix} \right), \\
y_{A;i} &= 1[a_i > 0], \\
y_{B;i} &= 1[b_i > 0].
\end{aligned}
$$

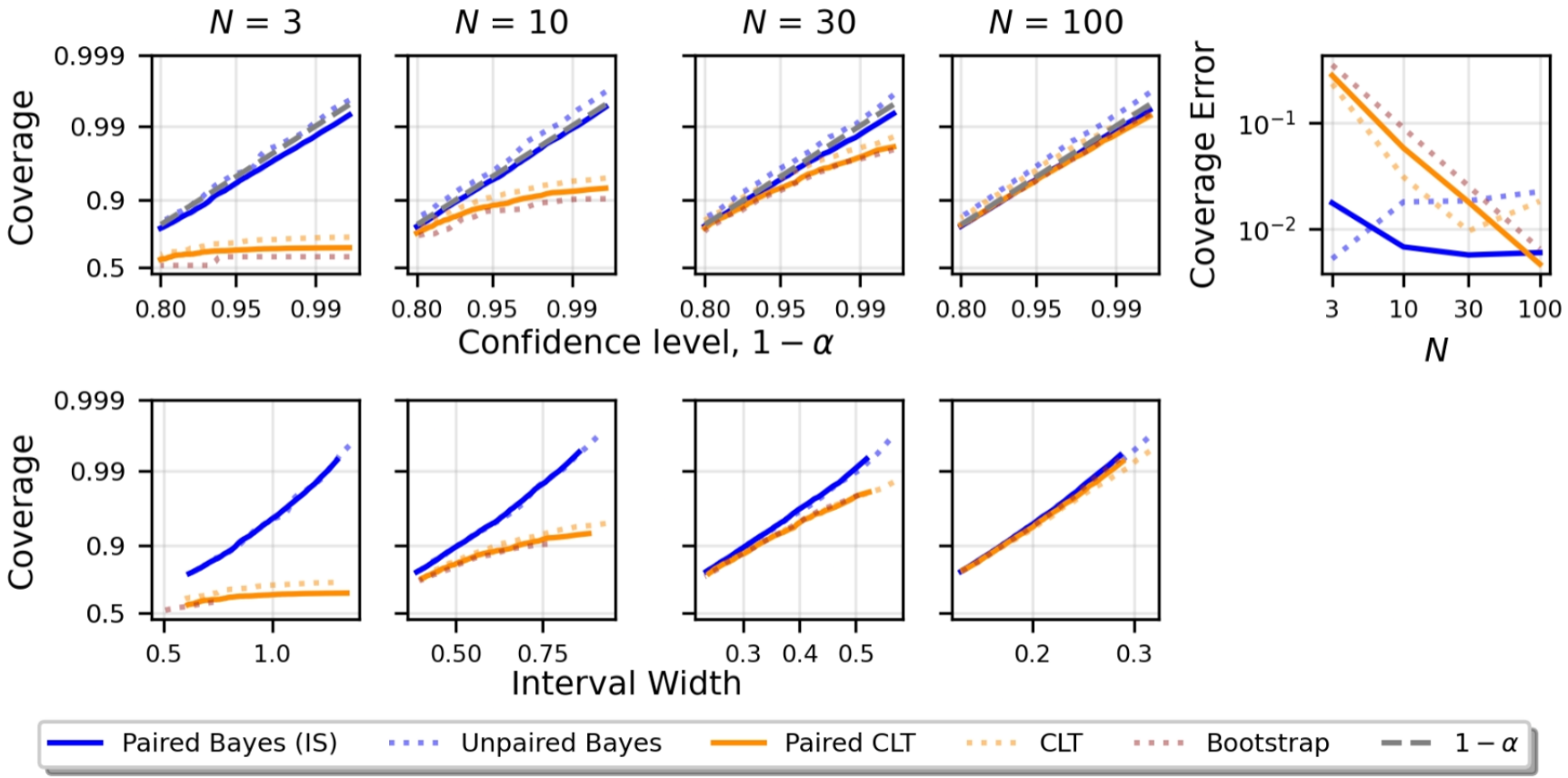


where Φ is the standard normal CDF.

Ensures $y_{A;i} \sim \text{Ber}(\theta_A)$ and $y_{B;i} \sim \text{Ber}(\theta_B)$, whilst still allowing for correlation between the two models.

(Dotted lines are unpaired methods, solid lines are paired methods.)

Model Comparison (Paired)



Prior Mismatch

Prior Mismatch

- By default, we avoid using informative/subjective priors and stick to $\theta \sim \text{Uniform}[0, 1]$.

