Cohere Research Talk: Massively Parallel Inference & Bayesian Evals

Sam Bowyer

November 2025

Outline

1. About Me
2. Alan: Massively Parallel Probabilistic Programming
3. Bayesian Evals: Uncertainty Quantification for LLM Evals

About Me

- Fourth (final) year PhD student at Bristol's Compass CDT (Computational Statistics and Data Science).

About Me

- Fourth (final) year PhD student at Bristol's Compass CDT (Computational Statistics and Data Science).
- Based in the School of Maths, but supervised by Laurence Aitchison (CS/Engineering Maths), Mengyue Yang (Engineering Maths), and Song Liu (Maths).

About Me

- Fourth (final) year PhD student at Bristol's Compass CDT (Computational Statistics and Data Science).
- Based in the School of Maths, but supervised by Laurence Aitchison (CS/Engineering Maths), Mengyue Yang (Engineering Maths), and Song Liu (Maths).
- Currently working on discrete diffusion models (training an 'auxilliary' model with VI to suggest the order in which to decode tokens).

About Me

- Fourth (final) year PhD student at Bristol's Compass CDT (Computational Statistics and Data Science).
- Based in the School of Maths, but supervised by Laurence Aitchison (CS/Engineering Maths), Mengyue Yang (Engineering Maths), and Song Liu (Maths).
- Currently working on discrete diffusion models (training an 'auxilliary' model with VI to suggest the order in which to decode tokens).
- Two projects I'll be talking about today: Alan (massively parallel probabilistic programming) & Bayesian Evals.

Alan: A Massively Parallel Probabilistic Programming Language

Work done with Laurence Aitchison and Thomas Heap over the first two years of my PhD.

Alan: A Massively Parallel Probabilistic Programming Language

Work done with Laurence Aitchison and Thomas Heap over the first two years of my PhD.

- Dual goals:

Alan: A Massively Parallel Probabilistic Programming Language

Work done with Laurence Aitchison and Thomas Heap over the first two years of my PhD.

- Dual goals:
  - Develop ‘massively parallel’ Bayesian inference algorithms: fast, accurate, and scalable; desinged for GPU acceleration.

Alan: A Massively Parallel Probabilistic Programming Language

Work done with Laurence Aitchison and Thomas Heap over the first two years of my PhD.

- Dual goals:
  - Develop ‘massively parallel’ Bayesian inference algorithms: fast, accurate, and scalable; desinged for GPU acceleration.
  - Implement these algorithms in a probabilistic programming language in pytorch (`alan`), allowing users to specify general probabilistic models.

Regular Bayesian Inference

- **Bayesian inference**: Prior $P(z)$ and likelihood $P(x|z)$ for latent variables z and data x.

$$P(z|x) = \frac{P(x|z)P(z)}{\int_{\mathcal{Z}} P(x, z')dz'}$$

Regular Bayesian Inference

- **Bayesian inference**: Prior $P(z)$ and likelihood $P(x|z)$ for latent variables z and data x.

$$P(z|x) = \frac{P(x|z)P(z)}{\int_{\mathcal{Z}} P(x, z')dz'}$$

- **Importance sampling**:

Regular Bayesian Inference

- **Bayesian inference**: Prior $P(z)$ and likelihood $P(x|z)$ for latent variables z and data x.

$$P(z|x) = \frac{P(x|z)P(z)}{\int_{\mathcal{Z}} P(x, z')dz'}$$

- **Importance sampling**:
  1. Sample K latent variables from a proposal distribution Q (usually IID):

$$z = (z^1, \dots, z^K) \sim Q(z).$$

Regular Bayesian Inference

- **Bayesian inference**: Prior $P(z)$ and likelihood $P(x|z)$ for latent variables z and data x.

$$P(z|x) = \frac{P(x|z)P(z)}{\int_{\mathcal{Z}} P(x, z')dz'}$$

- **Importance sampling**:
  1. Sample K latent variables from a proposal distribution Q (usually IID):

$$z = (z^1, \dots, z^K) \sim Q(z).$$

  2. Compute importance weights:

$$r_k(z) = \frac{P(x, z^k)}{Q(z^k)}.$$

Regular Bayesian Inference

- **Bayesian inference**: Prior $P(z)$ and likelihood $P(x|z)$ for latent variables z and data x.

$$P(z|x) = \frac{P(x|z)P(z)}{\int_{\mathcal{Z}} P(x, z')dz'}$$

- **Importance sampling**:
  1. Sample K latent variables from a proposal distribution Q (usually IID):
  $$z = (z^1, \dots, z^K) \sim Q(z).$$
  2. Compute importance weights:
  $$r_k(z) = \frac{P(x, z^k)}{Q(z^k)}.$$
  3. Approximate the normalising constant using the 'global' estimator:
  $$\mathcal{P}_{\text{global}}(z) = \frac{1}{K}\sum_{k=1}^{K} r_k(z) \quad \text{such that} \quad \mathbb{E}_{z\sim Q}[\mathcal{P}_{\text{global}}(z)] = P(x).$$

Limitations of Importance Sampling (IS)

- Global IS scales poorly with the number of latent variables, n.

Limitations of Importance Sampling (IS)

- Global IS scales poorly with the number of latent variables, n.
  - $z^k = (z_1^k, \dots, z_n^k) \in \mathcal{Z}$.

Limitations of Importance Sampling (IS)

- Global IS scales poorly with the number of latent variables, n.
  - $z^k = (z_1^k, \dots, z_n^k) \in \mathcal{Z}$.
- Chatterjee & Diaconis (2018) show that as n increases, the number of samples, K, must increase with $O(e^{D_{\mathrm{KL}}(P(z|x)||Q(z))}) \approx O(e^n)$. (This is a lot!)

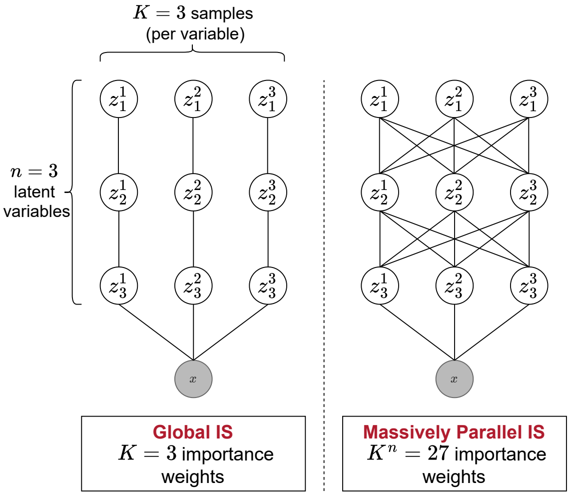
Limitations of Importance Sampling (IS)

- Global IS scales poorly with the number of latent variables, n.
  - $z^k = (z_1^k, \dots, z_n^k) \in \mathcal{Z}$.
- Chatterjee & Diaconis (2018) show that as n increases, the number of samples, K, must increase with $O(e^{D_{\mathrm{KL}}(P(z|x)||Q(z))}) \approx O(e^n)$. (This is a lot!)
- Solution: **Massively Parallel Importance Sampling (MP-IS)**

Limitations of Importance Sampling (IS)

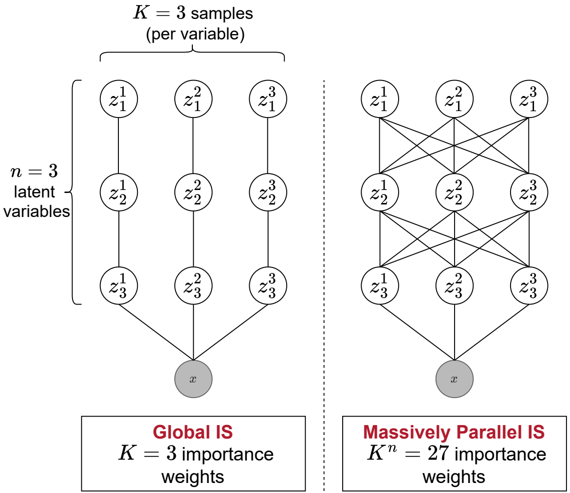
- Global IS scales poorly with the number of latent variables, n.
  - $z^k = (z_1^k, \ldots, z_n^k) \in \mathcal{Z}$.
- Chatterjee & Diaconis (2018) show that as n increases, the number of samples, K, must increase with $O(e^{D_{\mathrm{KL}}(P(z|x)||Q(z))}) \approx O(e^n)$. (This is a lot!)
- Solution: **Massively Parallel Importance Sampling (MP-IS)**
  - Reason about all K^n possible joint samples at once.

