Predictable Noise and patterns from millions of questions

Sida Wang
Research Scientist, FAIR at Meta

Data contribution: Sean OBrien, Alex Gu, Lovish Madaan, Dieuwke Hupkes, Jiawei Liu, Yuxiang Wei, Naman Jain, Yuhang Lai, Sten Sootla

Outline

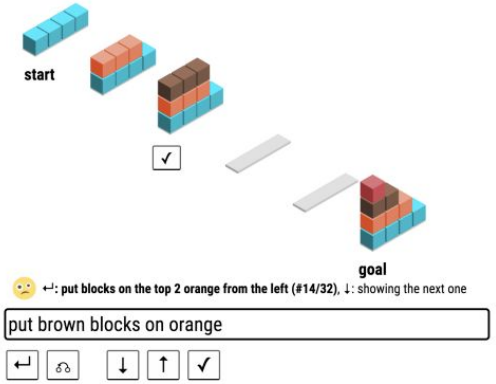
- Intro
- Motivation: benchmarks are getting harder and smaller
- Exploratory data analysis: What does the evaluation say beyond single accuracy
- Implications on the noise levels and why?
  - Recommendations about noise intervals
- How can we make this better + examples

Quick self-intro

Learning. When the human provides feedback in the form of a particular y, the system forms the following loss function:

$$\ell(\theta, x, y) = -\log p_\theta(y \mid x, s) + \lambda ||\theta||_1, \tag{4.2}$$

$$p_\theta(y \mid x, s) = \sum_{z:[\![z]\!]_s = y} p_\theta(z \mid x). \tag{4.3}$$



Monsters, Inc: initial – move forward – add green brown floor – add girl – go back and down – add d (some steps for moving are omitted)

2016: learn from human feedback using marginal likelihood, also originated here at UCB. an alternative to RL without emphasis on exploration

CWM: An Open-Weights LLM for Research on Code Generation with World Models: RL for SWE-bench, stabilizing RL

Code LLM: AST-FIM, SWE-RL, eval-arena, LEVER, Coder-reviewer, InCoder, MBR-exec.

Benchmarks: SWE-bench M, Spider 2.0, LiveCodeBench, SAFIM, CRUXEval, DS-1000

A bit of reflection on evaluation

Do 10% better on ImageNet, and no more doubts about deep learning

- necessary condition: no doubt about statistical significance

Do we similarly trust our current LLM evaluations?

Many papers report 2-10% improvements on HumanEval, or a few % on SWEbench

How can we interpret this?

- old answer is statistical testing: 2.5% on HumanEval is not stats significant

Test set sizes are now much smaller than MNIST

Deep learning was measured and validated by MNIST and ImageNet

Testset sizes: MNIST: **10k**, ImageNet: **100k**

LLM multiple choice: MMLU ~10k

Generative test-based eval (generate a short program)

HumanEval: 164, MBPP+: 378, CruxEval: 800, DS-1000: 1000, DMC: 165, LiveCodeBench ~1000, Alder: 225

Agent Evals (often need 100k+ tokens)

SweBench: 2.3k, -Verified: 500, -lite: 300, -multimodal: 517

T-Bench: 80

But each question is more informative!

- To answer each question, a lot of text/code is generated and evaluated by tests. Is this more informative than True/False and multiple choice?

```
def is_simple_power(x, n):
    """Your task is to write a function that returns true if a number x is a simple
    power of n and false in other cases.
    x is a simple power of n if n**int=x
    For example:
    is_simple_power(1, 4) => true ..
```

- For humans, you can ask a few good questions and get good information
  - IOI / IMO: 6 questions, Putnam: 12 questions
  - has an interesting range of difficulties, but then maybe humans are still fairly similar

Solve an open problem?

- solve even one major open problem is significant!
  - no one will object that a sample size of 1 is not statistically significant

Many daily agentic tasks are more economically important

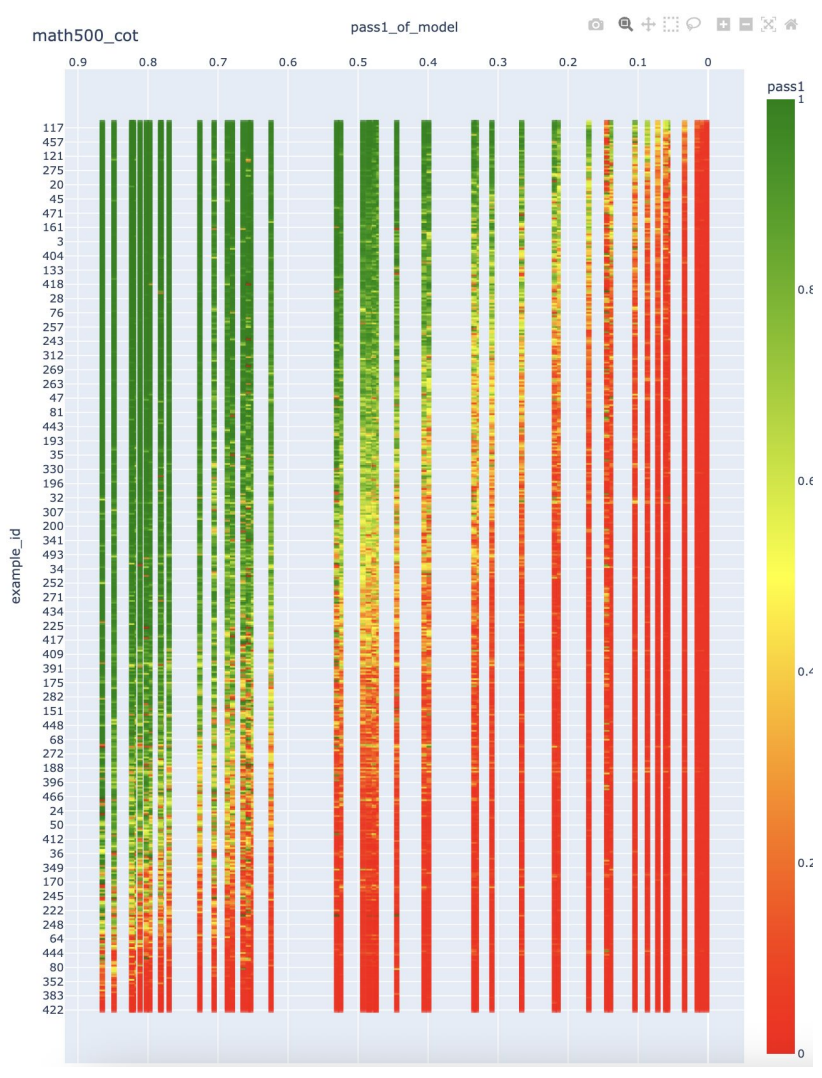
So what’s the case?