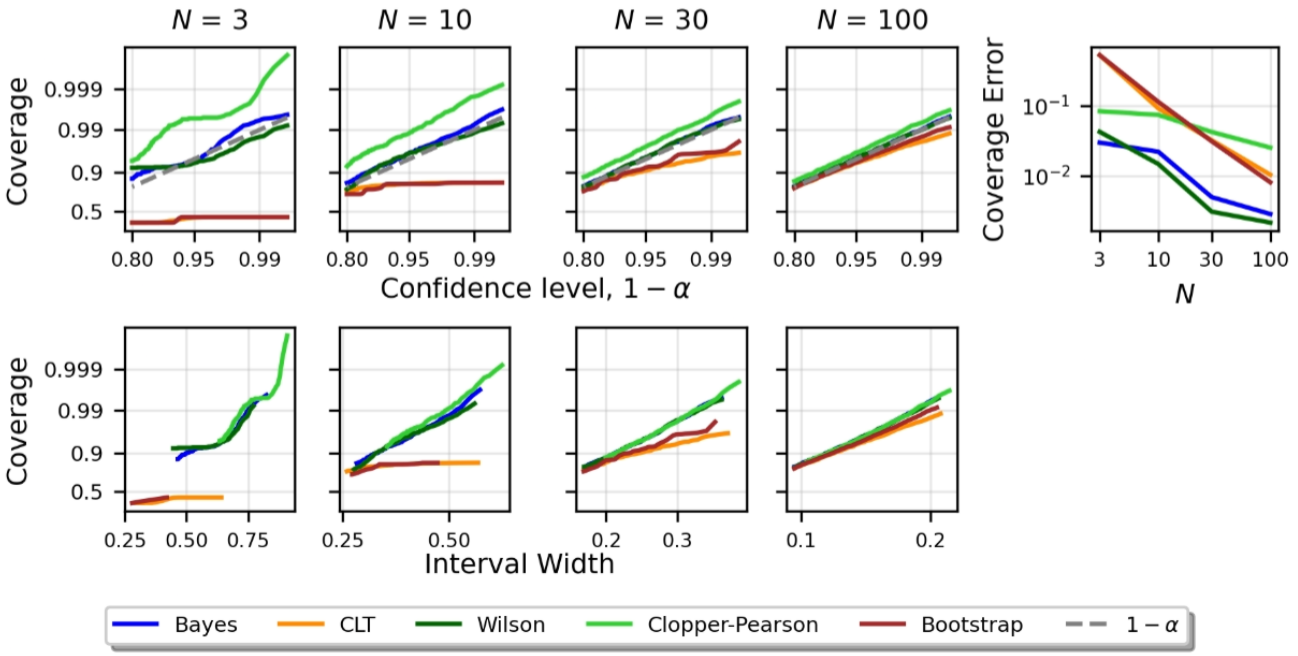
Prior Mismatch

- By default, we avoid using informative/subjective priors and stick to $\theta \sim \text{Uniform}[0, 1]$.
- Bayesian methods still generally outperform CLT-based approaches when the underlying prior is different.

Prior Mismatch

- By default, we avoid using informative/subjective priors and stick to $\theta \sim \text{Uniform}[0, 1]$.
- Bayesian methods still generally outperform CLT-based approaches when the underlying prior is different.

$$\text{e.g.} \quad \text{Beta}(100, 20), \quad \mathbb{E}[\theta] = 0.83, \quad \text{Var}[\theta] = 0.034^2$$



Conclusion

Conclusion

- Use Bayes (or Wilson), it's not hard (`scipy` or `bayes_evals`), it's safer, and it's still cheap for large N.

Conclusion

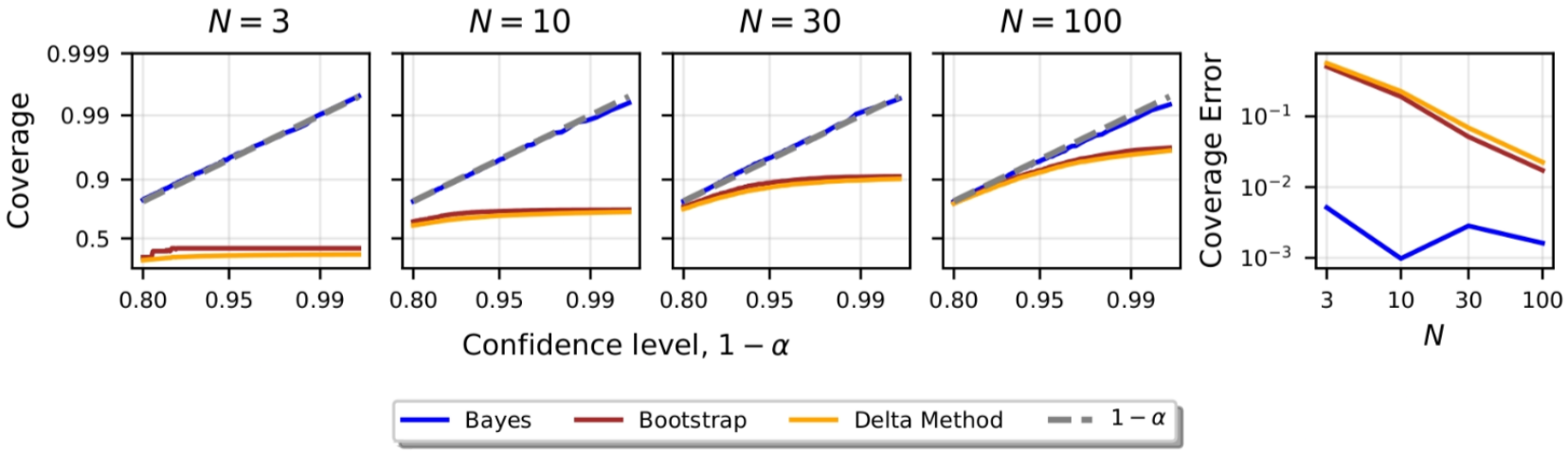
- Use Bayes (or Wilson), it's not hard (`scipy` or `bayes_evals`), it's safer, and it's still cheap for large N.
- Plus you get the flexibility of Bayes!

