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

Sam Bowyer, Laurence Aitchison, and Desi R. Ivanova

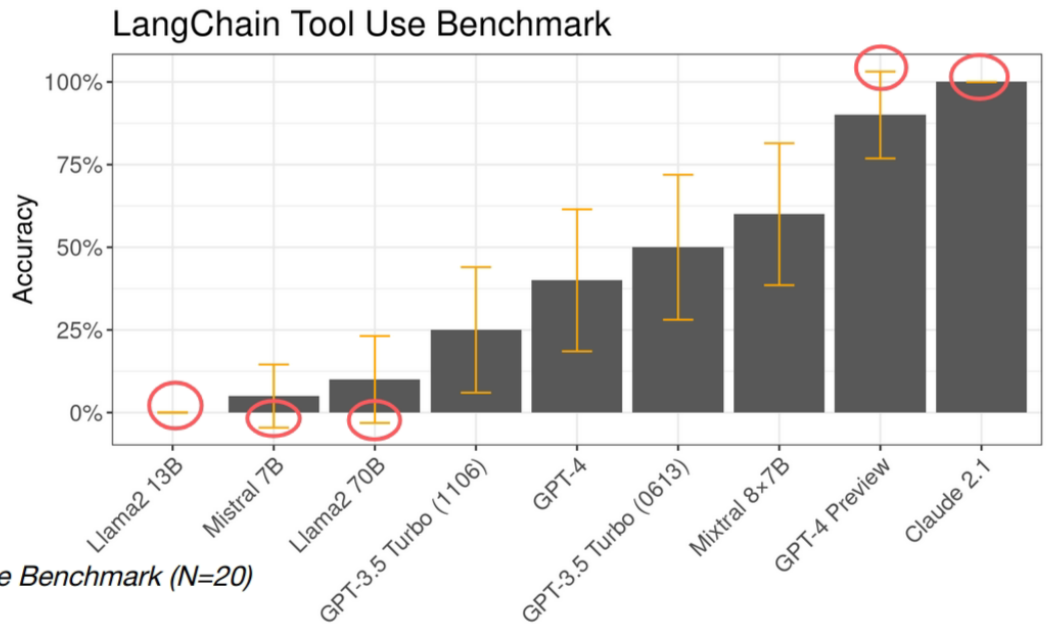
ICML 2025 Spotlight Position Paper

Motivation

CLT-based CI at confidence level $1-\alpha$ for binary data $X_i \sim \text{Bernoulli}(\theta)$:

$$\text{CI}_{1-\alpha}(\theta) = \bar{X} \pm z_{\alpha/2}\sqrt{\frac{\bar{X}(1-\bar{X})}{N}}$$

- Error bars are important for interpreting evals.
- The CLT is the most common method for computing error bars, but it's often unwise.
- Error bars can collapse to zero-width or extend past $[0, 1]$.



Langchain Typewriter Tool Use Benchmark (N=20)

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.

- The CLT relies on a large N assumption.
- As LLM capabilitites improve, constructing and running benchmarks is becoming more time-intensive.
- Researchers are increasingly using benchmarks with smaller N.
- We need alternative methods for computing error bars that work for both large *and* small N.

Alternative #1 – Beta-Binomial Model

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

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

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

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

Obtain quantile-based Bayesian *credible intervals* for θ from the closed form posterior (with confidence level $1 - \alpha$).

```
# 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 Alternatives

- Wilson score interval
  - Based on the normal approximation to the binomial distribution (but **not** the CLT).
- Clopper-Pearson exact interval
  - 'Worst-case' approach (very conservative method; guaranteed to never under-cover).

