Frequentist Uncertainties on Density Ratios with $w_i f_i$ Ensembles

Sean Benevedes

ML4Jets2024 - November 6 2024

Work in collaboration with Jesse Thaler

Organization

Most talks	This talk
1. Introduction	1. Introduction
2. Methods	2. Results
3. Results	3. Methods
4. Discussion	4. Discussion

Density ratio estimation

What: Given samples from two distributions $p(x)$ and $q(x)$, estimate $\log r(x) \equiv \log \frac{q(x)}{p(x)}$

Why:	Parameter estimation	Reweighting	...
	$\text{argmax}_\theta \log \frac{p(\text{jet} \mid \theta)}{p(\text{jet})}$ (this talk)	$\langle \mathcal{O}(x) \rangle_q = \left\langle \frac{q(x)}{p(x)} \mathcal{O}(x) \right\rangle_p$	

How: Machine learning!

- Neyman-Pearson Lemma: The best classifier between q and p is given by any monotonic function of r
- Idea: Train a classifier on samples from q and p, solve for r (see e.g. Cranmer et al., 1506.02169)

But… how do we use a point estimate for r? If the estimate is poor, downstream applications will suffer. Need a measure of uncertainty in the estimate!

See Hermans et al. 2110.06581 for more on problems caused by neglecting these uncertainties

$w_i f_i$ ensembles

$$\boxed{\log r(x \mid \{w_i\}) = w_i f_i(x)}$$

where w_i are scalar weights and $f_i(x)$ are frozen networks

Pros	Cons
• Fully frequentist, no prior dependence • Intuitive: just like nuisance parameters! • Computationally efficient; no bootstrapping	• Does not assess uncertainty from training of the f_i

$w_i f_i$ ensembles

$$\boxed{\log r(x \mid \{w_i\}) = w_i f_i(x)}$$

where w_i are scalar weights and $f_i(x)$ are frozen networks

Pros	Cons
• Fully frequentist, no prior dependence • Intuitive: just like nuisance parameters! • Computationally efficient; no bootstrapping	• Does not assess uncertainty from training of the f_i Potentially a big problem!

Where do these fit in?

The lay of the (frequentist) land

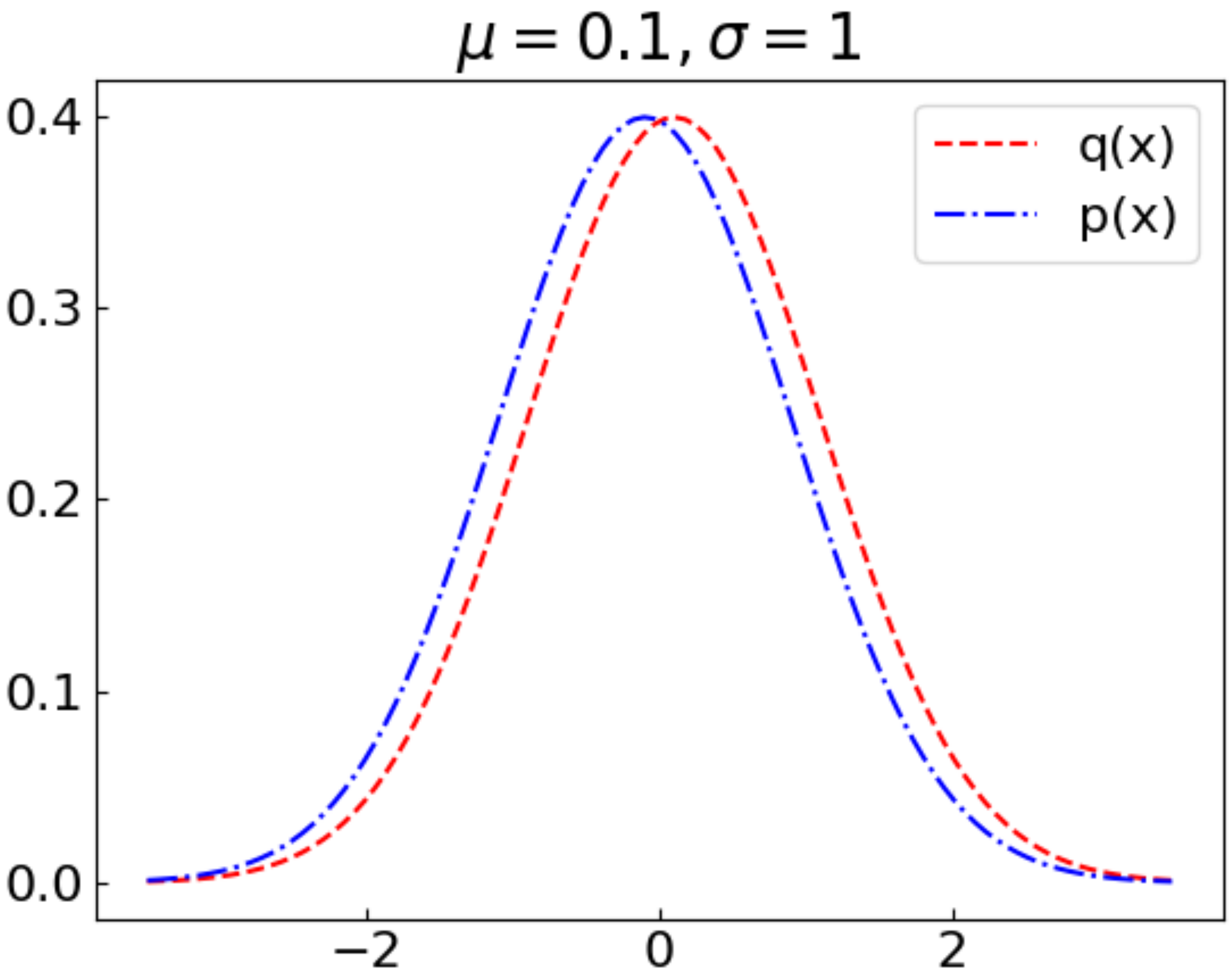
	Bootstrap	No bootstrap
Doesn't include training uncertainties	Pareto excluded	KLIEP (Sugiyama et al. 2008) $w_i f_i$ ensembles
Includes training uncertainties	Neyman construction (e.g. https://cds.cern.ch/record/2915316)	Hopefully next year's talk!