Conclusion

- Use Bayes (or Wilson), it's not hard (`scipy` or `bayes_evals`), it's safer, and it's still cheap for large N.
- Plus you get the flexibility of Bayes!
  - Computing probabilities $\mathbb{P}(\theta_A > \theta_B)$.

Conclusion

- Use Bayes (or Wilson), it's not hard (`scipy` or `bayes_evals`), it's safer, and it's still cheap for large N.
- Plus you get the flexibility of Bayes!
  - Computing probabilities $\mathbb{P}(\theta_A > \theta_B)$.
  - Intervals on nonlinear functions of sample means e.g. F1 score (harmonic mean of precision and recall).



Questions?

Paper

https://arxiv.org/pdf/2503.01747

bayes_evals package

https://github.com/sambowyer/bayes_evals

Summary Table

Table 1: **Overview of methods**. *Coverage* describes whether the method provides the desired nominal coverage in small-sample settings. *Efficiency* describes how tight (and precise) the resulting confidence/credible intervals are given the nominal coverage (e.g., CLT-based intervals can be invalid or too wide). Although the *computational cost* of these methods is negligible compared to the cost of evaluating LLMs, we indicate their relative costs for comparison among the methods.

	Coverage small N	Efficiency small N	Computational cost	Easy to implement
CLT	✗	✗	Very low	Yes
CLT-based variants (e.g. Delta method)	✗	✗	Very low	Moderate
Custom frequentist (e.g. Wilson)	✓	✓	Very low	Moderate
Bootstrap	✗	✗	Low	Moderate
Bayes (conjugate)	✓	✓	Very low	Yes
Bayes (importance sampling)	✓	✓	Low	Moderate

Appendix – Clustered Importance Sampling Code

```
S_t, N_t: np.arrays of length T with total successes & questions per task
# set number of samples, K
K = 10_000

# get K samples from the prior (with extra dimension for broadcasting over tasks)
thetas = np.random.beta(1,1, size=(K,1))
ds = np.random.gamma(1,1, size=(K,1))

# obtain weights via the likelihood (sum the per-task log-probs)
log_weights = scipy.stats.betabinom(N_t, (ds*thetas), (ds*(1-thetas))).logpmf(S_t).sum(-1)

# normalise the weights
weights = np.exp(log_weights - log_weights.max())
weights /= weights.sum()

# obtain samples from the posterior
posterior = thetas[np.random.choice(K, size=K, replace=True, p=weights)]

# Bayesian credible interval
bayes_ci = np.percentile(posterior, [2.5, 97.5])
```

Appendix – Paired Importance Sampling Code

```
# y_A, y_B: length N binary "eval" vectors
from binorm import binorm_cdf # 2D Gaussian CDF, defined elsewhere
K = 10_000
# get K samples from the prior
theta_As, theta_Bs, rhos = np.random.beta(1,1, size=K), np.random.beta(1,1,size=K), 2*np.random.beta(4,2, size=K) - 1
# 2x2 contingency table (flattened)
S = (y_A * y_B).sum(-1)            # S = A correct,   B correct
T = (y_A * (1 - y_B)).sum(-1)      # T = A correct,   B incorrect
U = ((1 - y_A) * y_B).sum(-1)      # U = A incorrect, B correct
V = ((1 - y_A) * (1 - y_B)).sum(-1) # V = A incorrect, B incorrect
# calculate the bivariate normal mean
mu_As, mu_Bs = scipy.stats.norm(0,1).ppf(theta_As), scipy.stats.norm(0,1).ppf(theta_Bs)
# Calculate probabilities of each cell in the 2x2 table
theta_V = binorm_cdf(x1=0, x2=0, mu1=mu_As, mu2=mu_Bs, sigma1=1, sigma2=1, rho=rhos)
theta_S = theta_As + theta_Bs + theta_V - 1
theta_T = 1 - theta_Bs - theta_V
theta_U = 1 - theta_As - theta_V
# (probabilities may be very small and negative instead of 0)
valid_idx = (theta_S > 0) & (theta_T > 0) & (theta_U > 0) & (theta_V > 0)
log_weights = S*np.log(theta_S[valid_idx]) + T*np.log(theta_T[valid_idx]) + \
              U*np.log(theta_U[valid_idx]) + V*np.log(theta_V[valid_idx])
# normalise the weights and obtain samples from the posterior
weights = np.zeros(K)
weights[valid_idx] = np.exp(log_weights - log_weights.max())
posterior = (theta_As - theta_Bs)[np.random.choice(K, size=K, replace=True, p=weights/weights.sum())]
bayes_ci = np.percentile(posterior, [2.5, 97.5])
```