Massively Parallel Importance Sampling (MP-IS)



- Suppose each latent sample $z^k = (z_1^k, \ldots, z_n^k) \sim Q(z)$ is comprised of n variables.

Massively Parallel Importance Sampling (MP-IS)

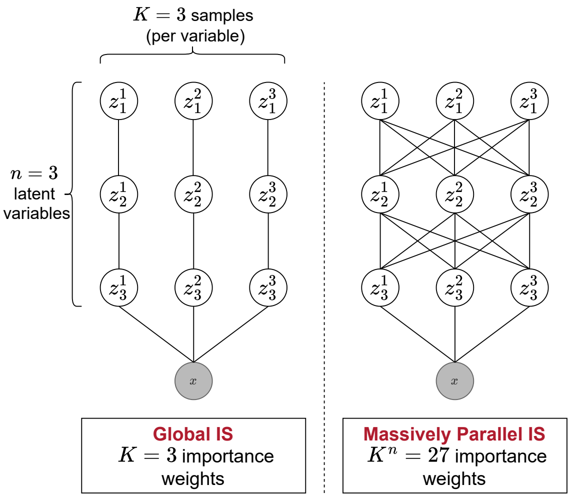


- Suppose each latent sample $z^k = (z_1^k, \ldots, z_n^k) \sim Q(z)$ is comprised of n variables.
- We can construct K^n different samples from the full joint space

$$(z_1^{k_1}, \ldots, z_n^{k_n}) \in \mathcal{Z}$$

where $\mathbf{k} = (k_1, \ldots, k_n) \in [K]^n$ is the indexing vector for each latent variable.

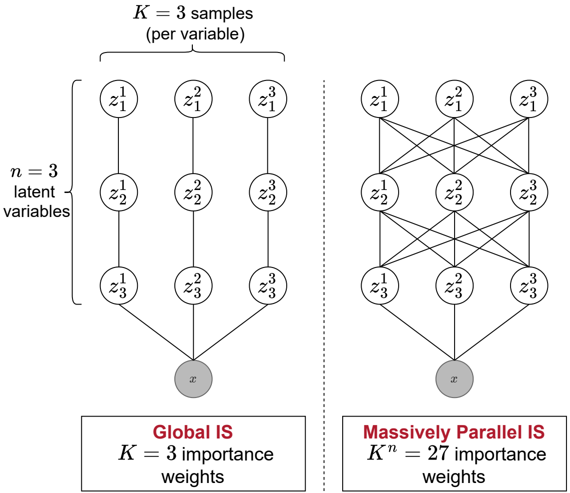
Massively Parallel Importance Sampling (MP-IS)



- Rather than using the global IS estimator

$$\mathcal{P}_{\text{global}}(z) = \frac{1}{K}\sum_{k=1}^{K}\frac{P(x, z^k)}{Q(z^k)}.$$

Massively Parallel Importance Sampling (MP-IS)



- Rather than using the global IS estimator

$$\mathcal{P}_{\text{global}}(z) = \frac{1}{K}\sum_{k=1}^{K}\frac{P(x, z^k)}{Q(z^k)}.$$

- ...we can use the MP-IS estimator

$$\mathcal{P}_{\text{MP}}(z) = \frac{1}{K^n}\sum_{\mathbf{k}\in[K]^n}\frac{P(x, z^{\mathbf{k}})}{Q_{\text{MP}}(z^{\mathbf{k}}, \mathbf{k})}.$$

(Which is still unbiased.)

MP-IS: Some Complications...

$$\mathcal{P}_{\text{MP}}(z) = \frac{1}{K^n} \sum_{\mathbf{k} \in [K]^n} \frac{P(x, z^{\mathbf{k}})}{Q_{\text{MP}}(z^{\mathbf{k}}, \mathbf{k})} = \frac{1}{K^n} \sum_{\mathbf{k} \in [K]^n} r_{\mathbf{k}}(z).$$

- We have to be careful about how we define Q_{MP} over the space of all K^n joint samples.

MP-IS: Some Complications...

$$\mathcal{P}_{\text{MP}}(z) = \frac{1}{K^n} \sum_{\mathbf{k} \in [K]^n} \frac{P(x, z^{\mathbf{k}})}{Q_{\text{MP}}(z^{\mathbf{k}}, \mathbf{k})} = \frac{1}{K^n} \sum_{\mathbf{k} \in [K]^n} r_{\mathbf{k}}(z).$$

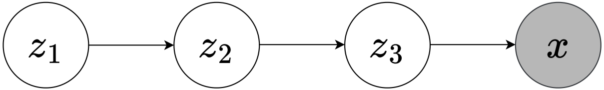
- We have to be careful about how we define Q_{MP} over the space of all K^n joint samples.
- Also, at first glance, this thing doesn't look all that nice to compute...

MP-IS: Some Complications...

$$\mathcal{P}_{\text{MP}}(z) = \frac{1}{K^n} \sum_{\mathbf{k} \in [K]^n} \frac{P(x, z^{\mathbf{k}})}{Q_{\text{MP}}(z^{\mathbf{k}}, \mathbf{k})} = \frac{1}{K^n} \sum_{\mathbf{k} \in [K]^n} r_{\mathbf{k}}(z).$$

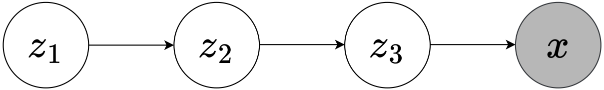
- We have to be careful about how we define Q_{MP} over the space of all K^n joint samples.
- Also, at first glance, this thing doesn't look all that nice to compute...
- But we can exploit the conditional independencies in the model to render it tractable.

MP-IS: Some Complications...



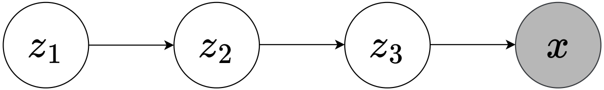
- E.g. with the model from before with $n = 3$, $P(x, z) = P(z_1)P(z_2|z_1)P(z_3|z_2)P(x|z_3)$, we can move the sums inside the product and get a bunch of tensor products:

MP-IS: Some Complications...



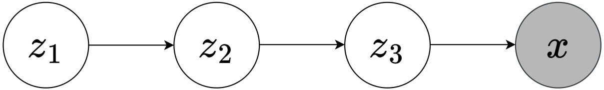
- E.g. with the model from before with $n = 3$, $P(x, z) = P(z_1)P(z_2|z_1)P(z_3|z_2)P(x|z_3)$, we can move the sums inside the product and get a bunch of tensor products:

MP-IS: Some Complications...



- E.g. with the model from before with $n = 3$, $P(x, z) = P(z_1)P(z_2|z_1)P(z_3|z_2)P(x|z_3)$, we can move the sums inside the product and get a bunch of tensor products:

$$\mathcal{P}_{\text{MP}}(z) = \frac{1}{K^3} \sum_{k_1 \in [K]} \sum_{k_2 \in [K]} \sum_{k_3 \in [K]} \frac{P(z_1^{k_1})P(z_2^{k_2}|z_1^{k_1})P(z_3^{k_3}|z_2^{k_2})P(x|z_3^{k_3})}{Q(z_1^{k_1})Q(z_2^{k_2})Q(z_3^{k_3})}$$



- E.g. with the model from before with $n = 3$, $P(x, z) = P(z_1)P(z_2|z_1)P(z_3|z_2)P(x|z_3)$, we can move the sums inside the product and get a bunch of tensor products:

$$\mathcal{P}_{\mathrm{MP}}(z) = \frac{1}{K^3} \sum_{k_1 \in [K]} \sum_{k_2 \in [K]} \sum_{k_3 \in [K]} \frac{P(z_1^{k_1})P(z_2^{k_2}|z_1^{k_1})P(z_3^{k_3}|z_2^{k_2})P(x|z_3^{k_3})}{Q(z_1^{k_1})Q(z_2^{k_2})Q(z_3^{k_3})}$$

$$= \frac{1}{K^3} \sum_{k_1 \in [K]} \underbrace{\frac{P(z_1^{k_1})}{Q(z_1^{k_1})}}_{\text{Vector of size } K} \sum_{k_2 \in [K]} \underbrace{\frac{P(z_2^{k_2}|z_1^{k_1})}{Q(z_2^{k_2})}}_{\text{Matrix of size } K \times K} \sum_{k_3 \in [K]} \underbrace{\frac{P(z_3^{k_3}|z_2^{k_2})}{Q(z_3^{k_3})}}_{\text{Matrix of size } K \times K} \underbrace{P(x|z_3^{k_3})}_{\text{Vector of size } K}$$

Can we hide these complications from the user?