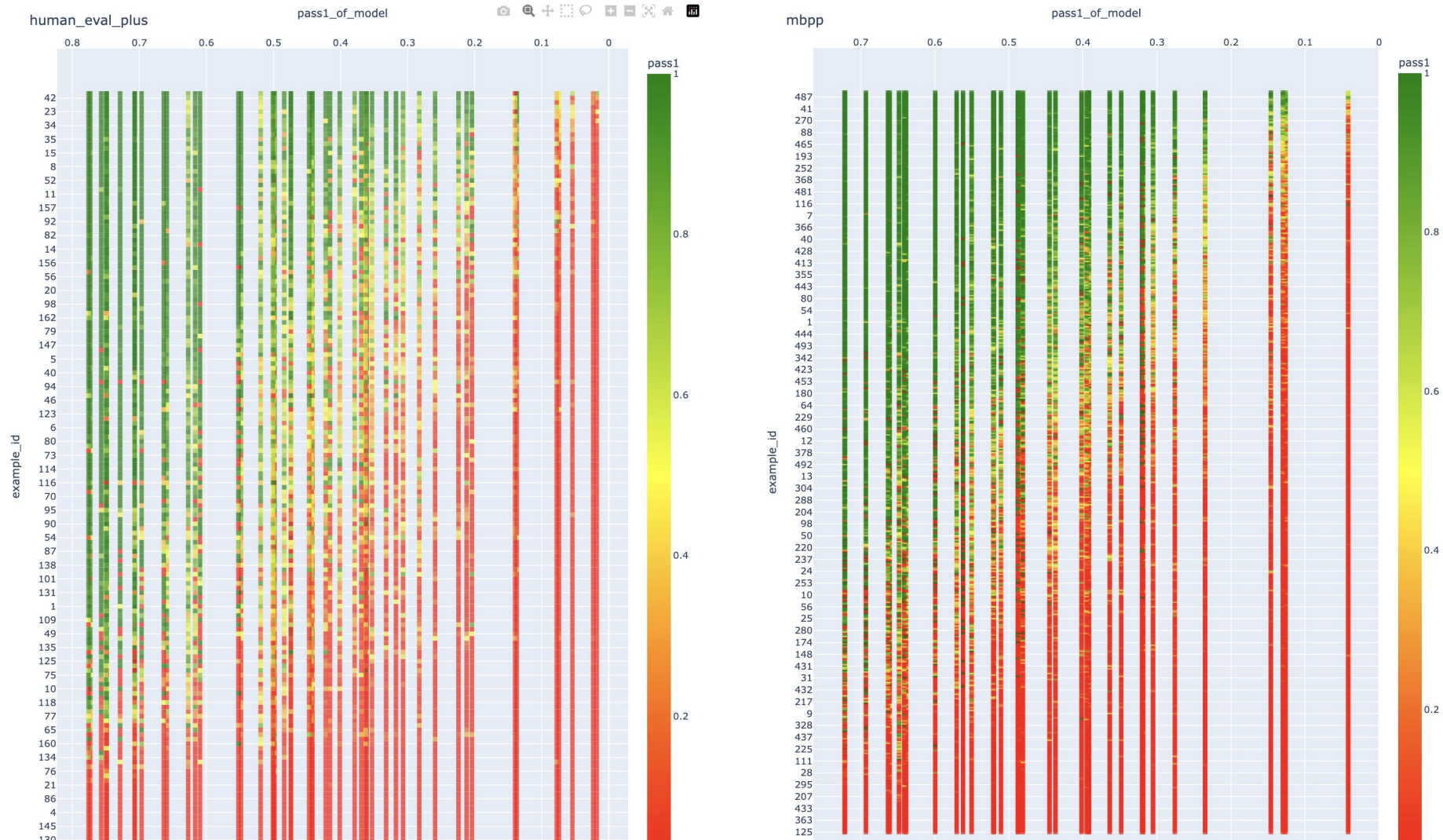
- A: small benchmarks are not reliable even if they contains some hard generative/agentic problems.

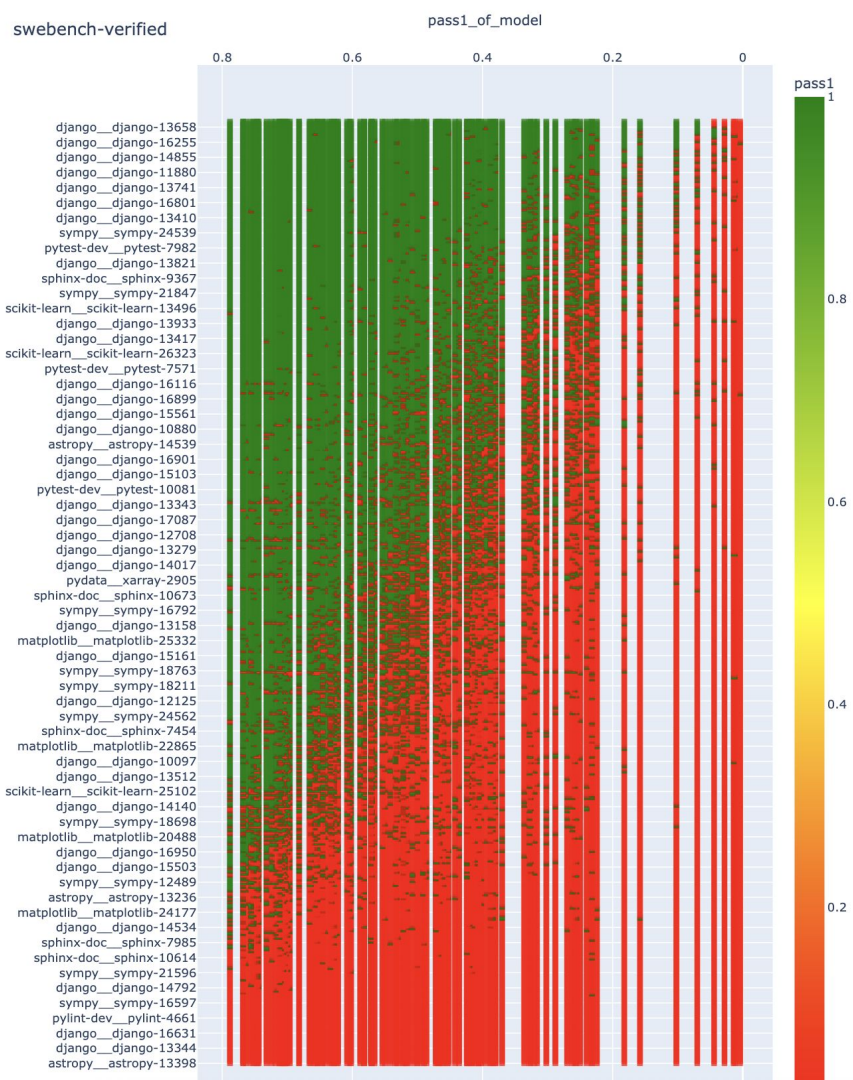
- B: Solving a single hard problem could be significant already, so we can and should use small but very hard benchmarks

