Scaling BART

Antonio R. Linero, Guoqing Zhang

The Basic Question

- BART models are successful partially because of their *unreasonably effective defaults.*

- **But how universal are these defaults?**
  - How should the number of trees scale with sample size?
  - What about the number of predictors?

- **Cross validation is expensive, especially for large samples! Would be good to have a shortcut that gives us (approximately) the right answer.**

Sidequest: How Fast can a BART Be?

To answer the questions we are interested in, we need a *fast* version of BART.

BART is already pretty fast, but to understand how things should scale we might need to be able to accomodate

- **Millions of observations**
- **Hundreds of thousands of trees**

Implementations of BART in R are not adequate for this purpose.

Sidequest: How Fast can BART Be?

Very Fast Bayesian Additive Regression Trees on GPU

Giacomo Petrillo [1]

Abstract

Bayesian Additive Regression Trees (BART) is a nonparametric Bayesian regression technique based on an ensemble of decision trees. It is part of the toolbox of many statisticians. The overall statistical quality of the regression is typically higher than other generic alternatives, and it requires less manual tuning, making it a good default choice. However, it is a niche method compared to its natural competitor `XGBoost`, due to the longer running time, making sample sizes above 10 000–100 000 a nuisance. I present a GPU-enabled implementation of BART, faster by up to 200x relative to a single CPU core, making BART competitive in running time with `XGBoost`. This implementation is available in the Python package `bartz`.

BART vs. XGBoost According to the Kaggle (2021, p. 35) survey, the most popular regression method in the class of BART is the variant of gradient boosting implemented in the software `XGBoost` (Chen & Guestrin, 2016), used by half of the surveyed data scientists. Linero & Yang (2018, table 2, p. 1105) show that BART predicts better than `XGBoost` on average on a set of benchmark datasets (with both methods cross validated). The other advantages of BART over `XGBoost` are that, due to being Bayesian, it provides full uncertainty quantification out of the box, a principled way to extend the model for special settings, and good results even without cross validation. However, BART is much slower than `XGBoost`, making it impractical for large datasets: the largest BART analysis I know of had $n \approx 500\,000$ observations/examples and $p = 90$ predictors/features (Pratola et al., 2020), while `XGBoost` boasts "billions of examples" on its home page.

Really fast!

Sidequest: How Fast can BART Be?

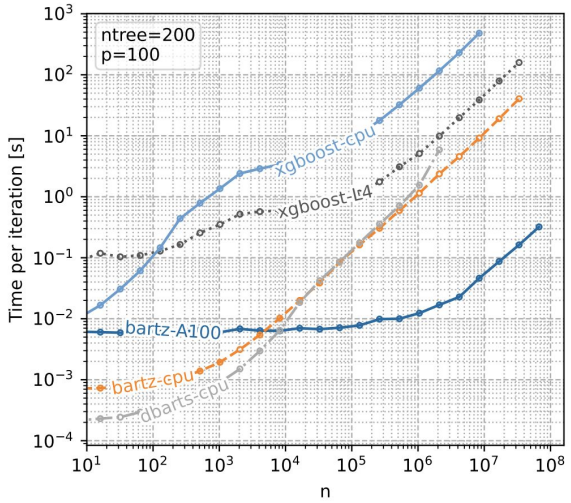
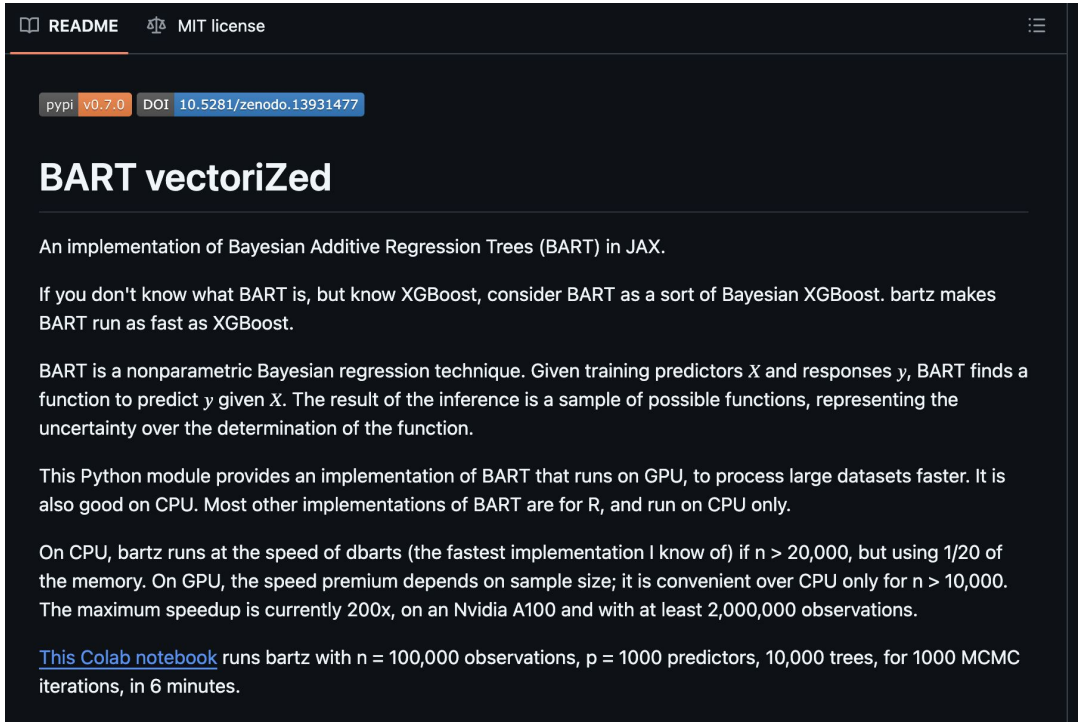