The big surprise: for the problems we’ve checked, this actually works!

Sean Benevedes

1D toy problem: Gaussians

- Let q be $\mathcal{N}(\mu, \sigma^2)$ and p be $\mathcal{N}(-\mu, \sigma^2)$, then $\log r = 2\mu x/\sigma^2$
- Evaluation:
  - Directly check coverage of r with pseudoexperiments
  - "Mixture fraction task" gives us a measure of performance in downstream task



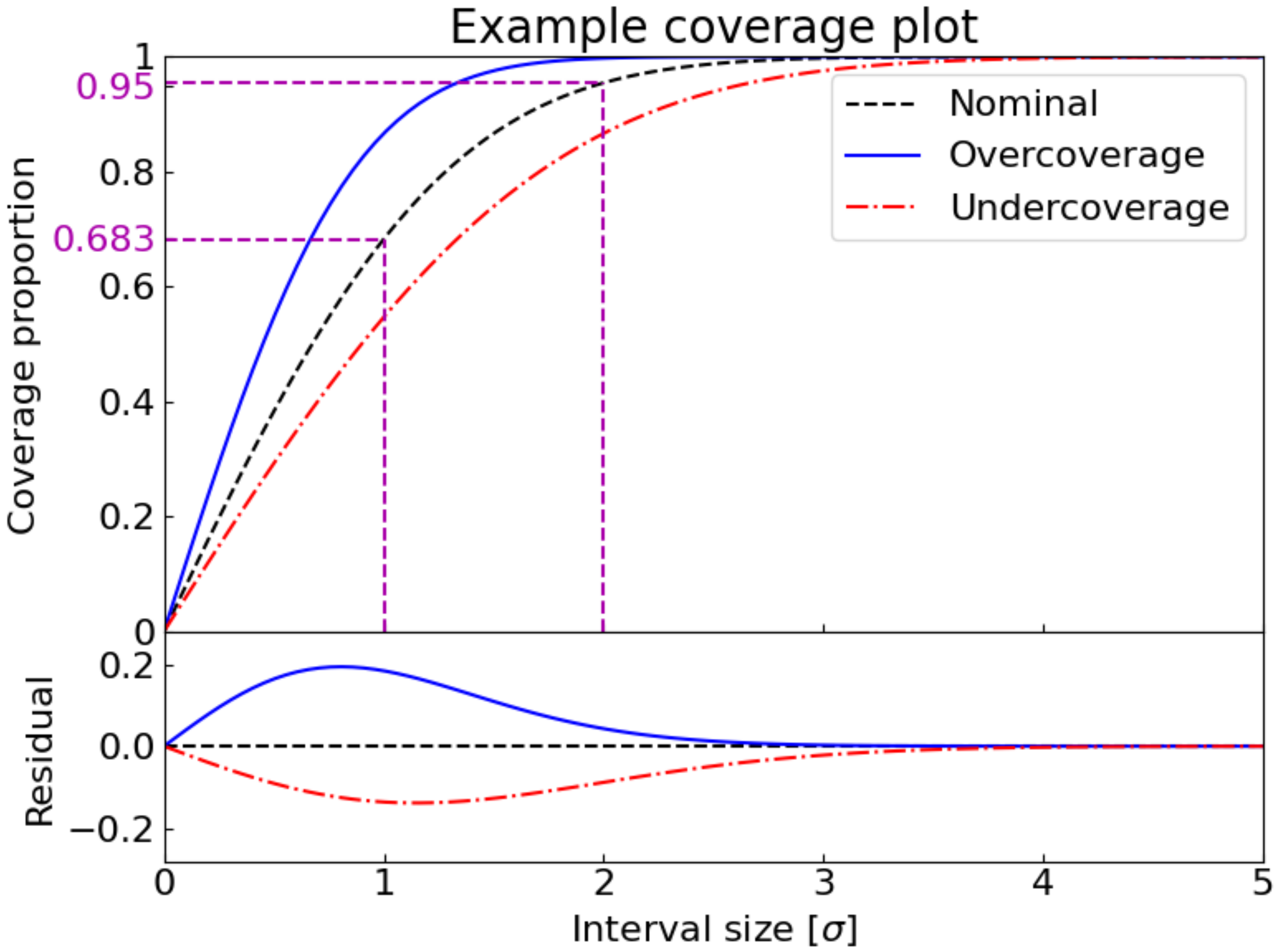
Mixture fraction task

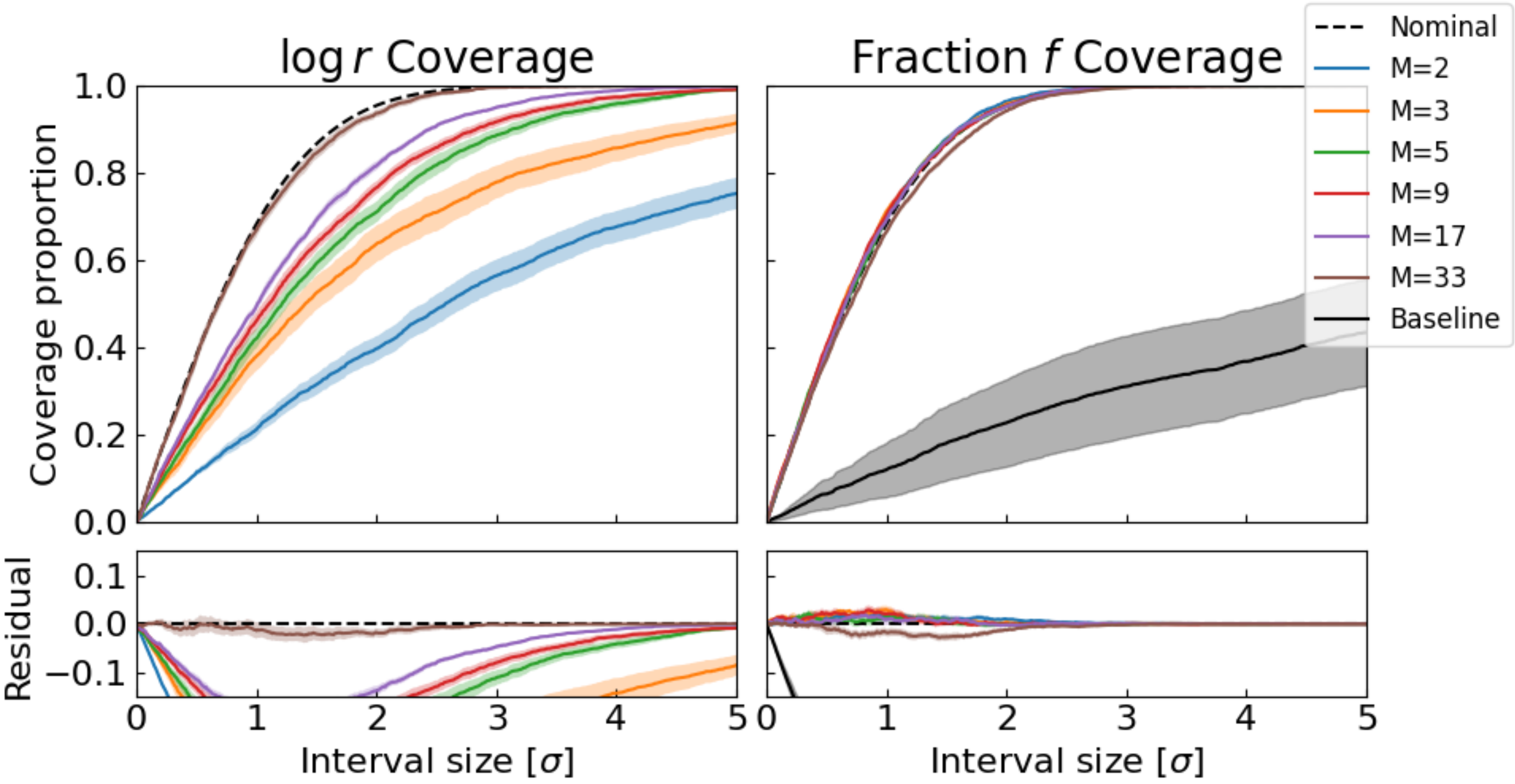
Estimate the parameter f given a set of samples D drawn i.i.d. from