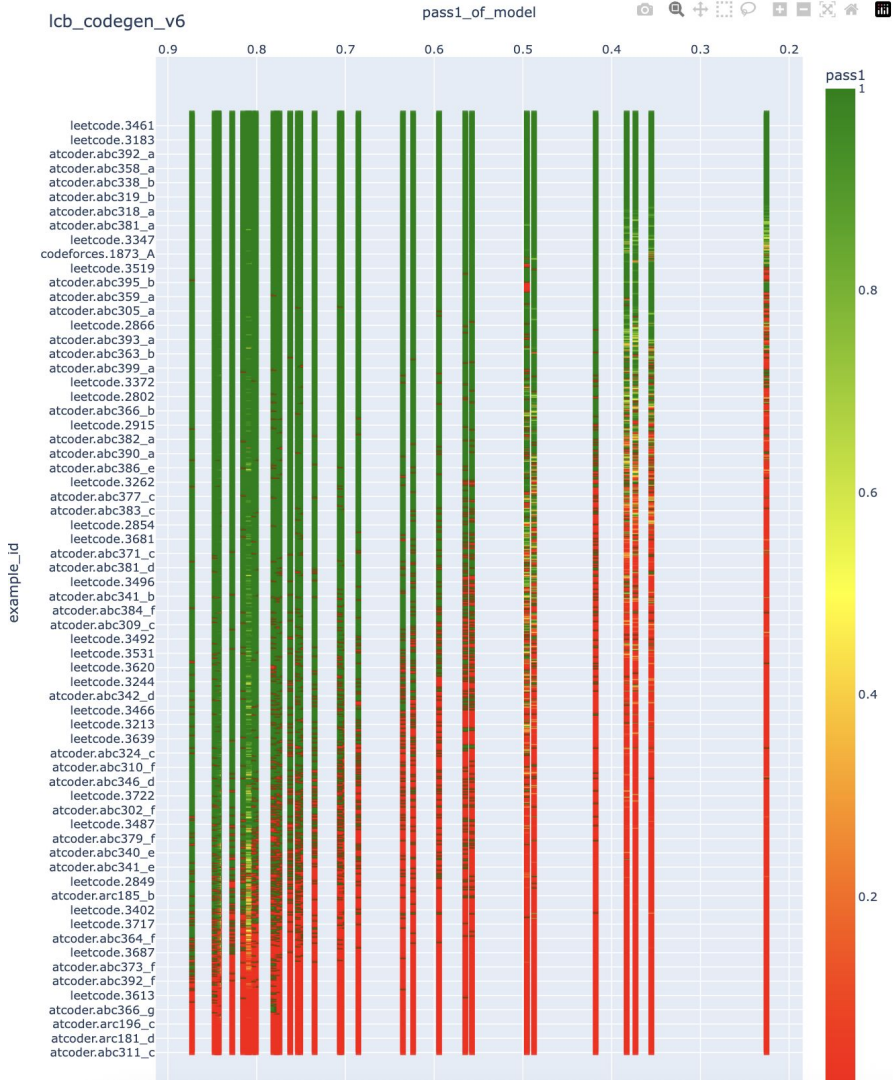
Let’s do some exploratory data analysis

- Each column is a model, x axis is their pass@1
- Each row is a question, sorted by their difficulty. From easiest at top to hardest at bottom
- Color is pass@1 of model on a question, over 10+ samples whenever possible




human_eval_plus
pass1_of_model
0.8 0.7 0.6 0.5 0.4 0.3 0.2 0.1 0
example_id
42 23 34 35 15 8 52 11 157 92 82 14 156 56 20 98 162 79 147 5 40 94 46 123 6 80 73 114 116 70 95 90 54 87 138 101 131 1 109 49 135 125 75 10 118 77 65 160 134 76 21 86 4 145
pass1
1 0.8 0.6 0.4 0.2
mbpp
pass1_of_model
0.7 0.6 0.5 0.4 0.3 0.2 0.1 0
example_id
487 41 270 88 465 193 252 368 481 116 7 366 40 428 413 355 443 80 54 1 444 493 342 423 453 180 64 229 460 12 378 492 13 304 288 204 98 50 220 237 24 253 10 56 25 280 174 148 431 31 432 217 9 328 437 225 111 28 295 207 433 363 125
pass1
1 0.8 0.6 0.4 0.2


swebench-verified
pass1_of_model
0.8
0.6
0.4
0.2
0
pass1
1
0.8
0.6
0.4
0.2
example_id
django__django-13658
django__django-16255
django__django-14855
django__django-11880
django__django-13741
django__django-16801
django__django-13410
sympy__sympy-24539
pytest-dev__pytest-7982
django__django-13821
sphinx-doc__sphinx-9367
sympy__sympy-21847
scikit-learn__scikit-learn-13496
django__django-13933
django__django-13417
scikit-learn__scikit-learn-26323
pytest-dev__pytest-7571
django__django-16116
django__django-16899
django__django-15561
django__django-10880
astropy__astropy-14539
django__django-16901
django__django-15103
pytest-dev__pytest-10081
django__django-13343
django__django-17087
django__django-12708
django__django-13279
django__django-14017
pydata__xarray-2905
sphinx-doc__sphinx-10673
sympy__sympy-16792
django__django-13158
matplotlib__matplotlib-25332
django__django-15161
sympy__sympy-18763
sympy__sympy-18211
django__django-12125
sympy__sympy-24562
sphinx-doc__sphinx-7454
matplotlib__matplotlib-22865
django__django-10097
django__django-13512
scikit-learn__scikit-learn-25102
django__django-14140
sympy__sympy-18698
matplotlib__matplotlib-20488
django__django-16950
django__django-15503
sympy__sympy-12489
astropy__astropy-13236
matplotlib__matplotlib-24177
django__django-14534
sphinx-doc__sphinx-7985
sphinx-doc__sphinx-10614
sympy__sympy-21596
django__django-14792
sympy__sympy-16597
pylint-dev__pylint-4661
django__django-16631
django__django-13344
astropy__astropy-13398


lcb_codegen_v6
pass1_of_model
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
pass1
1
0.8
0.6
0.4
0.2
example_id
leetcode.3461
leetcode.3183
atcoder.abc392_a
atcoder.abc358_a
atcoder.abc338_b
atcoder.abc319_b
atcoder.abc318_a
atcoder.abc381_a
leetcode.3347
codeforces.1873_A
leetcode.3519
atcoder.abc395_b
atcoder.abc359_a
atcoder.abc305_a
leetcode.2866
atcoder.abc393_a
atcoder.abc363_b
atcoder.abc399_a
leetcode.3372
leetcode.2802
atcoder.abc366_b
leetcode.2915
atcoder.abc382_a
atcoder.abc390_a
atcoder.abc386_e
leetcode.3262
atcoder.abc377_c
atcoder.abc383_c
leetcode.2854
leetcode.3681
atcoder.abc371_c
atcoder.abc381_d
leetcode.3496
atcoder.abc341_b
atcoder.abc384_f
atcoder.abc309_c
leetcode.3492
leetcode.3531
leetcode.3620
leetcode.3244
atcoder.abc342_d
leetcode.3466
leetcode.3213
leetcode.3639
atcoder.abc324_c
atcoder.abc310_f
atcoder.abc346_d
leetcode.3722
atcoder.abc302_f
leetcode.3487
atcoder.abc379_f
atcoder.abc340_e
atcoder.abc341_e
leetcode.2849
atcoder.arc185_b
leetcode.3402
leetcode.3717
atcoder.abc364_f
leetcode.3687
atcoder.abc373_f
atcoder.abc392_f
leetcode.3613
atcoder.abc366_g
atcoder.arc196_c
atcoder.arc181_d
atcoder.abc311_c