Figure 3. Time to run an iteration of the algorithm vs. training set size n, comparing `bartz` with the fastest BART implementation (`dbarts`) and `XGBoost`, on CPU (`-cpu`) and GPU (`-A100`, `-L4`). Keep into account that `XGBoost` requires only one iteration to produce a usable result, while BART requires $O(1000)$ iterations.

Really fast!

But only regression...

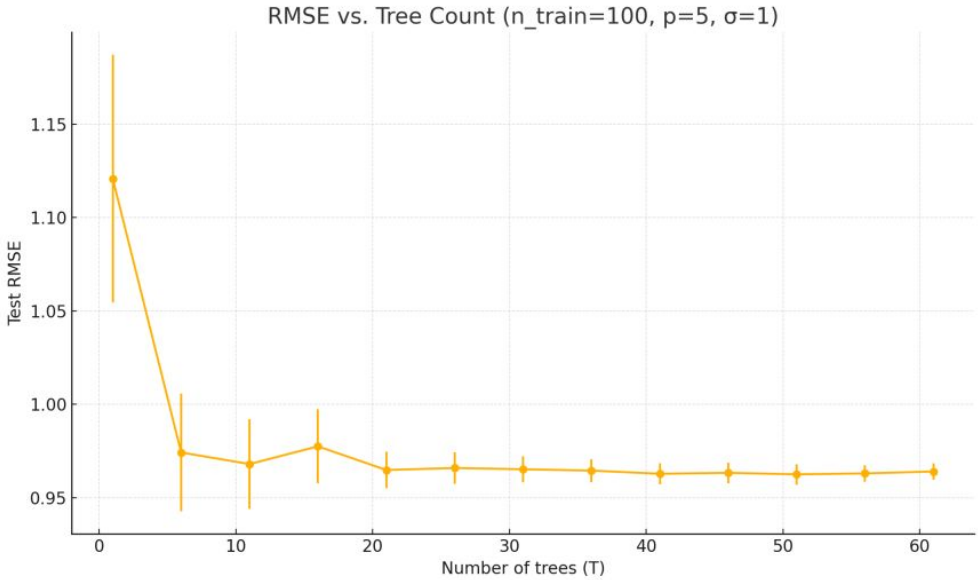
Sidequest: How Fast can BART Be?



https://github.com/Gattocrucco/bartz

Claim: It Is Usually OK To “Overshoot” The Number of Trees

Simulation Example (N = 100, P = 5, σ = 1)



RMSE curve across tree counts

Theoretical Guard: Gaussian Process Regression Limit

Suppose $r_T \sim$ BART with T trees and with leaf node prior

$$\mu_{t\ell} \sim \text{Normal}(0, \sigma_\mu^2/T)$$

Then, as $T \to \infty$, we have

$$r_T(\cdot) \to \text{Gaussian Process}(0, \kappa) \quad \text{where}$$
$$\kappa(x, x') = \Pr(x \text{ and } x' \text{ are associated to same leaf a-priori})$$