$$p(x \mid f) = fq(x) + (1 - f)p(x)$$

$r(x)$ suffices to estimate f with maximum likelihood estimation:

$$\hat{f} = \text{argmax}_f \sum_\alpha \log \frac{p(x_\alpha \mid f)}{p(x_\alpha)} = \text{argmax}_f \sum_\alpha \log(fr(x_\alpha) + (1 - f))$$


Example coverage plot
Coverage proportion
Residual
Interval size [σ]
Nominal
Overcoverage
Undercoverage
0.95
0.683
1
0.8
0.6
0.4
0.2
0
0.2
0.0
−0.2
0
1
2
3
4
5

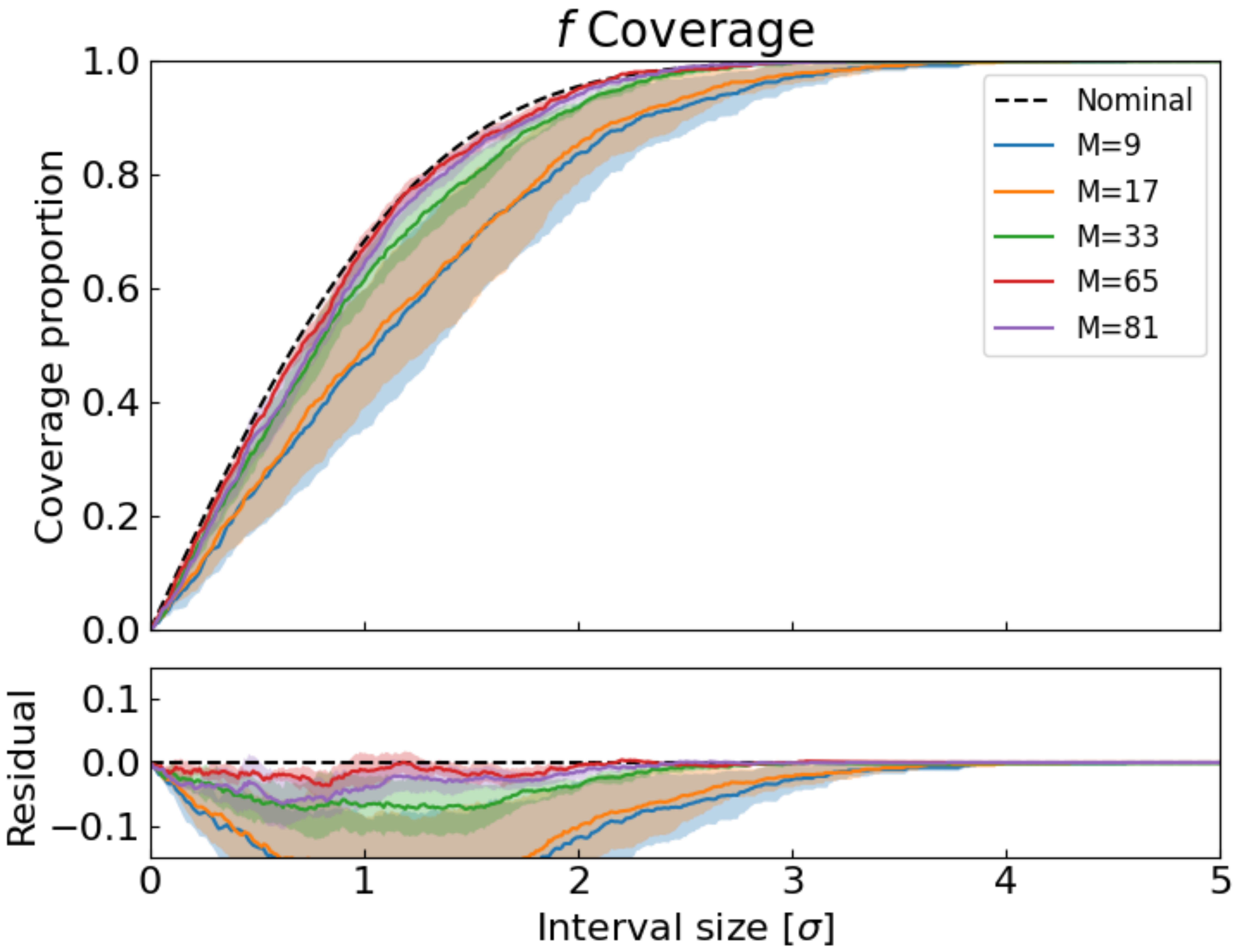


$N_{\text{train}} = 25{,}000$, $N_{\text{trainings}} = 10$, $N_{\text{samp}} = 300$, f scanned

ML4Jets: Quark/gluon likelihood

- Now, $q \sim$ Pythia 8.226 quark jets, $p \sim$ Pythia 8.226 gluon jets
  - We use the dataset included in the EnergyFlow package (you can find the jets here: https://zenodo.org/records/3164691)
- Since I don't know the exact answer for the likelihood ratio (let me know if you do!), we only report coverage for the mixture fraction task

Technical detail: We construct an IRC-safe ensemble with energy-flow networks, so the objective function is actually the projection of $\log r$ to an IRC-safe observable