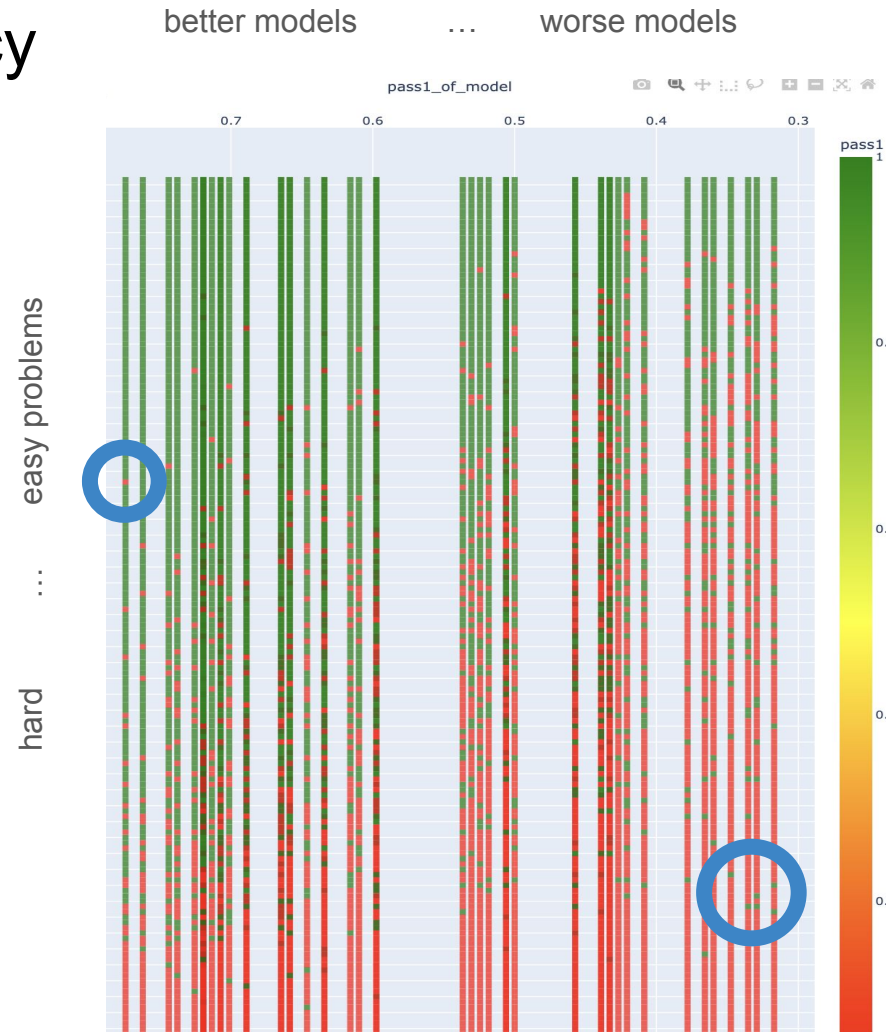
There is a lot of inconsistency

HumanEval

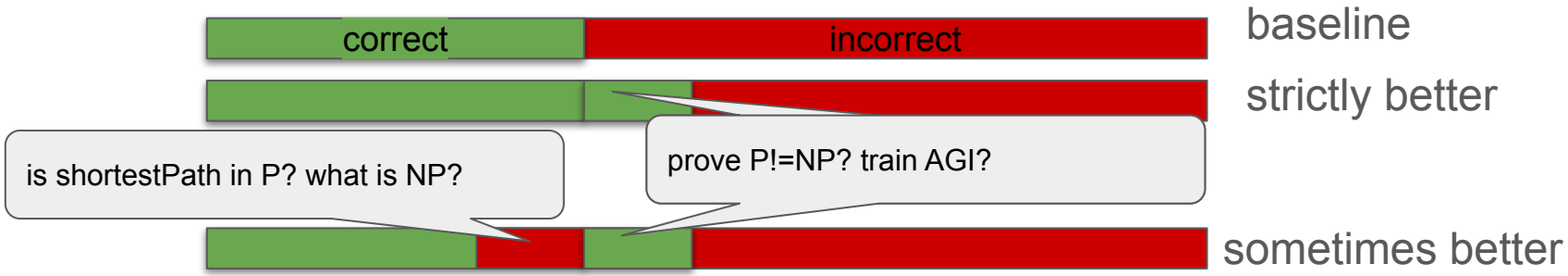
The worst models sometimes succeeds on hard problem

The best models still fails on easy problems

So perhaps the problems are not too special



Can we make up for the low statistical power with hard questions with long answers?



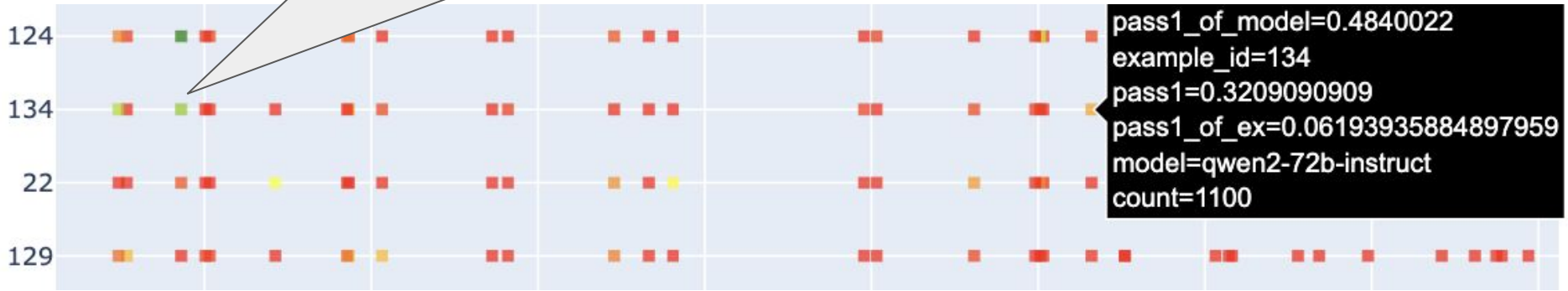
Not if the answers are inconsistent:

- wrote down a hard and long proof of P!=NP but cannot define NP? copied it somewhere?

red are not sharp 0s, but just lower probability

This question cannot be a very hard open problem if a much weaker model just have a smaller probability of solving the problem

Increasing from 1% correct to 99% may not be that significant (especially on verifiable problems). The first 1% is the most impressive.