```
# 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 and Clopper-Pearson exact interval
wilson_ci = result.proportion_ci("wilson", 0.95)
cp_ci = result.proportion_ci("exact", 0.95)
```

Interval Comparison

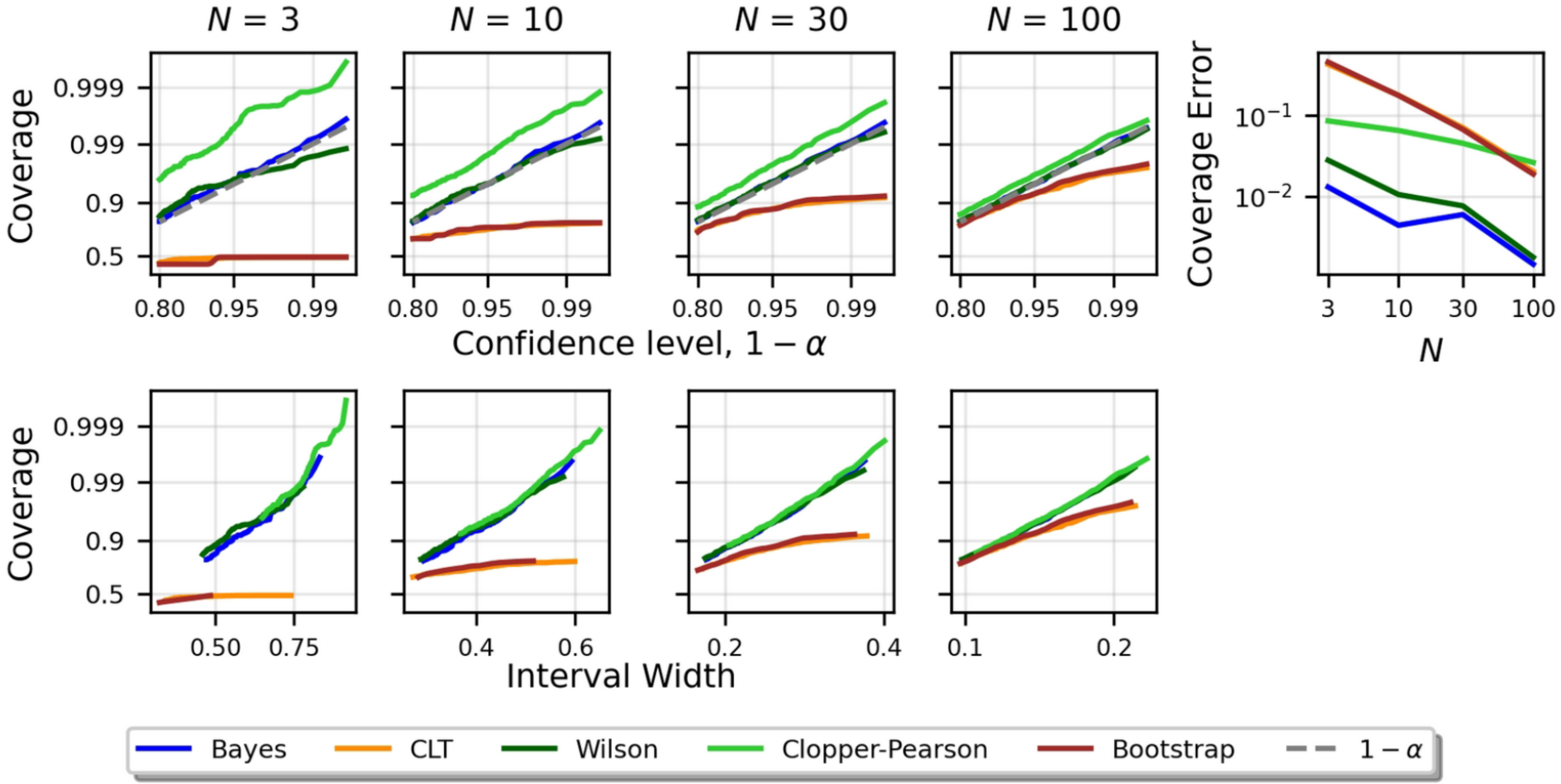
We'll focus on two metrics for evaluating intervals:

- 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. $1 - \alpha$).
- Width
  - Ideally, our intervals would be as tight as possible.

We have to rely on synthetic eval 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 intervals with various $1 - \alpha$ confidence levels.
- Repeat many times and calculate the true coverage and width of the intervals.

IID Questions Setting



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 $N_A, N_B, \bar{y}_A$, and $\bar{y}_B$.

Paired Comparisons

Compare θ_A and θ_B for two different models, each with <u>the same</u> 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).

Conclusion

Use Bayes or Wilson Score intervals.

- It's not hard (use `scipy` or `bayes_evals`).
- It's safer than CLT-based methods.
- It's still cheap for large N.

Paper

https://arxiv.org/pdf/2503.01747

bayes_evals package

https://github.com/sambowyer/bayes_evals