$N_{\text{train}} = 20{,}000, N_{\text{trainings}} = 3, N_{\text{samp}} = 300, f = 0.1$

Sean Benevedes

Ok, what did we actually do?

$w_i f_i$ ensembles: the recipe

Ingredients

- M randomly initialized networks
- A training dataset with N_{train} samples each from q and p
- (Optional) A dataset of size N_{test} for downstream applications

1. Train the f_i on the training dataset
2. Find $\hat{w}_i$ using the MLC loss (introduced in Tito D'Agnolo et al. 1806.02350)
3. Compute $C_{ij} \equiv \text{Cov}(\hat{w}_i, \hat{w}_j)$ with analytic formulae
4. Propagate uncertainties for downstream tasks

Training the f_i

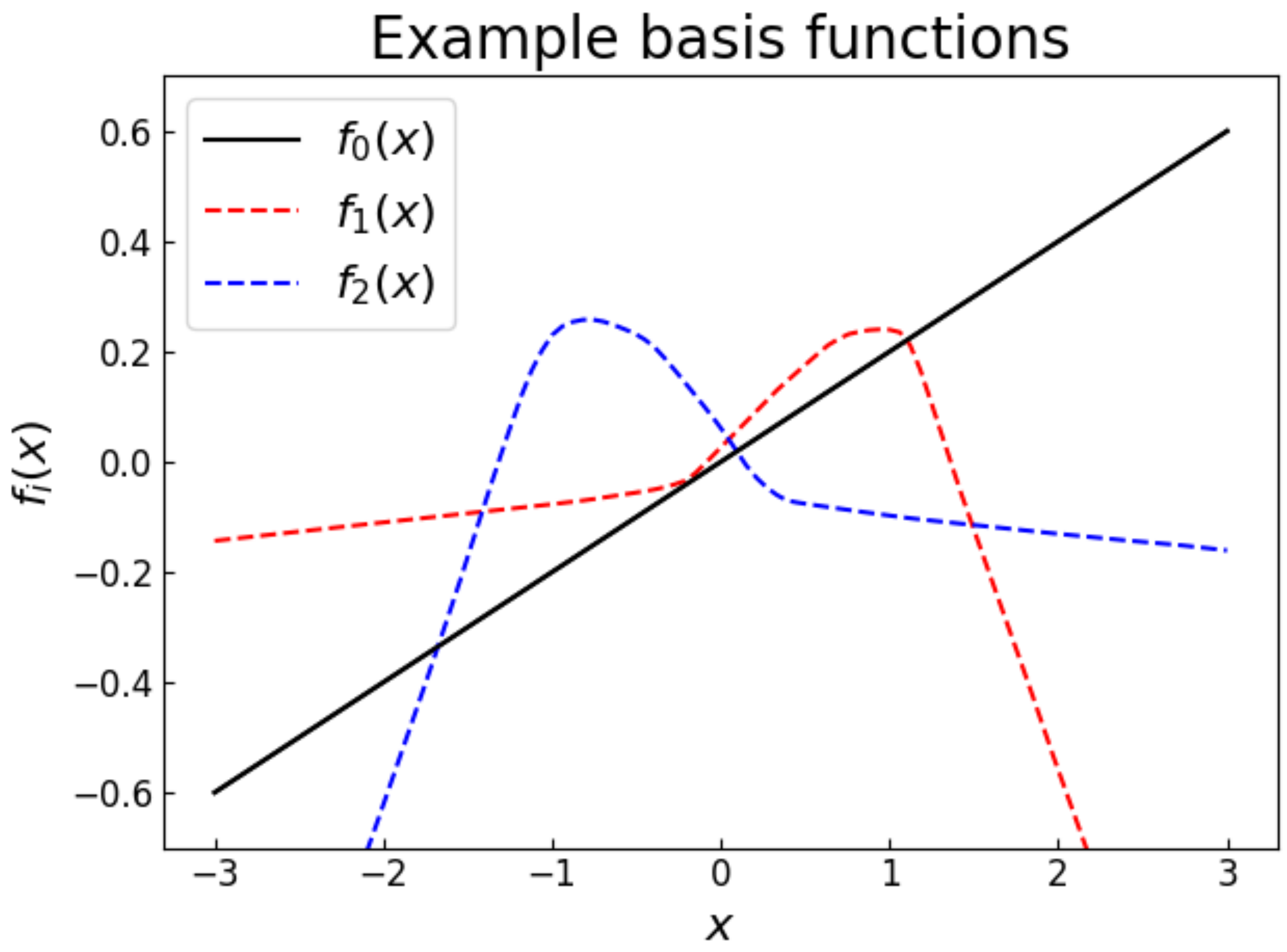
- Objective: Train f_i such that there is some set of w_i for which $w_i f_i$ is a good model of $\log r$… that leaves a lot of freedom!
- (At least) two families of approaches:
    1. Train one distinguished f_0 to model $\log r$ as well as possible (just like before!), use the remaining f_i to model the residuals
        - "$w_i = (1,0,0,0,...)_i$"
    2. Train the f_i on equal footing (conventional ensembling)
        - "$w_i = (1/M, 1/M, \ldots)_i$"