So what’s the case?

- **A: small benchmarks are not reliable even if they contains some hard generative/agentic problems**

- ~~B: Solving a single hard problem could be significant already, so we can and should use small and hard benchmarks~~

Help fix my kettle

- kettle (boils water)
- agent says it’s a robotic vacuum
- should I trust its later advice (could still be correct)

label the parts for me

Thought process

Dissected robotic vacuum parts for comprehensive labeling

Looking at this disassembled device (which appears to be a robot vacuum wheel/drive assembly), here are the main components I can identify:

Key Parts:

1. **Drive Motor** - The central circular component with the brass/bronze colored ring - this provides power to rotate the wheel
2. **Motor Housing/Mount** - The black and

- Hard problems often have many steps
- Probability of error increases exponentially as the number of steps increases

So probably statistical analysis is still important

How does it work? Let’s go over the basics

I’ll put down some math, and convey the intuition with some examples

The super-population (basic stats, Miller 2024)

- With sample questions from a super population, then the mean and variance of the population can be estimated from these sample

On a evaluation, we compute the average of all N questions to get the mean accuracy

$$\bar{A} := \frac{1}{N}\sum_{i=1}^{N} A(x_i) \approx \mathrm{E}_x[A(x)].$$

The estimated variance of A and standard error (SE) of the mean $\bar{A}$ are respectively

$$\mathrm{Var}_x[A(x)] \approx \frac{1}{N}\sum_{i=1}^{N} \left(A(x_i) - \bar{A}\right)^2,$$

$$\mathrm{SE}(A) = N^{-1/2} \cdot \mathrm{Var}[A]^{1/2}.$$

Paired comparison. $A(x)$ and $B(x)$ are correlated, and in an actual benchmark, the same set of question x is used for both $A(x)$ and $B(x)$. To be more powerful for free, we can use the paired variance

$$\mathrm{Var}_x[A(x) - B(x)] = \mathrm{Var}_x[A] + \mathrm{Var}_x[B] - 2\mathrm{Cov}_x[A(x), B(x)]. \quad (4)$$

The Cov term may potentially reduce the paired variance to 0 when A and B are perfectly correlated.

Ask if this result is likely just due to chance

Once we have the standard error, we obtain the z-score $z = \frac{\bar{A}-\bar{B}}{\mathrm{SE}\,\mathrm{Var}[A-B]}$ which follows a standard normal distribution. For example, $\Pr[|z| > 1] = 0.32$ is very weak, $\Pr[|z| > 5] < 10^{-6}$ is beyond doubt, and $\Pr[|z| > 1.96] = 0.05$ for a p-value of 5% is a conventional and reasonable standard.

Noise seems pretty large for small datasets

Example 1 (HumanEval)**.** For the popular HumanEval dataset, $N = 164$, say a model has $\bar{A} = 0.5$ accuracy. The SE is 4%. you'd need to produce a difference of $4\% \times \sqrt{2} \times 1.96 = 10.8\%$ to get 0.05 p-value using the unpaired test. With a paired test, the difference SE is typically still 4%, so a difference of $4\% \times 1.96 = 7.84\%$ is needed to produce a 0.05 p-value. Both are a larger difference than the gains reported by many papers.

Paired vs. Unpaired: intuition

Example 2 (Paired vs unpaired)**.** In the unpaired case, suppose A scored 50% on this year's exam, B scored 60% on last year's exam and the exams contain different exact questions drawn from the same distribution. So B might have done better because last year's exam was easier. In the paired case, suppose B scored 60% on the same exact exam as A with exactly the same questions, then the same 10% difference is more indicative of the better performance by B.

Paired: xi ~ Data -> A(xi), B(xi)
Unpaired: xi, xj ~ Data -> A(xi), B(xj)

Use the ability to draw independent samples?

This is a remarkable ability of the models, humans can't seem to do this. A common confusion is to consider this the only type of noise.

To capture this setting, we use ϵ for the seed generating the noise, which is independent of the question x, thus allowing the concise notation $\mathrm{Var}_x[\mathrm{E}_\epsilon[A]] = \mathrm{Var}_x[E_{\epsilon|x}[A(x,\epsilon) \mid x]]$. As before, but now also averaging over the K samples for each question x, we consider the average score

$$\bar{A} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{K}\sum_{j=1}^{K} A(x_i, \epsilon_{ij}) \approx \mathrm{E}_{x,\epsilon}[A(x,\epsilon)], \tag{5}$$

The estimated variance is then

$$\mathrm{Var}_{x,\epsilon}[A(x,\epsilon)] \approx \frac{1}{N}\sum_{i=1}^{N}\frac{1}{K}\sum_{j=1}^{K}(A(x_i,\epsilon_{ij}) - \bar{A})^2. \tag{6}$$

Total variance to standard errors

We have the law of total variance satisfied by the components

$$\mathrm{Var}_{x,\epsilon}[A] = \mathrm{Var}_x[\mathrm{E}_\epsilon[A]] + \mathrm{E}_x[\mathrm{Var}_\epsilon[A]]. \tag{7}$$

$\mathrm{Var}_{x,\epsilon}[A]$ is the total variance of first drawing a question x from the pool of questions, and then drawing a sample answer for this question. $\mathrm{Var}_x[\mathrm{E}_\epsilon[A]]$ is variance of sampling from the data, which is also the variance of the expected score $\mathrm{E}_\epsilon[A]$. $\mathrm{E}_x[\mathrm{Var}_\epsilon[A]]$ is the variance due to sampling from the model. All of which can be normalized to be the standard errors,

$$\mathrm{SE}(A) = N^{-1/2}(\mathrm{Var}_{x,\epsilon}[A])^{1/2}, \tag{8}$$

$$\mathrm{SE}_x(A) = N^{-1/2}(\mathrm{Var}_x[\mathrm{E}_\epsilon[A]])^{1/2}, \tag{9}$$

$$\mathrm{SE}_\epsilon(A) = N^{-1/2}(\mathrm{E}_x[\mathrm{Var}_\epsilon[A]])^{1/2}. \tag{10}$$

Paired comparison

Paired comparison. When we have two models A and B, we can consider the paired standard deviation $\mathrm{Var}_{x,\epsilon}[A(x,\epsilon) - B(x,\epsilon)]$, which also satisfy the law of total variance on $A - B$,

$$\mathrm{Var}_{x,\epsilon}[A - B] = \mathrm{Var}_x[\mathrm{E}_\epsilon[A - B]] + \mathrm{E}_x[\mathrm{Var}_\epsilon[A - B]]. \tag{11}$$

This formula allows us to compute from the samples directly.