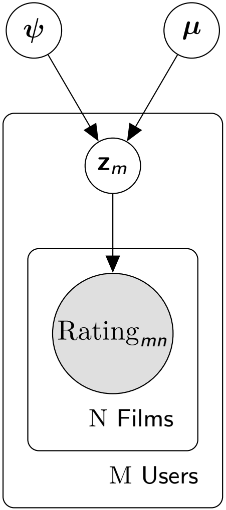
- Yes! We do this with `alan`.

Can we hide these complications from the user?

- Yes! We do this with `alan`.
- User specifies the model with P and Q as pytorch modules, and we handle the massively parallel inference for them.

Alan: A Probabilistic Programming Language

```python
from alan import Normal, Bernoulli, Plate, BoundPlate, OptParam, Data, Problem
import torch as t

# Set up the model
d_z = 10

P = Plate(
    mu_z = Normal(t.zeros((d_z,)), t.ones((d_z,))),
    psi_z = Normal(t.zeros((d_z,)), t.ones((d_z,))),
    plate_1 = Plate(
        z = Normal("mu_z", lambda psi_z: psi_z.exp()),
        plate_2 = Plate(
            obs = Bernoulli(logits = lambda z, x: z @ x),
        )
    ),
)

Q = Plate(
    mu_z = Normal(OptParam(t.zeros((d_z,))), OptParam(t.zeros((d_z,)), transformation=t.exp)),
    psi_z = Normal(OptParam(t.zeros((d_z,))), OptParam(t.zeros((d_z,)), transformation=t.exp)),
    plate_1 = Plate(
        z = Normal(OptParam(t.zeros((d_z,))), OptParam(t.zeros((d_z,)), transformation=t.exp)),
        plate_2 = Plate(
            obs = Data()
        )
    ),
)

P = BoundPlate(P, platesizes={'plate_1': num_users, 'plate_2': num_movies}, inputs = {'x': x})
Q = BoundPlate(Q, platesizes={'plate_1': num_users, 'plate_2': num_movies}, inputs = {'x': x})

prob = Problem(P, Q)
```



MP-VI

- Using $\mathcal{P}_{\text{MP}}(z)$, we can do variational inference (VI) by maximising the ELBO:

$$\log P(x) \geq \mathcal{L}_{\text{MP}}(\theta) = \mathbb{E}_{z \sim Q_{\text{MP}}(\theta)}[\log \mathcal{P}_{\text{MP}}(z)]$$

MP-VI

- Using $\mathcal{P}_{\text{MP}}(z)$, we can do variational inference (VI) by maximising the ELBO:

$$\log P(x) \geq \mathcal{L}_{\text{MP}}(\theta) = \mathbb{E}_{z \sim Q_{\text{MP}}(\theta)}[\log \mathcal{P}_{\text{MP}}(z)]$$

- Aitchison (2019) showed that MP-VI is a tighter bound than the global VI objective (IWAE):

$$\log P(x) \geq \mathcal{L}_{\text{MP}}(\theta) \geq \mathcal{L}_{\text{global}}(\theta) = \mathbb{E}_{z \sim Q(\theta)}[\log \mathcal{P}_{\text{global}}(z)]$$

MP-VI

- Using $\mathcal{P}_{\text{MP}}(z)$, we can do variational inference (VI) by maximising the ELBO:
$$\log P(x) \geq \mathcal{L}_{\text{MP}}(\theta) = \mathbb{E}_{z \sim Q_{\text{MP}}(\theta)}[\log \mathcal{P}_{\text{MP}}(z)]$$
- Aitchison (2019) showed that MP-VI is a tighter bound than the global VI objective (IWAE):
$$\log P(x) \geq \mathcal{L}_{\text{MP}}(\theta) \geq \mathcal{L}_{\text{global}}(\theta) = \mathbb{E}_{z \sim Q(\theta)}[\log \mathcal{P}_{\text{global}}(z)]$$

MP-VI

- Using $\mathcal{P}_{\text{MP}}(z)$, we can do variational inference (VI) by maximising the ELBO:

$$\log P(x) \geq \mathcal{L}_{\text{MP}}(\theta) = \mathbb{E}_{z \sim Q_{\text{MP}}(\theta)}[\log \mathcal{P}_{\text{MP}}(z)]$$

- Aitchison (2019) showed that MP-VI is a tighter bound than the global VI objective (IWAE):

$$\log P(x) \geq \mathcal{L}_{\text{MP}}(\theta) \geq \mathcal{L}_{\text{global}}(\theta) = \mathbb{E}_{z \sim Q(\theta)}[\log \mathcal{P}_{\text{global}}(z)]$$

```python
P = BoundPlate(P, platesizes={'plate_1': num_users, 'plate_2': num_movies}, inputs = {'x': x})
Q = BoundPlate(Q, platesizes={'plate_1': num_users, 'plate_2': num_movies}, inputs = {'x': x})

prob = Problem(P, Q)
opt = t.optim.Adam(prob.Q.parameters(), lr=lr)

# Train Q with VI
for i in range(num_iterations):
    opt.zero_grad()
    elbo = prob.sample(K=K).elbo_vi()
    elbo.backward()
    opt.step()
```

MP Algorithms

- We can obtain unbiased posterior moment estimates via autodiff (Bowyer et al. (2024)).

$$m_{\mathrm{MP}}(z) = \frac{1}{K^n} \sum_{\mathbf{k} \in [K]^n} \frac{r_{\mathbf{k}}(z)}{\mathcal{P}_{\mathrm{MP}}(z)} m(z^{\mathbf{k}})$$

MP Algorithms

- We can obtain unbiased posterior moment estimates via autodiff (Bowyer et al. (2024)).

$$m_{\text{MP}}(z) = \frac{1}{K^n} \sum_{\mathbf{k} \in [K]^n} \frac{r_{\mathbf{k}}(z)}{\mathcal{P}_{\text{MP}}(z)} m(z^{\mathbf{k}})$$

- We define a modified marginal likelihood estimator with an auxiliary variable $J \in \mathbb{R}$:

$$\mathcal{P}_{\text{MP}}^{\exp}(z, J) = \frac{1}{K^n} \sum_{\mathbf{k} \in [K]^n} r_{\mathbf{k}}(z) \exp(J m(z^{\mathbf{k}}))$$

MP Algorithms

- We can obtain unbiased posterior moment estimates via autodiff (Bowyer et al. (2024)).

$$m_{\mathrm{MP}}(z) = \frac{1}{K^n} \sum_{\mathbf{k} \in [K]^n} \frac{r_{\mathbf{k}}(z)}{\mathcal{P}_{\mathrm{MP}}(z)} m(z^{\mathbf{k}})$$

- We define a modified marginal likelihood estimator with an auxiliary variable $J \in \mathbb{R}$:

$$\mathcal{P}_{\mathrm{MP}}^{\exp}(z, J) = \frac{1}{K^n} \sum_{\mathbf{k} \in [K]^n} r_{\mathbf{k}}(z) \exp(J m(z^{\mathbf{k}}))$$

- Then differentiating the log of this with respect to J and setting $J = 0$ we get:

$$\frac{\partial}{\partial J}\bigg|_{J=0} \log \mathcal{P}_{\mathrm{MP}}^{\exp}(z, J) = \frac{\frac{\partial}{\partial J}\Big|_{J=0} \mathcal{P}_{\mathrm{MP}}^{\exp}(z, J)}{\mathcal{P}_{\mathrm{MP}}^{\exp}(z, 0)} = m_{\mathrm{MP}}(z)$$

MP Algorithms

- We can obtain unbiased posterior moment estimates via autodiff (Bowyer et al. (2024)).

$$m_{\text{MP}}(z) = \frac{1}{K^n} \sum_{\mathbf{k} \in [K]^n} \frac{r_{\mathbf{k}}(z)}{\mathcal{P}_{\text{MP}}(z)} m(z^{\mathbf{k}})$$