Training the f_i

- Objective: Train f_i such that there is some set of w_i for which $w_i f_i$ is a good model of $\log r$… that leaves a lot of freedom!
- (At least) two families of approaches:
  1. Train one distinguished f_0 to model $\log r$ as well as possible (just like before!), use the remaining f_i to model the residuals
     - "$w_i = (1,0,0,0,...)_i$"
  2. Train the f_i on equal footing (conventional ensembling)
     - "$w_i = (1/M, 1/M, \ldots)_i$"

Training the f_i

1. Train f_0 as a classifier between q and p (using e.g. the MLC loss)
2. Train $\{f_1, f_2, \ldots, f_{M-1}\}$ to be uncorrelated with each other and f_0, but not as classifiers (ask me why!)



Fitting the w_i

Now, find the $\hat{w}_i$ minimizing the symmetrized MLC loss:

$$\mathscr{L}(w) = -\left(\sum_{x_\alpha \sim q}^{N_{train}} [w_i f_i(x_\alpha) - (e^{-w_i f_i(x_\alpha)} - 1)] + \sum_{x_\alpha \sim p}^{N_{train}} [-w_i f_i(x_\alpha) - (e^{w_i f_i(x_\alpha)} - 1)]\right)$$

Estimators of this form are called *M-estimators* (Hubert 1964) in the statistics literature, and their covariances are known!

$$\boxed{C_{ij} = V_{ik}^{-1} U_{kl} V_{lj}^{-1}} \quad \left(\text{where } V_{ik} \equiv \mathrm{Cov}\left(\frac{\partial \mathscr{L}}{\partial w_i}, \frac{\partial \mathscr{L}}{\partial w_j}\right), U_{kl} \equiv \frac{\partial^2 \mathscr{L}}{\partial w_i \partial w_j}\right)$$

Sean Benevedes

Uncertainty propagation

- As an example of a downstream application, consider the mixture fraction task

(Reminder, given r: $\hat{f} = \text{argmax}_f \sum_{\alpha}^{N_{test}} \log(fr(x_\alpha) + (1-f))$)

- Gong-Samaniego theorem (Gong and Samaniego 1981): With a consistent estimator of the $\hat{w}_i$, and therefore of $\log r$ (under the well-specified assumption), the estimator for f that we get by just plugging $\hat{r}$ into the MLE is asymptotically consistent with known variance!

$$\boxed{\sigma^2_{\hat{f}} = \sigma^2_{\hat{f},\text{MLE}}\left(1 + A_i C_{ij} A_j\right), A_i \equiv \frac{\partial \mathscr{L}}{\partial w_i \partial f}}$$

That’s it!