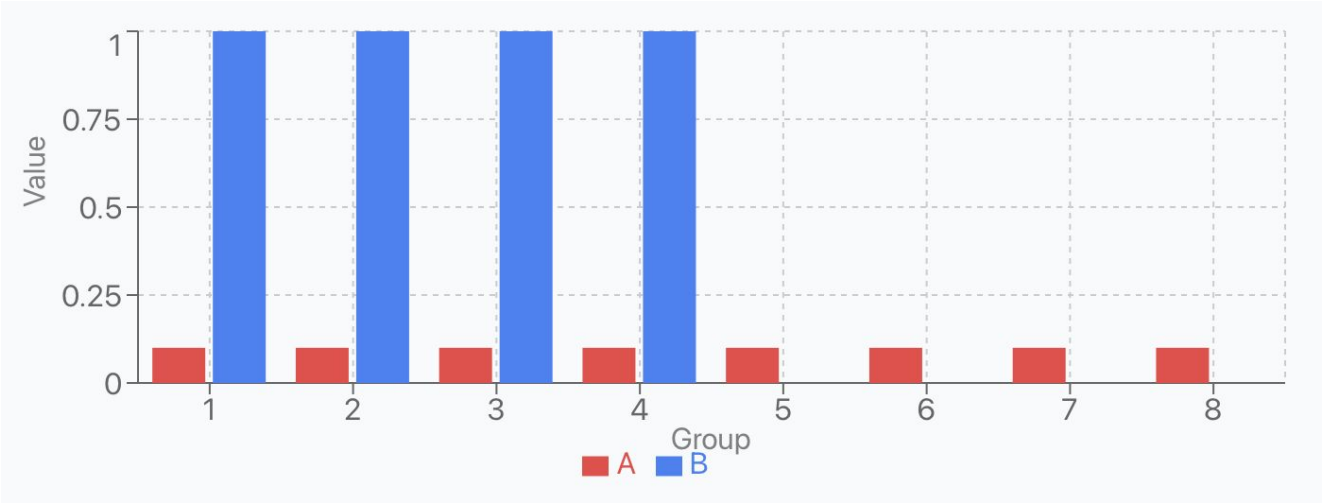
$$\mathrm{Var}_{x,\epsilon}[A - B] = \mathrm{Var}_{x,\epsilon}[A] + \mathrm{Var}_{x,\epsilon}[B] - 2\mathrm{Cov}_{x,\epsilon}[A, B] \tag{12}$$

$$= \mathrm{Var}_{x,\epsilon}[A] + \mathrm{Var}_{x,\epsilon}[B] - 2\mathrm{Cov}_x[\mathrm{E}_\epsilon[A], \mathrm{E}_\epsilon[B]]. \tag{13}$$

By the law of total covariance, $\mathrm{Cov}_{x,\epsilon}[A, B] = \mathrm{Cov}_x[\mathrm{E}_\epsilon[A], \mathrm{E}_\epsilon[B]] + \mathrm{E}_x[\mathrm{Cov}_\epsilon[A, B]]$ where the second term is 0 since different random samples are independent.

Correct with probability 0.8 leads to some weirdness

Example 4 (better mean vs. better max)**.** Suppose B gains a consistent improvement over A on all questions i, $\mathrm{E}_\epsilon[A(x_i, \epsilon)] = 0.6$, $\mathrm{E}_\epsilon[B(x_i, \epsilon)] = 0.8$. While this can easily be statistically significant, it might be also be achievable by majority voting, verification or just sharpening the distribution. Now suppose that on half of the questions, $\mathrm{E}_\epsilon[A] = 0.1$, $\mathrm{E}_\epsilon[B] = 1$, but on the other half we have $\mathrm{E}_\epsilon[A] = 0.1$, $\mathrm{E}_\epsilon[B] = 0$. While B has a much higher average, A might actually be much better. For example, if the questions are problems that can be verified such as search problems or proofs, then A is successful 100% of the time with a verifier in the loop.



Compute with simple array operations

~~The answer is p(1-p), let's compute with bootstrap…~~

Name	formula	code	Bernoulli
total variance	$\mathrm{Var}_{x,\epsilon}[A]$	`var(A)`	$\bar{p}(1-\bar{p})$
data variance	$\mathrm{Var}_{x}[\mathrm{E}_{\epsilon}[A]]$	`var(mean(A, axis=1))`	$\frac{1}{N}\sum_i (p_i-\bar{p})^2$
model variance	$\mathrm{E}_{x}[\mathrm{Var}_{\epsilon}[A]]$	`mean(var(A, axis=1))`	$\frac{1}{N}\sum_i p_i(1-p_i)$

Table 1: translation of unpaired to code where $A \in \mathbb{R}^{N\times K}$ with K samples for each of N questions.

Formula	Code
$\mathrm{Var}_{x,\epsilon}[A-B]$	`var(A)+var(B)-2*cov(mean(A,axis=1),mean(B,axis=1))`
$\mathrm{Var}_{x}[\mathrm{E}_{\epsilon}[A-B]]$	`var(mean(A,axis=1)-mean(B,axis=1))`
$\mathrm{E}_{x}[\mathrm{Var}_{\epsilon}[A-B]]$	`mean(var(A,axis=1)+var(B,axis=1))`

Table 2: code for paired estimators (with bias)

The other viewpoints gives about the same results

Bootstrap: Drawing samples (with replacement) from the empirical distribution

≅Drawing more samples from unseen super population

Sign test: wherever predictions are different, how likely is it to be due to randomness (same for A and B that are close. if not close, then already statistically different)

Eval-Arena (doc/code)

Results: https://crux-eval.github.io/eval-arena/main/

For each benchmark, do pairwise comparisons / statistical testing on all models pairs. Establish that no model is consistently better than another model (among statistically questionable models).

- Assuming no special problems and the model has some chance of error on each problem, we can measure noise meaningfully for each dataset
- **you can just read off the numbers in eval-arena instead of trusting the authors to do their testing!**

Predictable noise

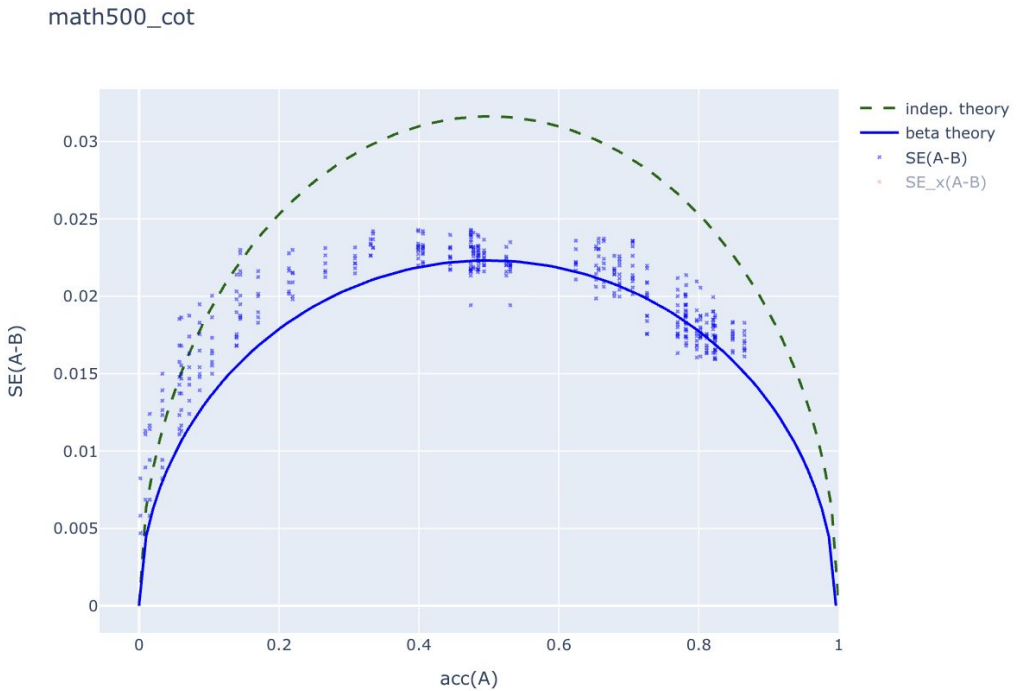
SE(A) ~ SE(A-B)