- We define a modified marginal likelihood estimator with an auxiliary variable $J \in \mathbb{R}$:

$$\mathcal{P}_{\text{MP}}^{\exp}(z, J) = \frac{1}{K^n} \sum_{\mathbf{k} \in [K]^n} r_{\mathbf{k}}(z) \exp(J m(z^{\mathbf{k}}))$$

- Then differentiating the log of this with respect to J and setting $J = 0$ we get:

$$\left.\frac{\partial}{\partial J}\right|_{J=0} \log \mathcal{P}_{\text{MP}}^{\exp}(z, J) = \frac{\left.\frac{\partial}{\partial J}\right|_{J=0} \mathcal{P}_{\text{MP}}^{\exp}(z, J)}{\mathcal{P}_{\text{MP}}^{\exp}(z, 0)} = m_{\text{MP}}(z)$$

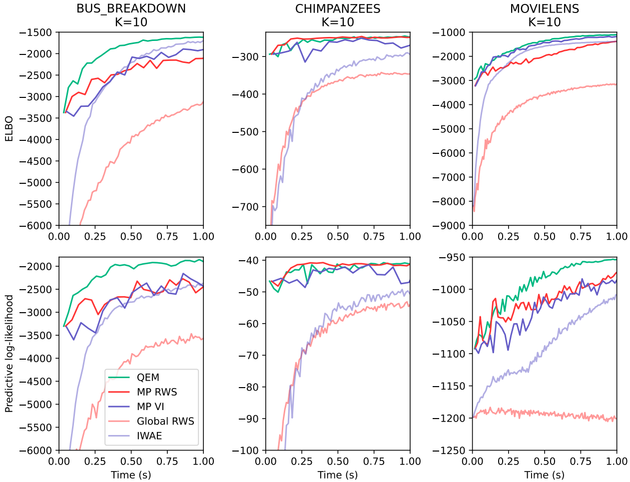
- By similar arguments: $J \in \mathbb{R}^K$ gives us marginal importance weights; $J \in \mathbb{R}^{K^{1+|pa(i)|}}$ gives us importance samples for z_i, given its parents $pa(i)$.

QEM: An Adaptive Importance Sampling Algorithm

QEM (Heap et al. (2025))

1. Start with an initial approximate posterior Q_0.
2. Compute posterior moment estimates $m_{\text{MP}}(z)$ using MP-IS.
3. Update the approximate posterior Q_{t+1} using the moment estimates.

Can be seen as an EM-like algorithm for adaptive imortance sampling.



Conclusions

- This project was a great (if, at times, complicated) way to learn about Bayesian inference and numerical and probabilistic programming.

Conclusions

- This project was a great (if, at times, complicated) way to learn about Bayesian inference and numerical and probabilistic programming.
- The results were pretty promising, but but there are some drawbacks to massively parallel methods:

Conclusions

- This project was a great (if, at times, complicated) way to learn about Bayesian inference and numerical and probabilistic programming.
- The results were pretty promising, but but there are some drawbacks to massively parallel methods:
    - The algorithms are complex to implement (hence wrapping them in a PPL).

Conclusions

- This project was a great (if, at times, complicated) way to learn about Bayesian inference and numerical and probabilistic programming.
- The results were pretty promising, but but there are some drawbacks to massively parallel methods:
  - The algorithms are complex to implement (hence wrapping them in a PPL).
  - Not all models have lots of conditional independencies to exploit.

Conclusions

- This project was a great (if, at times, complicated) way to learn about Bayesian inference and numerical and probabilistic programming.
- The results were pretty promising, but but there are some drawbacks to massively parallel methods:
  - The algorithms are complex to implement (hence wrapping them in a PPL).
  - Not all models have lots of conditional independencies to exploit.
  - Although it's slower and harder to tune, HMC is often hard to beat in terms of quality of inference.

Bayesian Evals: Uncertainty Quantification for LLM Evals



Work done with Laurence Aitchison and Desi R. Ivanova.

- Two directions:

Bayesian Evals: Uncertainty Quantification for LLM Evals

Work done with Laurence Aitchison and Desi R. Ivanova.

- Two directions:
  - Improved UQ for evals with Bayesian methods.

Bayesian Evals: Uncertainty Quantification for LLM Evals

Work done with Laurence Aitchison and Desi R. Ivanova.

- Two directions:
  - Improved UQ for evals with Bayesian methods.
  - Interpretability of evals with Bayesian hierarchical modelling and SAE-like approaches.

Bayesian Evals: Uncertainty Quantification for LLM Evals

Work done with Laurence Aitchison and Desi R. Ivanova.

- Two directions:
  - Improved UQ for evals with Bayesian methods.
  - Interpretability of evals with Bayesian hierarchical modelling and SAE-like approaches.
- The former direction led to an ICML spotlight position paper: *'Position: Don't Use the CLT in LLM Evals With Fewer Than a Few Hundred Datapoints'*.

Bayesian Evals: Uncertainty Quantification for LLM Evals

Work done with Laurence Aitchison and Desi R. Ivanova.

- Two directions:
    - Improved UQ for evals with Bayesian methods.
    - Interpretability of evals with Bayesian hierarchical modelling and SAE-like approaches.
- The former direction led to an ICML spotlight position paper: *'Position: Don't Use the CLT in LLM Evals With Fewer Than a Few Hundred Datapoints'*.
- The latter fell by the wayside, but is something I'd like to come back to at some point.

Motivation

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

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

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

Motivation

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

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

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

- The CLT-based confidence interval is:

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

where $z_{\alpha/2}$ is the $100(1-\alpha/2)$-th percentile of $\mathcal{N}(0,1)$ and $\mathrm{SE}(\hat{\mu}) = \sqrt{\frac{\hat{\sigma}^2}{N}}$ is the standard error of the sample mean.

Motivation

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

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

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

- The CLT-based confidence interval is:

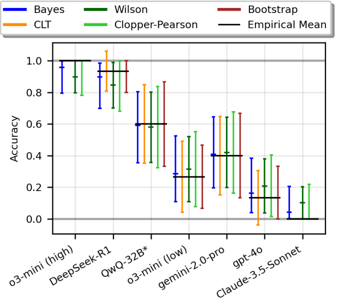
$$\mathrm{CI}_{1-\alpha}(\mu) = \hat{\mu} \pm z_{\alpha/2}\mathrm{SE}(\hat{\mu})$$

  where $z_{\alpha/2}$ is the $100(1-\alpha/2)$-th percentile of $\mathcal{N}(0,1)$ and $\mathrm{SE}(\hat{\mu}) = \sqrt{\frac{\hat{\sigma}^2}{N}}$ is the standard error of the sample mean.
- In the case of binary data $X_i \in \{0, 1\}$, this becomes:

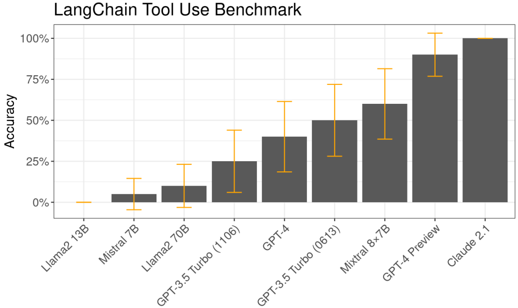
$$\mathrm{CI}_{1-\alpha}(\theta) = \bar{X} \pm z_{\alpha/2}\sqrt{\bar{X}(1-\bar{X})/N}.$$

Real-World Failures of the CLT

- If N is too small, CLT-based error bars can collapse to zero-width or extend past $[0, 1]$.



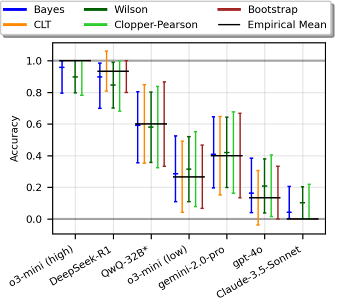
Math Arena's AIME II 2025 Benchmark (N=15). Various 95% interval types shown.



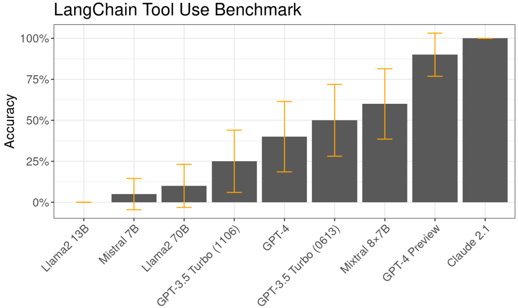
Langchain Typewriter Tool Use Benchmark (N=20). CLT-based 95% intervals only.