Gaussian Process Regression (Rasmussen and Williams)

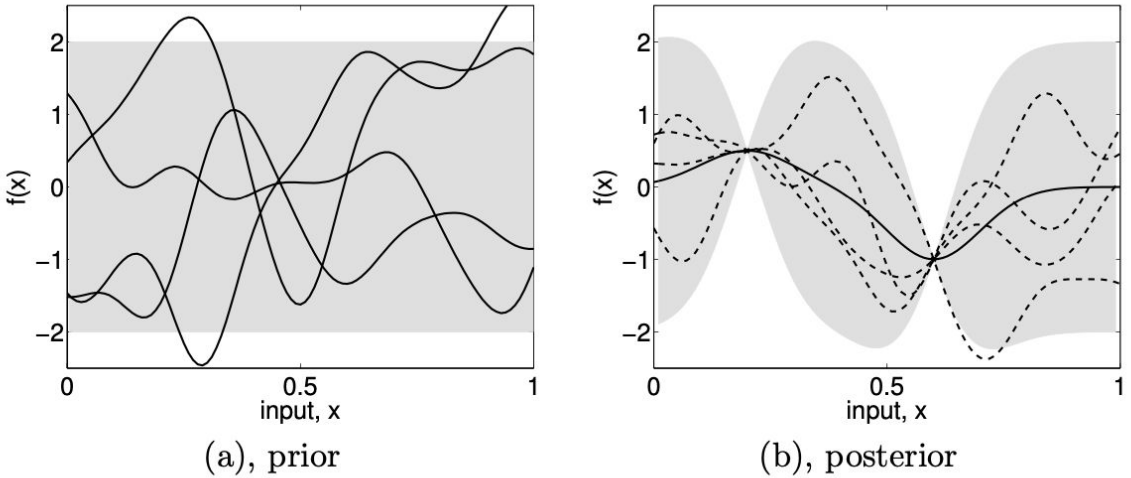


Figure 1.1: Panel (a) shows four samples drawn from the prior distribution. Panel (b) shows the situation after two datapoints have been observed. The mean prediction is shown as the solid line and four samples from the posterior are shown as dashed lines. In both plots the shaded region denotes twice the standard deviation at each input value x.

GPs are *priors on functions.* Commonly used alternative to BART with lots of nice theoretical properties.

Lesson: it is *really* hard to put in too many trees

Limiting Covariance Function (Petrillo, 2025)