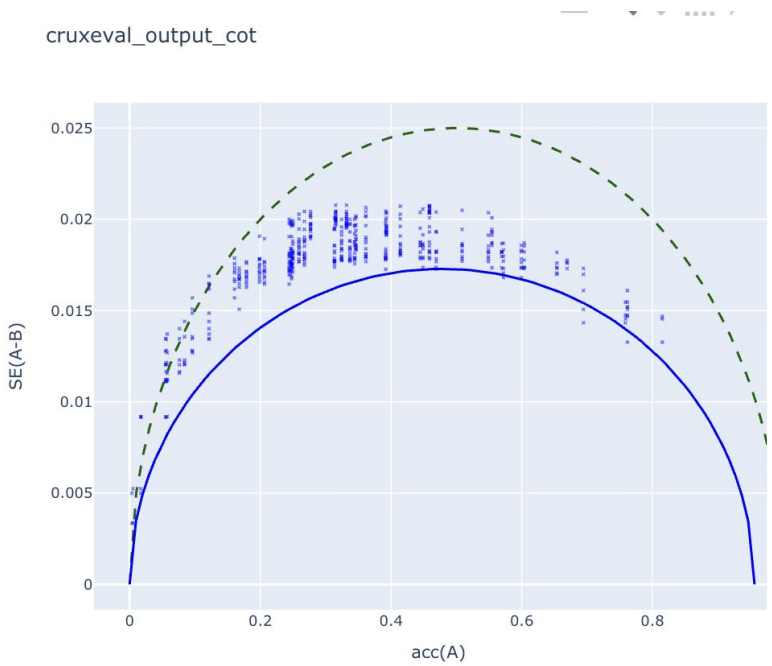
are roughly similar

-> correlation ~ 0.5

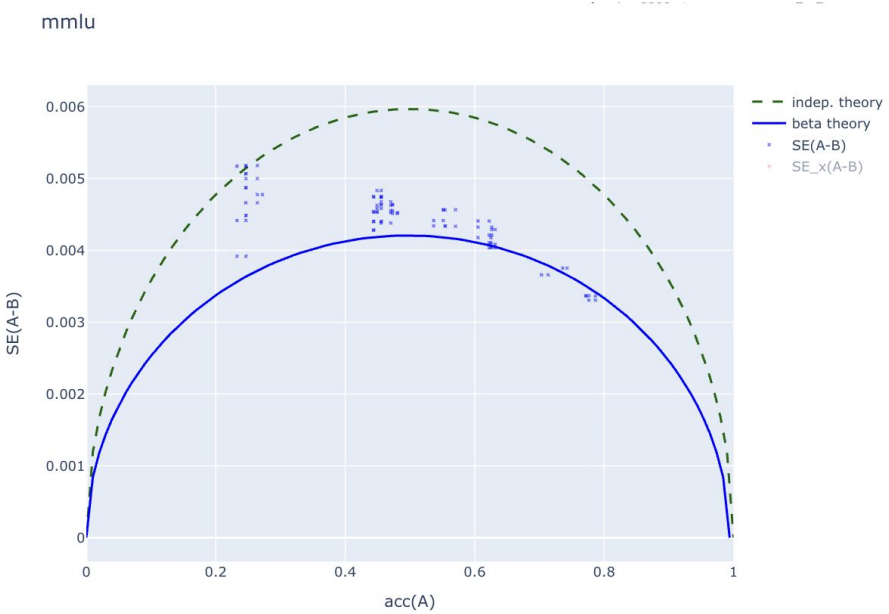
benchmark_id	size	models	SE(A)	SE(A-B)	corr(A,B)
CRUXEval-input-T0.2	800	37	1.7	1.8	56
CRUXEval-input-T0.8	800	31	1.6	1.8	69
CRUXEval-output-T0.2	800	37	1.7	1.6	64
CRUXEval-output-T0.8	800	31	1.7	1.6	75
DS1000	1000	105	1.3	1.5	46
agi_english	2546	35	0.93	1.1	25
arc_challenge	1165	38	1.4	1.3	62
gsm8k	1319	37	1.1	1.3	38
hellaswag	10042	36	0.41	0.26	77
humaneval	164	78	3.5	3.7	47

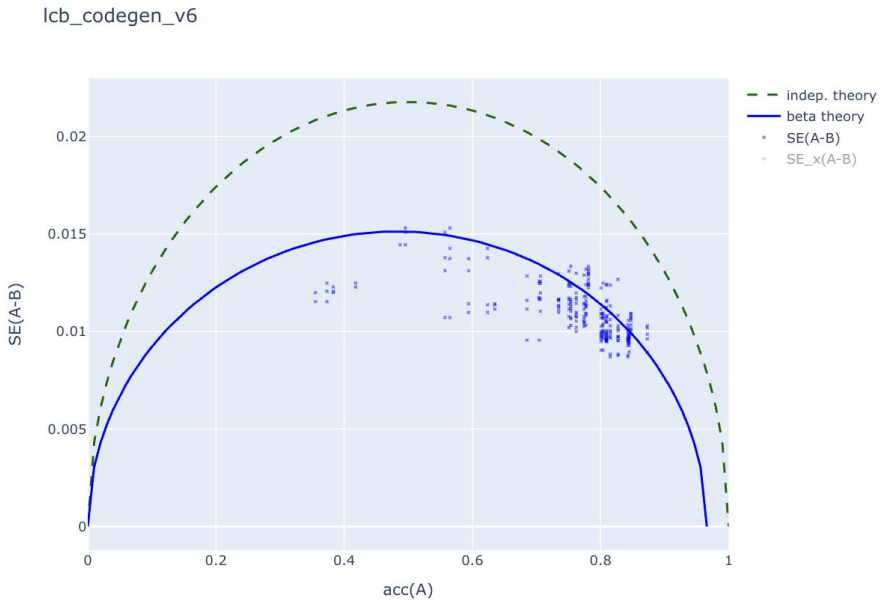
There is a dependence on overall acc(A)