Real-World Failures of the CLT

- If N is too small, CLT-based error bars can collapse to zero-width or extend past $[0, 1]$.
- Smaller, more intricate, and expensive LLM benchmarks are becoming increasingly common, so we need to find alternatives for the few-data regime.



Math Arena's AIME II 2025 Benchmark (N=15). Various 95% interval types shown.



Langchain Typewriter Tool Use Benchmark (N=20). CLT-based 95% intervals only.

Bayesian Alternative: Beta-Binomial Model

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

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

Bayesian Alternative: Beta-Binomial Model

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

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

- Say y_i is correct if $y_i = 1$ and incorrect if $y_i = 0$.

Bayesian Alternative: Beta-Binomial Model

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

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

- Say y_i is correct if $y_i = 1$ and incorrect if $y_i = 0$.
- Obtain quantile-based Bayesian credible intervals for θ from the closed form posterior.

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

Bayesian Alternative: Beta-Binomial Model

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

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

- Say y_i is correct if $y_i = 1$ and incorrect if $y_i = 0$.
- Obtain quantile-based Bayesian credible intervals for θ from the closed form posterior.

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

Bayesian Alternative: Beta-Binomial Model

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

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

- Say y_i is correct if $y_i = 1$ and incorrect if $y_i = 0$.
- Obtain quantile-based Bayesian credible intervals for θ from the closed form posterior.

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

Beta-Bernoulli Bayesian Credible Interval

```
posterior = scipy.stats.beta(1 + sum(y), 1 + N - sum(y))
bayes_ci  = posterior.interval(confidence=0.95)
```

Frequentist Alternatives

- Wilson score interval: Based on normal approximation to the binomial distribution.

Frequentist Alternatives

- Wilson score interval: Based on normal approximation to the binomial distribution.

Frequentist Alternatives

- Wilson score interval: Based on normal approximation to the binomial distribution.

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

Frequentist Alternatives

- Wilson score interval: Based on normal approximation to the binomial distribution.

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

- Clopper-Pearson 'exact' interval: A 'worst-case' approach; very conservative method guaranteed to never under-cover. $\mathrm{CI}_{1-\alpha,\mathrm{CP}}(\theta) = [\theta_{\mathrm{lower}}, \theta_{\mathrm{upper}}]$,

Frequentist Alternatives

- Wilson score interval: Based on normal approximation to the binomial distribution.

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

- Clopper-Pearson 'exact' interval: A 'worst-case' approach; very conservative method guaranteed to never under-cover. $\mathrm{CI}_{1-\alpha,\text{CP}}(\theta) = [\theta_{\text{lower}}, \theta_{\text{upper}}]$,

Frequentist Alternatives

- Wilson score interval: Based on normal approximation to the binomial distribution.

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

- Clopper-Pearson 'exact' interval: A 'worst-case' approach; very conservative method guaranteed to never under-cover. $\mathrm{CI}_{1-\alpha,\mathrm{CP}}(\theta) = [\theta_{\mathrm{lower}}, \theta_{\mathrm{upper}}]$,

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

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

Frequentist Alternatives

- Wilson score interval: Based on normal approximation to the binomial distribution.

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

- Clopper-Pearson 'exact' interval: A 'worst-case' approach; very conservative method guaranteed to never under-cover. $\mathrm{CI}_{1-\alpha,\mathrm{CP}}(\theta) = [\theta_{\mathrm{lower}}, \theta_{\mathrm{upper}}]$,

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

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

Wilson & Clopper-Pearson Confidence Interval

```
result = scipy.stats.binomtest(k=sum(y), n=N)
wilson_ci = result.proportion_ci("wilson", 0.95)
clop_ci   = result.proportion_ci("exact",  0.95)
```

Interval Comparison Simulations

We use synthetic eval data so that we know the true parameter θ.

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

Interval Comparison Simulations

We use synthetic eval 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.

Interval Comparison Simulations

We use synthetic eval 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 intervals with various methods for various $1 - \alpha$ confidence levels.

Interval Comparison Simulations

We use synthetic eval 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 intervals with various methods for various $1 - \alpha$ confidence levels.
- Repeat many times and calculate the true coverage and average width of the intervals.

Interval Comparison Simulations

We use synthetic eval data so that we know the true parameter θ.

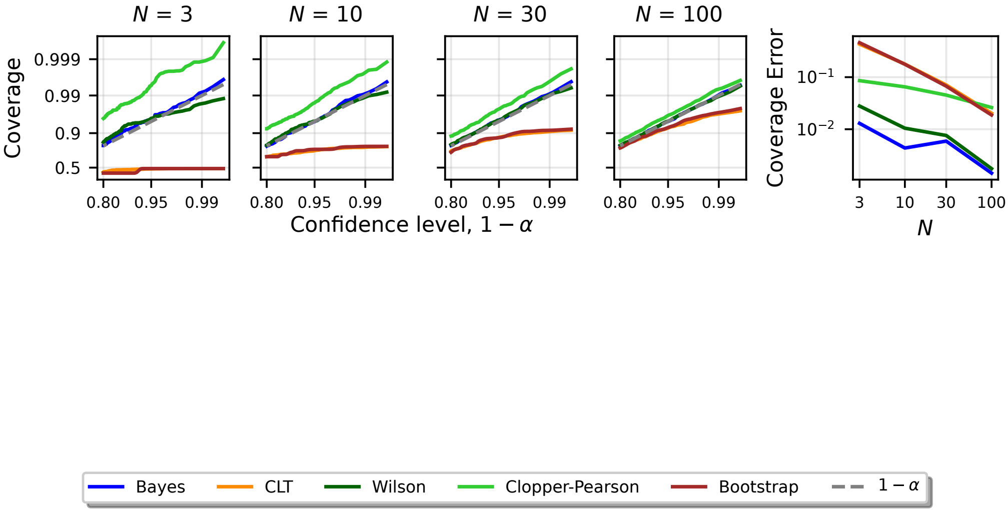
- Draw $\theta \sim \text{Uniform}[0, 1]$.
- Draw $N \in \{3, 10, 30, 100\}$ IID Bernoulli datapoints with parameter θ.
- Construct intervals with various methods for various $1 - \alpha$ confidence levels.
- Repeat many times and calculate the true coverage and average width of the intervals.
  - Coverage: What proportion of the time does a $1 - \alpha$ confidence-level interval actually contain the true parameter? (A frequentist metric, really.)

Interval Comparison Simulations

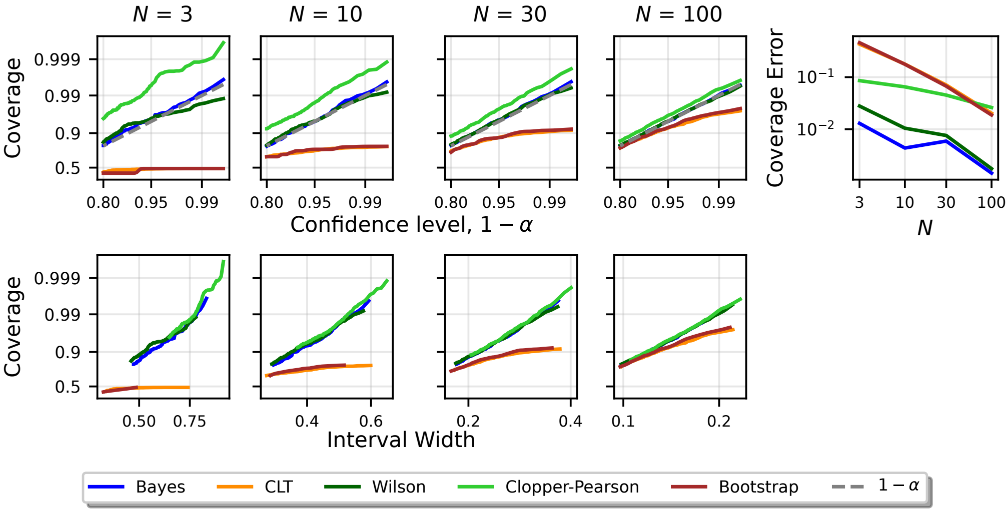
We use synthetic eval data so that we know the true parameter θ.