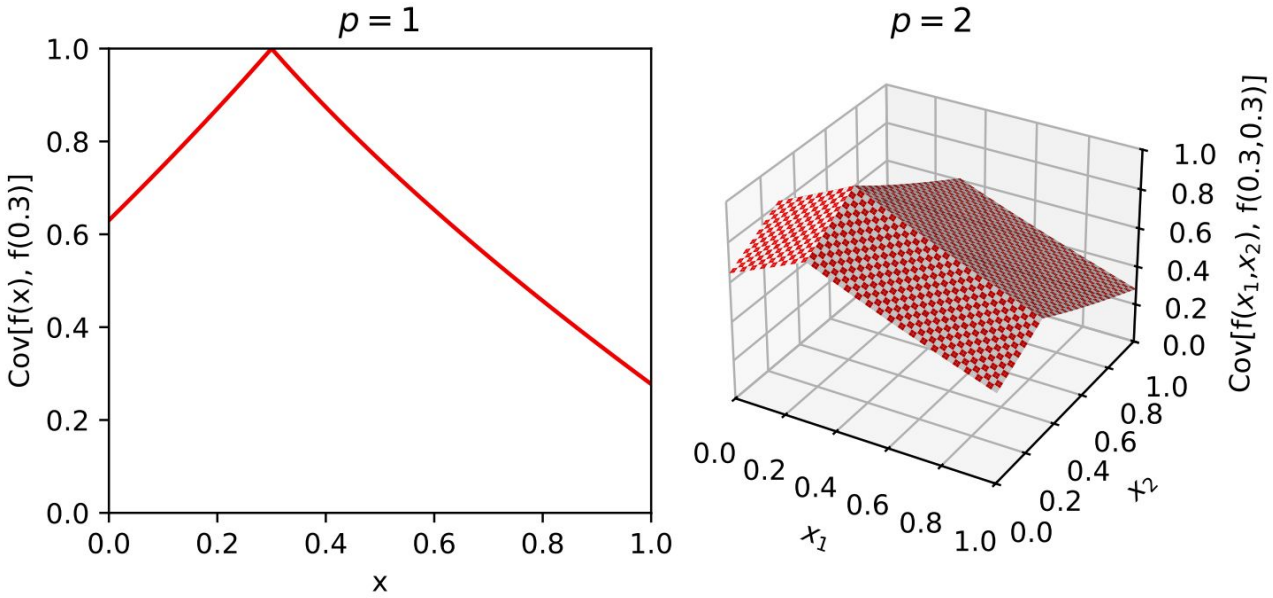


Figure 3: Section of the BART correlation function, with $\alpha = 0.95$, $\beta = 2$, and $n = 100$ evenly spaced splitting points between 0 and 1.

Linero (2017) argues this is almost a Laplace kernel, Petrillo gives the exact form for default BART

Claim: The Optimal Number of Trees Grows Slowly With the Sample Size

Theoretical Background

We are not flying blind! Results of Rockova and van der Paas suggest that:

- We should scale $T = Cn\,\epsilon_{sn}^2$ for some constant C.
- The constant C should grow with the number of relevant interactions and the signal to noise ratio.

Is this true in practice?

Some Possible Rates

Maximal Interaction	Rate (BART)	Rate (Soft BART)
1	$N^{1/3}$	$N^{1/5}$
2	$N^{1/2}$	$N^{1/3}$
3	$N^{3/5}$	$N^{3/7}$

But the constants matter!

Good Rates (Linero and Yang, 2018)

Theorem 4 (Posterior convergence rate for additive sparse truth). *Suppose that Assumptions G and P are satisfied. Let* $f_0 = \sum_{v=1}^{V} f_{0,v}$, *where the* v*th additive component* $f_{0,v}$ *belongs to* $\mathcal{C}^{\alpha_v, R}([0,1]^p)$, *and is bounded and only depends on at most* d_v *covariates for* $v = 1, \ldots, V$. *If* $\sum_{v=1}^{V} d_v \log p/n \to 0$, *then for all sufficiently large constant* $M > 0$, *we have*

$$\Pi_{n,\eta}\left[\|f - f_0\|_n \geq M\,\varepsilon_n\right] \to 0, \quad \textit{in probability} \quad \textit{as } n \to \infty,$$

where $\varepsilon_n = \sum_{v=1}^{V} n^{-\alpha_v/(2\alpha_v + d_v)} (\log n)^t + \sum_{v=1}^{V} \sqrt{n^{-1}\, d_v\, \log p}$ *for any* $t \geq \max_v \alpha_v(d_v+1)/(2\alpha_v + d_v)$.

(See also Rockova and van der Paas, 2020)

Friedman Simulation Experiment

Set

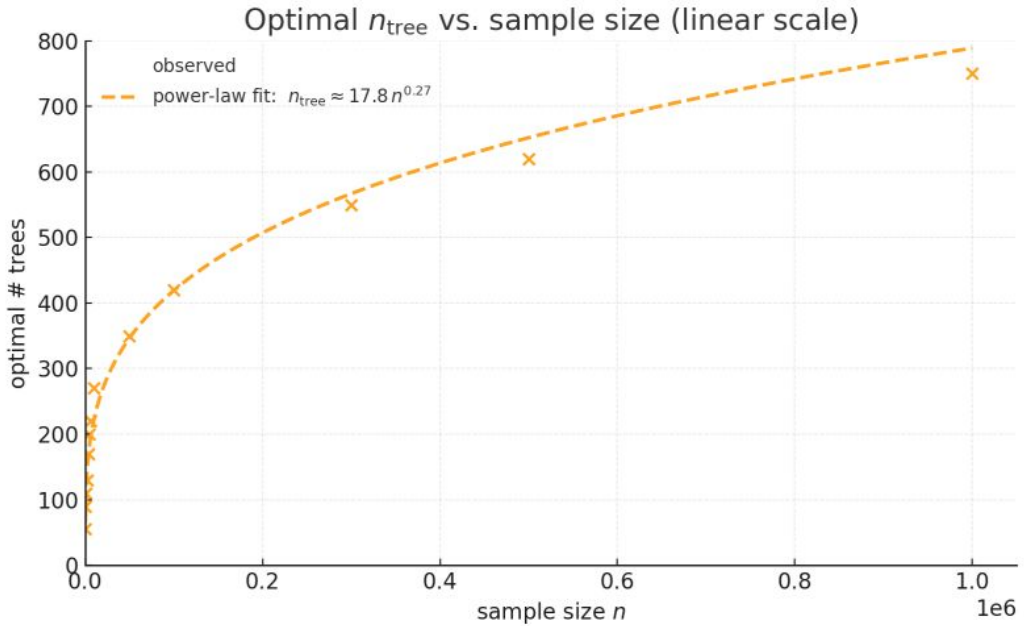
$$Y_i = 10 \sin(\pi X_{i1} X_{i2}) + 20(X_{i3} - 0.5)^2 + 10 X_{i4} + 5 X_{i5} + \epsilon_i$$