$w_i f_i$ ensembles: the recipe

Ingredients

- M randomly initialized networks
- A training dataset with N_{train} samples each from q and p
- (Optional) A dataset of size N_{test} for downstream applications

1. Train the f_i on the training dataset
2. Find $\hat{w}_i$ using the MLC loss (introduced in Tito D'Agnolo et al. 1806.02350)
3. Compute $C_{ij} \equiv \text{Cov}(\hat{w}_i, \hat{w}_j)$ with analytic formulae (M-estimator)
4. Propagate uncertainties for downstream tasks (e.g. GS)

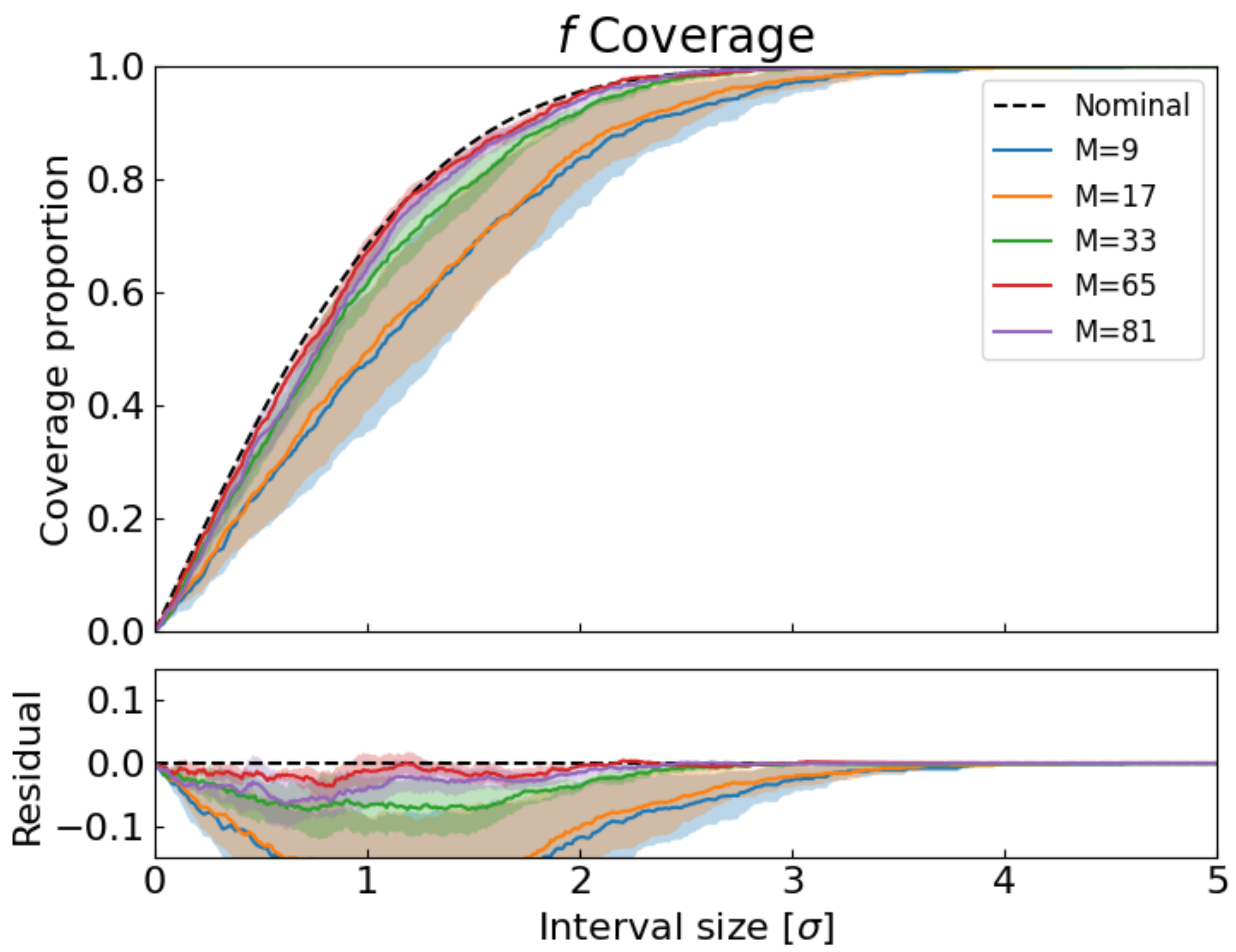
$w_i f_i$ ensembles

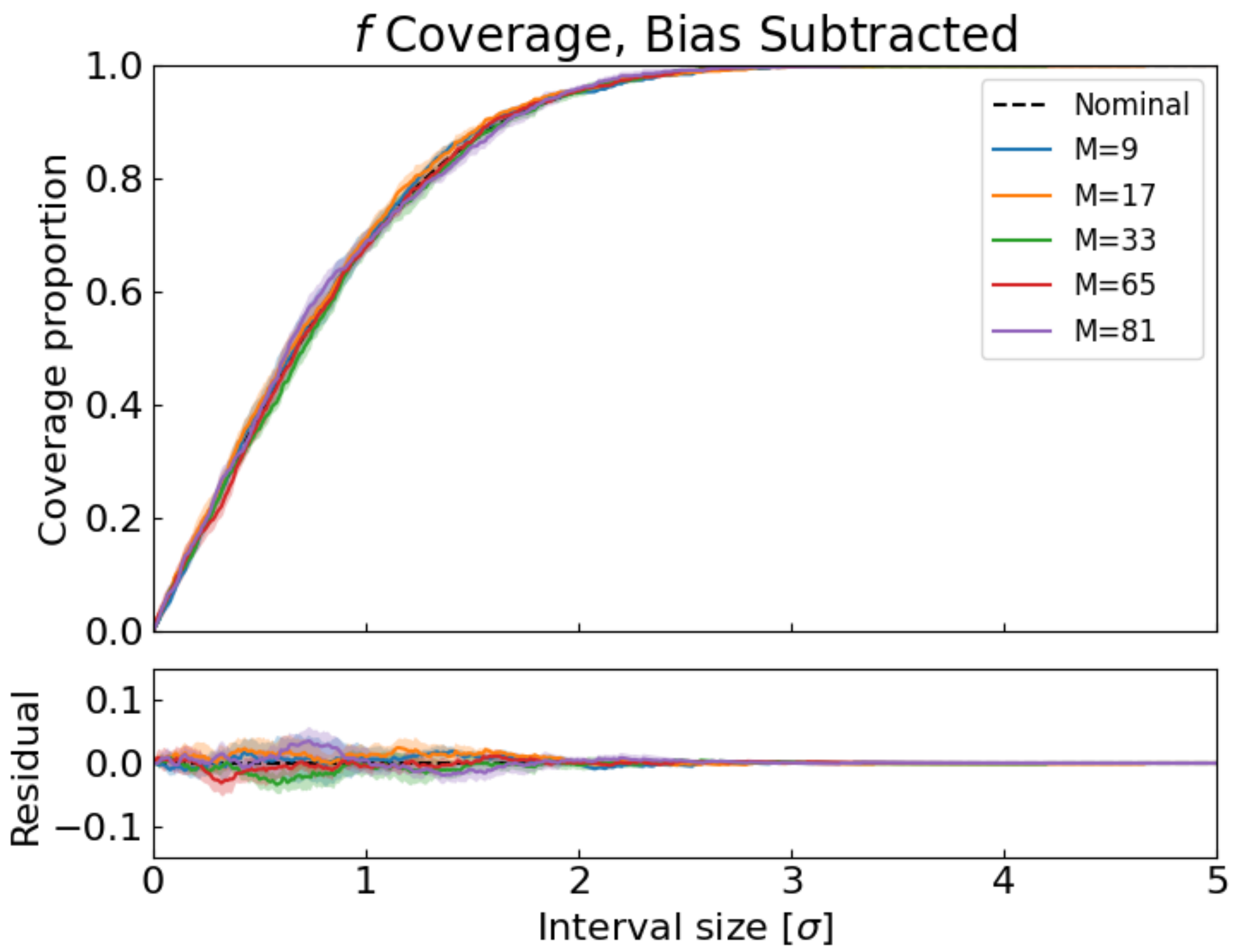
$$\boxed{\log r(x \mid \{w_i\}) = w_i f_i(x)}$$

Pros	Cons
• Fully frequentist, no prior dependence • Intuitive: just like nuisance parameters! • Computationally efficient; no bootstrapping	• Does not assess uncertainty from training of the f_i Maybe not a big problem!

Backup slide: Bias

- Two possible causes of bias in inference:
  1. Model misspecification (M too small/networks poorly trained)
  2. Breakdown of asymptotic regime (N too small and/or M too large)





Sean Benevedes

Backup slide: Why decorrelation?

- Better reason: In vanilla ensembles, uncorrelated ensemble members decrease variance for prediction (see e.g. Mehta et al. 2019); no reason to expect different behavior here
- Worse reason: Numerics! Minimization of the symMLC loss is way easier when covariance is diagonal

$$\frac{\partial \mathscr{L}}{\partial w_i \partial w_j} \propto \mathrm{Cov}\left(f_i, f_j\right)$$