- Draw $\theta \sim \text{Uniform}[0, 1]$.
- Draw $N \in \{3, 10, 30, 100\}$ IID Bernoulli datapoints with parameter θ.
- Construct intervals with various methods for various $1 - \alpha$ confidence levels.
- Repeat many times and calculate the true coverage and average width of the intervals.
  - Coverage: What proportion of the time does a $1 - \alpha$ confidence-level interval actually contain the true parameter? (A frequentist metric, really.)
  - Ideally, coverage $= 1 - \alpha$.

IID Questions Setting: Results



IID Questions Setting: Results



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.
- **Comparisons Between Two Models, θ_A and θ_B**

Other Eval Settings

- **Clustered Questions**
  - Instead of N IID questions, we have T tasks, each with N_t IID questions.
- **Comparisons Between Two Models, θ_A and θ_B**
  - Independent Comparisons (using $N_A, N_B, \bar{y}_A, \bar{y}_B$ only).

Other Eval Settings

- **Clustered Questions**
  - Instead of N IID questions, we have T tasks, each with N_t IID questions.
- **Comparisons Between Two Models, θ_A and θ_B**
  - Independent Comparisons (using $N_A, N_B, \bar{y}_A, \bar{y}_B$ only).
  - Paired Comparisons (using $N_A = N_B, \{y_{A;i}\}_{i=1}^{N}, \{y_{B;i}\}_{i=1}^{N}$).

Other Eval Settings

- **Clustered Questions**
  - Instead of N IID questions, we have T tasks, each with N_t IID questions.
- **Comparisons Between Two Models, θ_A and θ_B**
  - Independent Comparisons (using $N_A, N_B, \bar{y}_A, \bar{y}_B$ only).
  - Paired Comparisons (using $N_A = N_B, \{y_{A;i}\}_{i=1}^{N}, \{y_{B;i}\}_{i=1}^{N}$).
  - With Bayes you can calculate $\mathbb{P}(\theta_A > \theta_B | y_A, y_B)$.

Other Eval Settings

- **Clustered Questions**
  - Instead of N IID questions, we have T tasks, each with N_t IID questions.
- **Comparisons Between Two Models, θ_A and θ_B**
  - Independent Comparisons (using $N_A, N_B, \bar{y}_A, \bar{y}_B$ only).
  - Paired Comparisons (using $N_A = N_B, \{y_{A;i}\}_{i=1}^N, \{y_{B;i}\}_{i=1}^N$).
  - With Bayes you can calculate $\mathbb{P}(\theta_A > \theta_B | y_A, y_B)$.
- **Metrics that aren't simple averages of binary results** (e.g. F1 score).

Other Eval Settings

- **Clustered Questions**
  - Instead of N IID questions, we have T tasks, each with N_t IID questions.
- **Comparisons Between Two Models, θ_A and θ_B**
  - Independent Comparisons (using $N_A, N_B, \bar{y}_A, \bar{y}_B$ only).
  - Paired Comparisons (using $N_A = N_B, \{y_{A;i}\}_{i=1}^N, \{y_{B;i}\}_{i=1}^N$).
  - With Bayes you can calculate $\mathbb{P}(\theta_A > \theta_B | y_A, y_B)$.
- **Metrics that aren't simple averages of binary results** (e.g. F1 score).
- **Prior Mismatch** (i.e. what if the uniform prior is incorrect, $\theta \not\sim \text{Uniform}[0, 1]$?).

Clustered Questions Setting: Frequentist Approach

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

Clustered Questions Setting: Frequentist Approach

- Instead of N IID questions, we have T tasks, each with N_t IID questions.
- Some tasks are easier than others.

Clustered Questions Setting: Frequentist Approach

- Instead of N IID questions, we have T tasks, each with N_t IID questions.
- Some tasks are easier than others.
- Frequentist approach (Miller, 2024):

$$\mathrm{CI}_{1-\alpha}(\theta) = \bar{X} \pm z_{\alpha/2}\mathrm{SE}_{\mathrm{Clustered}}$$

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

Clustered Questions Setting: A Bayesian Approach

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

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

Clustered Questions Setting: A Bayesian Approach

- '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: A Bayesian Approach

- '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 $\mathbb{E}[\theta_t|\theta] = \theta$),

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

Clustered Questions Setting: A Bayesian Approach

- '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 $\mathbb{E}[\theta_t|\theta] = \theta$),

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

- $\text{Var}(\theta_t|\theta) = \frac{\theta(1-\theta)}{d^2+1}$.

Clustered Questions Setting: A Bayesian Approach

- '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 $\mathbb{E}[\theta_t|\theta] = \theta$),

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

- $\text{Var}(\theta_t|\theta) = \frac{\theta(1-\theta)}{d^2+1}$.
- If d is large, then θ_t is close to θ for all tasks.

Clustered Questions Setting: A Bayesian Approach

- '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 $\mathbb{E}[\theta_t|\theta] = \theta$),

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

- $\text{Var}(\theta_t|\theta) = \frac{\theta(1-\theta)}{d^2+1}$.
- If d is large, then θ_t is close to θ for all tasks.
- If d is small, then θ_t is more variable across tasks.

Clustered Questions Setting: A Bayesian Approach

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

Clustered Questions Setting: A Bayesian Approach

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

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

Clustered Questions Setting: A Bayesian Approach

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

- 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: A Bayesian Approach

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

- 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: A Bayesian Approach

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

- Can we integrate out the per-task difficulty parameters θ_t?
- Yes! The number of successes per task is Beta-Binomial distributed:

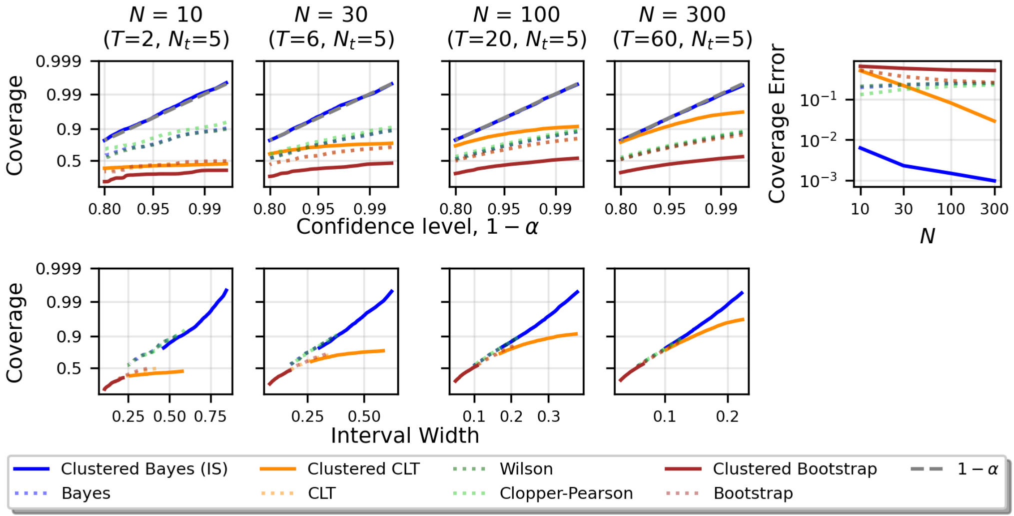
$$\sum_{i=1}^{N_t} y_{i,t} = Y_t \sim \text{BetaBinomial}(N_t, d\theta, d(1-\theta)).$$

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

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

- Compute quantile-based Bayesian credible intervals for θ.

Clustered Questions Setting: Results



Conclusion

Advice to practitioners:

- Use Bayesian Beta-Bernoulli or Wilson Score intervals for IID setting.

Conclusion

Advice to practitioners:

- Use Bayesian Beta-Bernoulli or Wilson Score intervals for IID setting.
- Use simple Bayesian models for other settings (for flexibility and interpretability).

Conclusion

Advice to practitioners:

- Use Bayesian Beta-Bernoulli or Wilson Score intervals for IID setting.
- Use simple Bayesian models for other settings (for flexibility and interpretability).
- It's **easy** (use `scipy` or `bayes_evals`).

Conclusion

Advice to practitioners:

- Use Bayesian Beta-Bernoulli or Wilson Score intervals for IID setting.
- Use simple Bayesian models for other settings (for flexibility and interpretability).
- It's **easy** (use `scipy` or `bayes_evals`).
- It's **safer** than CLT-based methods.