How does the optimal number of trees scale with N?

Scaling of Optimal Number of Trees

Friedman #1 ($p = 5,\ \sigma = 1$)

Empirical fit: $n^{\star}_{\text{tree}} \approx 17.8\, n^{0.27}$



Claim: The Optimal Number of Trees Grows Slightly Faster in the Predictors

Empirical Estimation of Rates

Is it possible **estimate the optimal number of trees, or rate, empirically?**

$$\log \mathrm{RMSE} = f(s, T, N, P) + \epsilon$$

where s is the signal-to-noise ratio, T is the number of trees, N is the sample size, P is the number of predictors.

Setting for Evaluating Different P

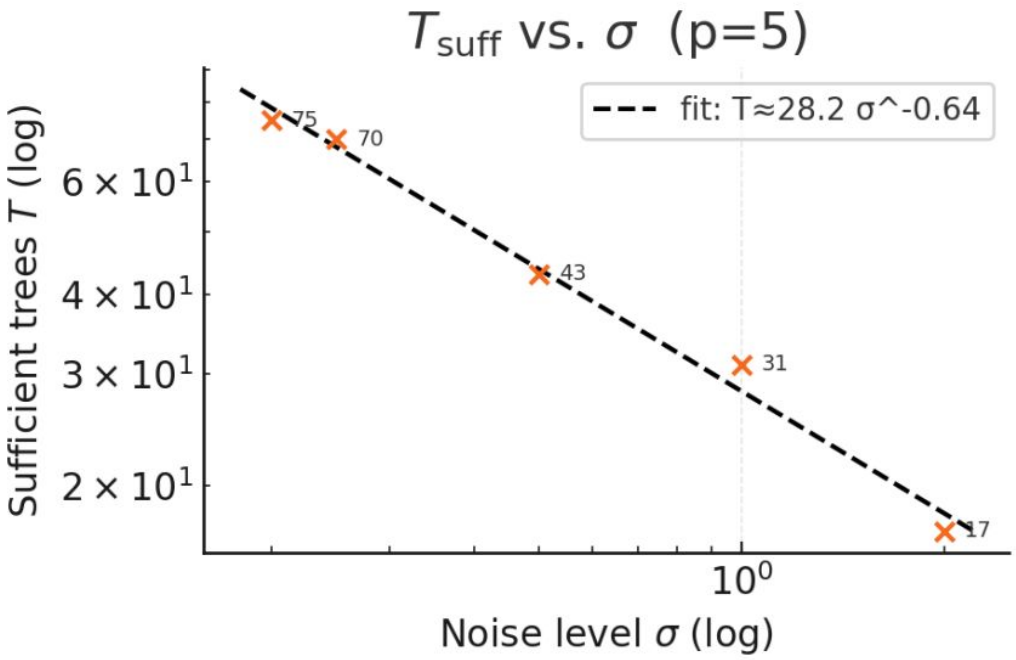
- **Data-Generating Process**
  - $f(x) = \beta \sum_{j=1}^{p} x_j,\ x_j \sim \mathrm{U}(0,1)$
  - Choose $\beta = \sqrt{12/p}$ so that $\mathrm{Var}\{f(X)\} = 1$.
  - Add Gaussian noise $\varepsilon \sim \mathcal{N}(0, \sigma^2)$; here $\sigma = 1 \Rightarrow \mathrm{SNR} = 1/\sigma^2 = 1$.
- **Simulation parameters**
  - $n_{\text{train}} = 100,\ n_{\text{test}} = 20$
  - $p = 5$ (five additive components)
  - 30 independent *gbart* fits per tree size (different random seeds)
  - MCMC: `nskip`=500, `ndpost`=5 000, `keepevery`=1
- **Grid of tree counts** $T \in \{1, 6, 11, \ldots, 196\}$.

Raw Data

```
> head(my data, n = 30)
      n p sigma   T rmse mean     rmse sd             source file
1  1000 5  0.25  20 0.1439928 0.006449053 rmse table all 1000.csv
2  1000 5  0.25  35 0.1297659 0.002342371 rmse table all 1000.csv
3  1000 5  0.25  50 0.1255966 0.005443801 rmse table all 1000.csv
4  1000 5  0.25  65 0.1197273 0.004341361 rmse table all 1000.csv
5  1000 5  0.25  80 0.1118308 0.003459003 rmse table all 1000.csv
6  1000 5  0.25  95 0.1170809 0.001104970 rmse table all 1000.csv
7  1000 5  0.25 110 0.1136197 0.006826672 rmse table all 1000.csv
8  1000 5  0.25 125 0.1130920 0.003284491 rmse table all 1000.csv
9  1000 5  0.25 140 0.1180638 0.002347378 rmse table all 1000.csv
10 1000 5  0.25 155 0.1167006 0.001382252 rmse table all 1000.csv
11 1000 5  0.25 170 0.1161516 0.002207389 rmse table all 1000.csv
12 1000 5  0.25 185 0.1181522 0.002197619 rmse table all 1000.csv
13 1000 5  0.50  10 0.2152712 0.018816177 rmse table all 1000.csv
14 1000 5  0.50  15 0.1972765 0.011581074 rmse table all 1000.csv
15 1000 5  0.50  20 0.1866483 0.006379539 rmse table all 1000.csv
16 1000 5  0.50  25 0.1736538 0.007501478 rmse table all 1000.csv
17 1000 5  0.50  30 0.1662500 0.006536652 rmse table all 1000.csv
18 1000 5  0.50  35 0.1587621 0.007332831 rmse table all 1000.csv
19 1000 5  0.50  40 0.1570417 0.006230706 rmse table all 1000.csv
20 1000 5  0.50  45 0.1541554 0.004533518 rmse_table_all_1000.csv
```

etc.

Signal to Noise Ratio



Setup: $n = 200,\ p = 5$

Fit:

$$T_{\text{suff}} \approx 28.2 \cdot \sigma^{-0.64}$$

or $\quad T \propto \text{SNR}^{0.32}$

Recommendation: Assume Small/Moderate SNR

Maybe (?) Useful Laws

$$P_{np}(x,y) = \mathbf{2.5243} - 0.2751x + 1.2172y + 0.03660x^2 - 0.01083y^2 - 0.05439xy,$$
$$Q_{n\sigma}(x,z) = 0.01801z - 0.07402z^2 - 0.08847xz$$

Final predictor: $\hat{T} = \exp\big(P_{np}(x,y) + Q_{n\sigma}(x,z)\big)$.

Estimation. Ordinary Least Squares (OLS) on the cleaned dataset (log-transformed; obvious outlier $(50000, 100, 1) \to 150$ removed).

Goodness of fit (log-scale). $R^2 \approx \mathbf{0.964}$.

rate $T \propto n^{0.372}\, p^{0.430}\, \sigma^{-0.584}$

Validation

p	σ	Observed T	Pred. $\hat{T}$	Rel. Error
5	0.25	150	134.2	10.6%
5	0.50	90	89.6	0.4%
5	1.00	50	55.8	11.6%
10	0.25	220	219.7	0.1%
10	0.50	140	146.8	4.9%
10	1.00	100	91.4	8.6%
20	0.25	320	356.1	11.3%
20	0.50	170	237.9	**40.0%**
20	1.00	120	148.1	**23.4%**
50	0.25	570	663.4	16.4%
50	0.50	360	443.3	**23.1%**
50	1.00	250	275.9	10.4%

Tends to be conservative

Summary. MAPE $\approx$ **13.4%** (median $\approx$ 10.9%). Largest errors at (p=20, σ=0.5/1) and (p=50, σ=0.5)

Summary and Future Work

- Results for scaling P assumed *additivity*. Really, should be number of main effects *and* interactions.
  - **No data for this, but we are working on it!**
- Overall messages:
  - **Gentle scaling in N and P**, but implications for both large are not great.
  - **Assume an SNR that is moderate** to upper-bound the number of trees.
  - **Hard to find situations that I work on where 1000 trees would be terrible.**