Conclusion

Advice to practitioners:

- Use Bayesian Beta-Bernoulli or Wilson Score intervals for IID setting.
- Use simple Bayesian models for other settings (for flexibility and interpretability).
- It's **easy** (use `scipy` or `bayes_evals`).
- It's **safer** than CLT-based methods.
- It's still **cheap** for large N.

Conclusion

Advice to practitioners:

- Use Bayesian Beta-Bernoulli or Wilson Score intervals for IID setting.
- Use simple Bayesian models for other settings (for flexibility and interpretability).
- It's **easy** (use `scipy` or `bayes_evals`).
- It's **safer** than CLT-based methods.
- It's still **cheap** for large N.

Conclusion

Advice to practitioners:

- Use Bayesian Beta-Bernoulli or Wilson Score intervals for IID setting.
- Use simple Bayesian models for other settings (for flexibility and interpretability).
- It's **easy** (use `scipy` or `bayes_evals`).
- It's **safer** than CLT-based methods.
- It's still **cheap** for large N.

My takeaways from this project:

- The project really cemented my understanding of Bayesianism vs Frequentism in practice.

Conclusion

Advice to practitioners:

- Use Bayesian Beta-Bernoulli or Wilson Score intervals for IID setting.
- Use simple Bayesian models for other settings (for flexibility and interpretability).
- It's **easy** (use `scipy` or `bayes_evals`).
- It's **safer** than CLT-based methods.
- It's still **cheap** for large N.

My takeaways from this project:

- The project really cemented my understanding of Bayesianism vs Frequentism in practice.
- There's a big communication gap between AI researchers and classical statisticians.

Conclusion

Advice to practitioners:

- Use Bayesian Beta-Bernoulli or Wilson Score intervals for IID setting.
- Use simple Bayesian models for other settings (for flexibility and interpretability).
- It's **easy** (use `scipy` or `bayes_evals`).
- It's **safer** than CLT-based methods.
- It's still **cheap** for large N.

My takeaways from this project:

- The project really cemented my understanding of Bayesianism vs Frequentism in practice.
- There's a big communication gap between AI researchers and classical statisticians.
- UQ for evals is an incredibly rich space.

Bayesian Evals for Interpretability

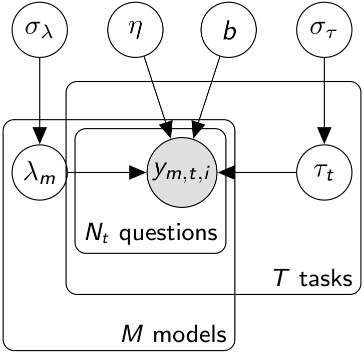
- Apply hierarchical Bayesian modelling to evals: infer latent variables for benchmark difficulty and model performance.

Bayesian Evals for Interpretability

- Apply hierarchical Bayesian modelling to evals: infer latent variables for benchmark difficulty and model performance.

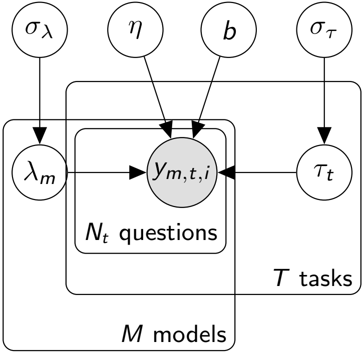
Bayesian Evals for Interpretability

- Apply hierarchical Bayesian modelling to evals: infer latent variables for benchmark difficulty and model performance.

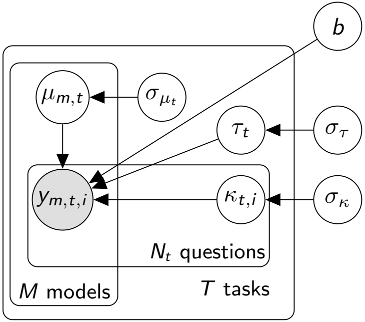


Task-based model.

- Apply hierarchical Bayesian modelling to evals: infer latent variables for benchmark difficulty and model performance.



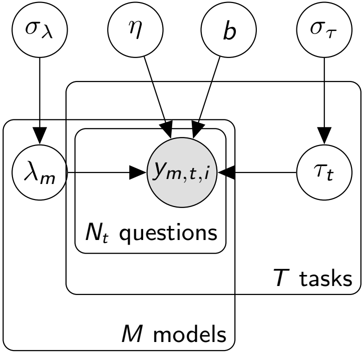
Task-based model.



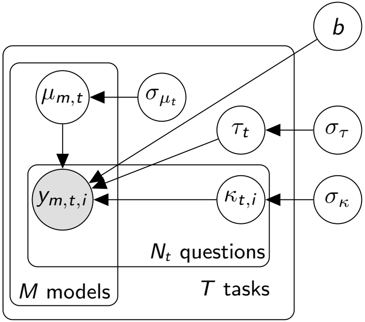
Question-task difficulty model.

Bayesian Evals for Interpretability

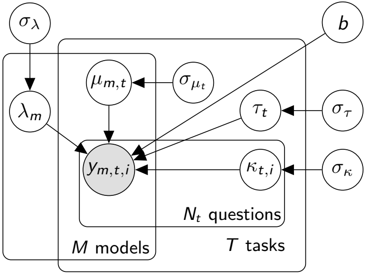
- Apply hierarchical Bayesian modelling to evals: infer latent variables for benchmark difficulty and model performance.



Task-based model.



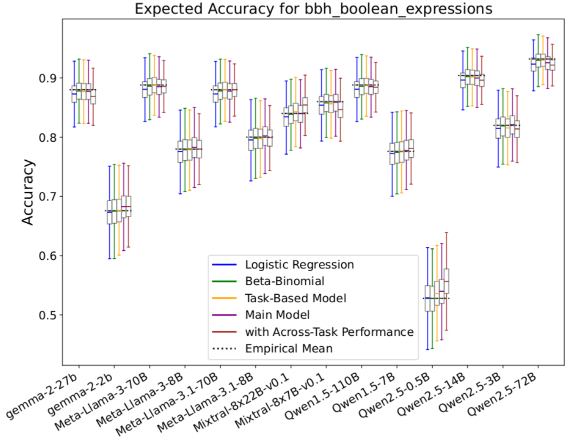
Question-task difficulty model.

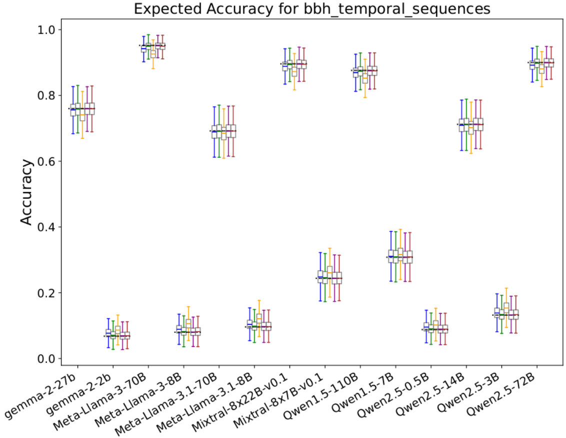


+ Across-task performance latent variable.

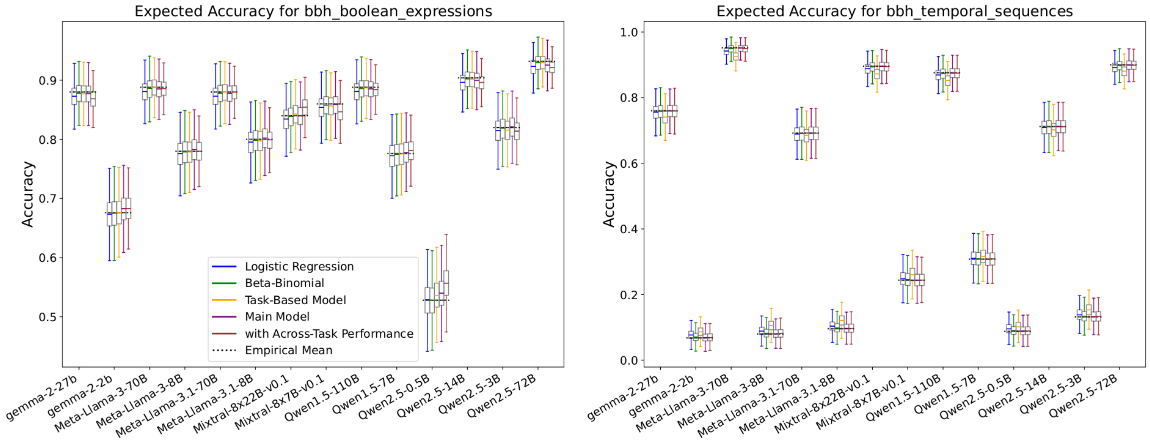
Graphical models of the proposed hierarchical models. Way too complicated!

Bayesian Evals for Interpretability





Bayesian Evals for Interpretability



So just use Beta-Binomial!

Future Directions

- If we make our latent variables for ‘model performance’ and ‘benchmark difficulty’ multivariate and sparse, maybe we can interpret their dimensions as different measures of model capability?

Future Directions

- If we make our latent variables for ‘model performance’ and ‘benchmark difficulty’ multivariate and sparse, maybe we can interpret their dimensions as different measures of model capability?
  - E.g. one dimension is ‘physics knowledge’, another is ‘reasoning ability’ etc.

Future Directions

- If we make our latent variables for ‘model performance’ and ‘benchmark difficulty’ multivariate and sparse, maybe we can interpret their dimensions as different measures of model capability?
  - E.g. one dimension is ‘physics knowledge’, another is ‘reasoning ability’ etc.
- The results for this were interesting, but not as straightforwardly interpretable as we hoped.

Future Directions

- If we make our latent variables for 'model performance' and 'benchmark difficulty' multivariate and sparse, maybe we can interpret their dimensions as different measures of model capability?
  - E.g. one dimension is 'physics knowledge', another is 'reasoning ability' etc.
- The results for this were interesting, but not as straightforwardly interpretable as we hoped.
- Perhaps these techniques could be used more successfully for predicting eval performance based on the mix of pretraining data?

Future Directions

- If we make our latent variables for ‘model performance’ and ‘benchmark difficulty’ multivariate and sparse, maybe we can interpret their dimensions as different measures of model capability?
  - E.g. one dimension is ‘physics knowledge’, another is ‘reasoning ability’ etc.
- The results for this were interesting, but not as straightforwardly interpretable as we hoped.
- Perhaps these techniques could be used more successfully for predicting eval performance based on the mix of pretraining data?
- Or predicting the effectiveness of post-training on downstream evals?

Thank You

- Any questions?

References I

Bowyer, Sam, Laurence Aitchison, and Desi R. Ivanova (Oct. 2025). "Position: Don't Use the CLT in LLM Evals With Fewer Than a Few Hundred Datapoints". en. In: *Proceedings of the 42nd International Conference on Machine Learning*. ISSN: 2640-3498. PMLR, pp. 81143–81184. URL: https://proceedings.mlr.press/v267/bowyer25a.html (visited on 11/09/2025).

Bowyer, Sam, Thomas Heap, and Laurence Aitchison (Sept. 2024). "Using Autodiff to Estimate Posterior Moments, Marginals and Samples". en. In: *Proceedings of the Fortieth Conference on Uncertainty in Artificial Intelligence*. ISSN: 2640-3498. PMLR, pp. 394–417. URL: https://proceedings.mlr.press/v244/bowyer24a.html (visited on 11/09/2025).

References II

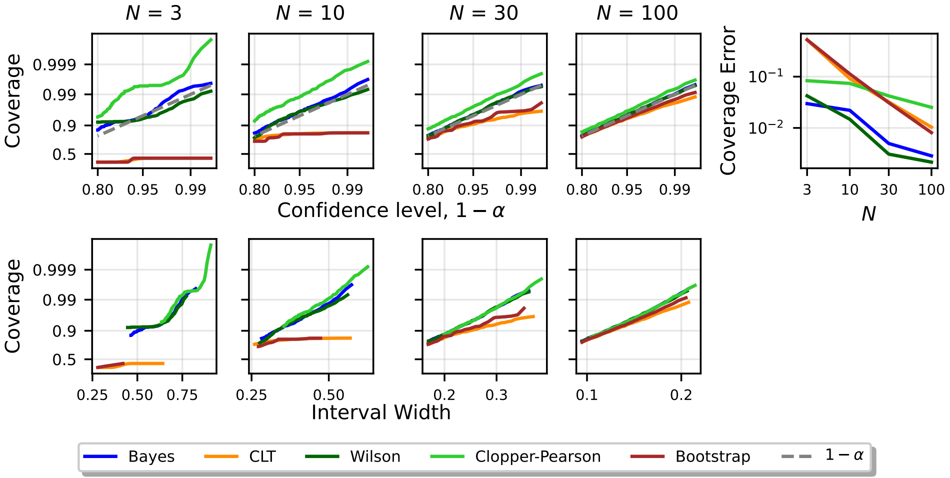
Chatterjee, Sourav and Persi Diaconis (Apr. 2018). "The sample size required in importance sampling". en. In: *The Annals of Applied Probability* 28.2. ISSN: 1050-5164. DOI: 10.1214/17-AAP1326. URL: https://projecteuclid.org/journals/annals-of-applied-probability/volume-28/issue-2/The-sample-size-required-in-importance-sampling/10.1214/17-AAP1326.full (visited on 02/03/2023).

Heap, Thomas, Sam Bowyer, and Laurence Aitchison (July 2025). "Massively Parallel Expectation Maximization For Approximate Posteriors". en. In: *Proceedings of the 7th Symposium on Advances in Approximate Bayesian Inference*. ISSN: 2640-3498. PMLR, pp. 25–66. URL: https://proceedings.mlr.press/v289/heap25a.html (visited on 11/09/2025).

Miller, Evan (Nov. 2024). *Adding Error Bars to Evals: A Statistical Approach to Language Model Evaluations*. en. arXiv:2411.00640 [stat]. DOI: 10.48550/arXiv.2411.00640. URL: http://arxiv.org/abs/2411.00640 (visited on 12/17/2024).

IID Questions Setting: Prior Mismatch

Use $\theta \sim \text{Uniform}[0,1]$ as the prior, but $\theta \sim \text{Beta}(100,20)$ as the true data distribution. ($\mathbb{E}[\theta] = 0.83$ and $\text{Var}(\theta) = 0.034^2$.)



Bayesian Evals for Interpretability

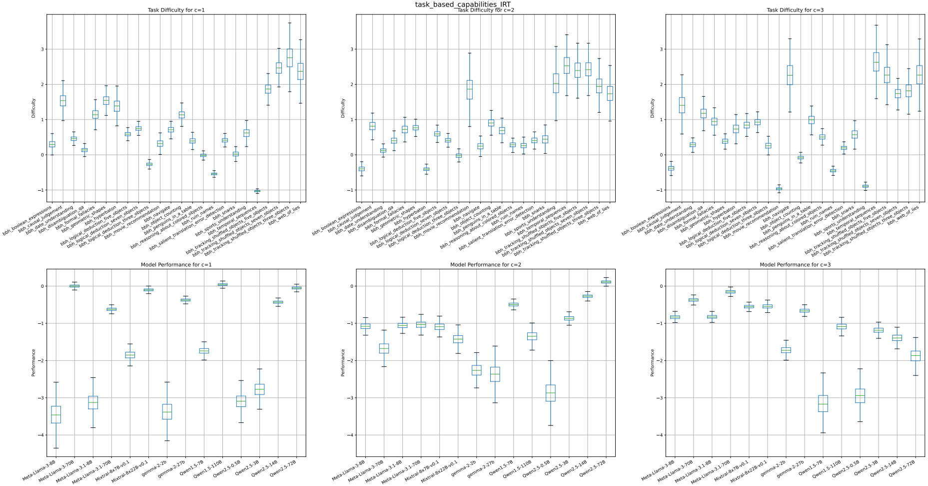
- Given eval questions and reponses, use an SAE-like model to enforce sparsity on features across the latent dimensions of ‘model performance’ and ‘benchmark difficulty’ vectors.

Bayesian Evals for Interpretability

- Given eval questions and reponses, use an SAE-like model to enforce sparsity on features across the latent dimensions of 'model performance' and 'benchmark difficulty' vectors.

Bayesian Evals for Interpretability

- Given eval questions and reponses, use an SAE-like model to enforce sparsity on features across the latent dimensions of ‘model performance’ and ‘benchmark difficulty’ vectors.



Clustered Questions Setting: Bayesian Implementation

Snippet 4: Bayesian analysis for clustered evals

```python
# S_t, N_t: np.arrays of length T with total
# sucesses & questions per task
import numpy as np
from scipy.stats import betabinom

# 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 = 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])
```