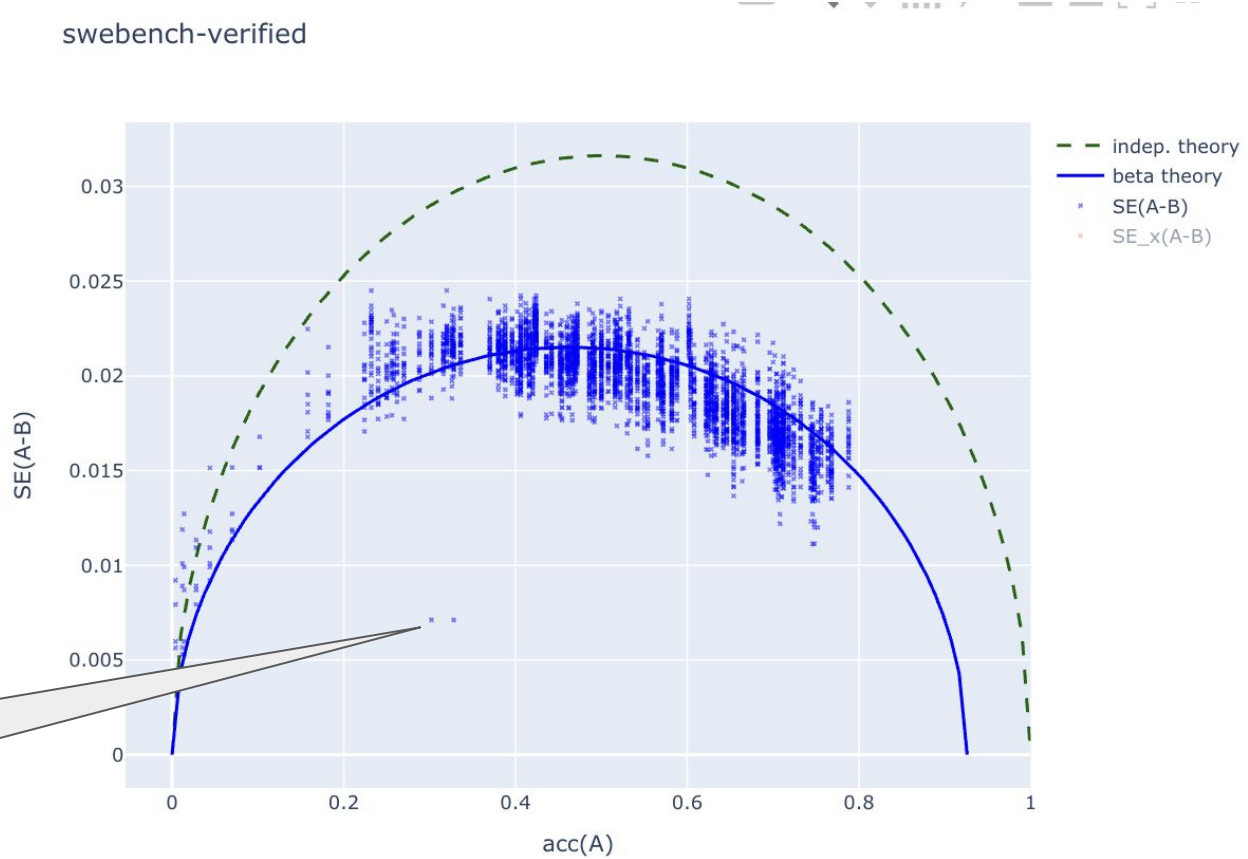


There is a dependence on overall acc(A)





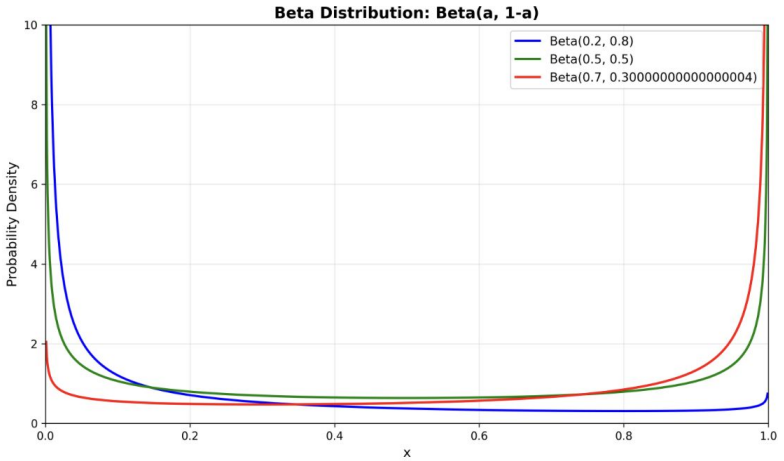
Swebench verified

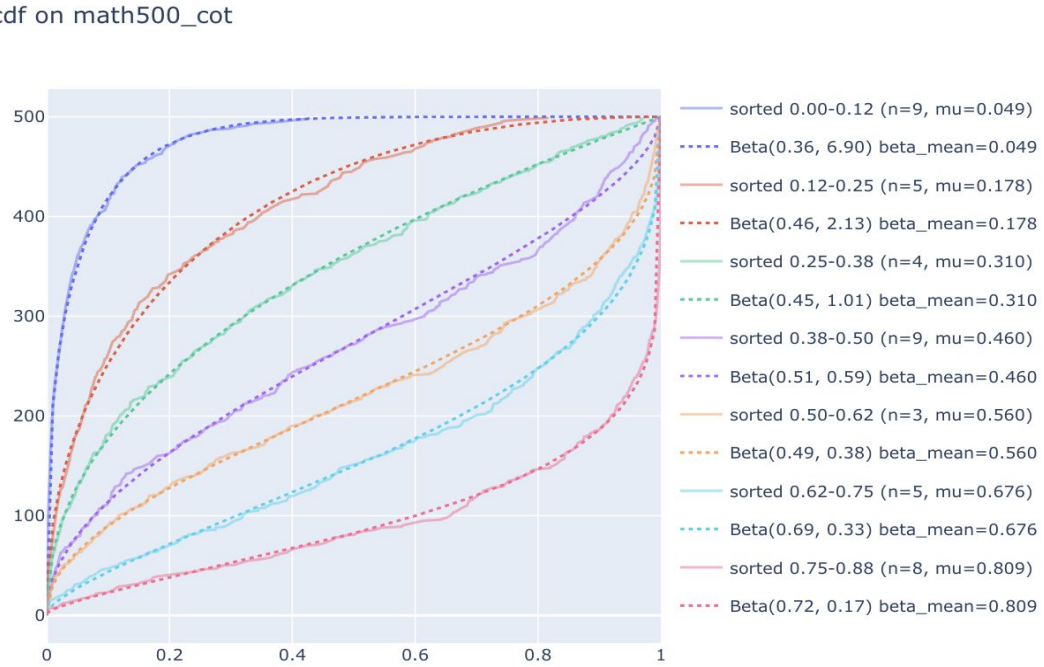


SWEFixer, vs.
SWEFixer+P2P filter

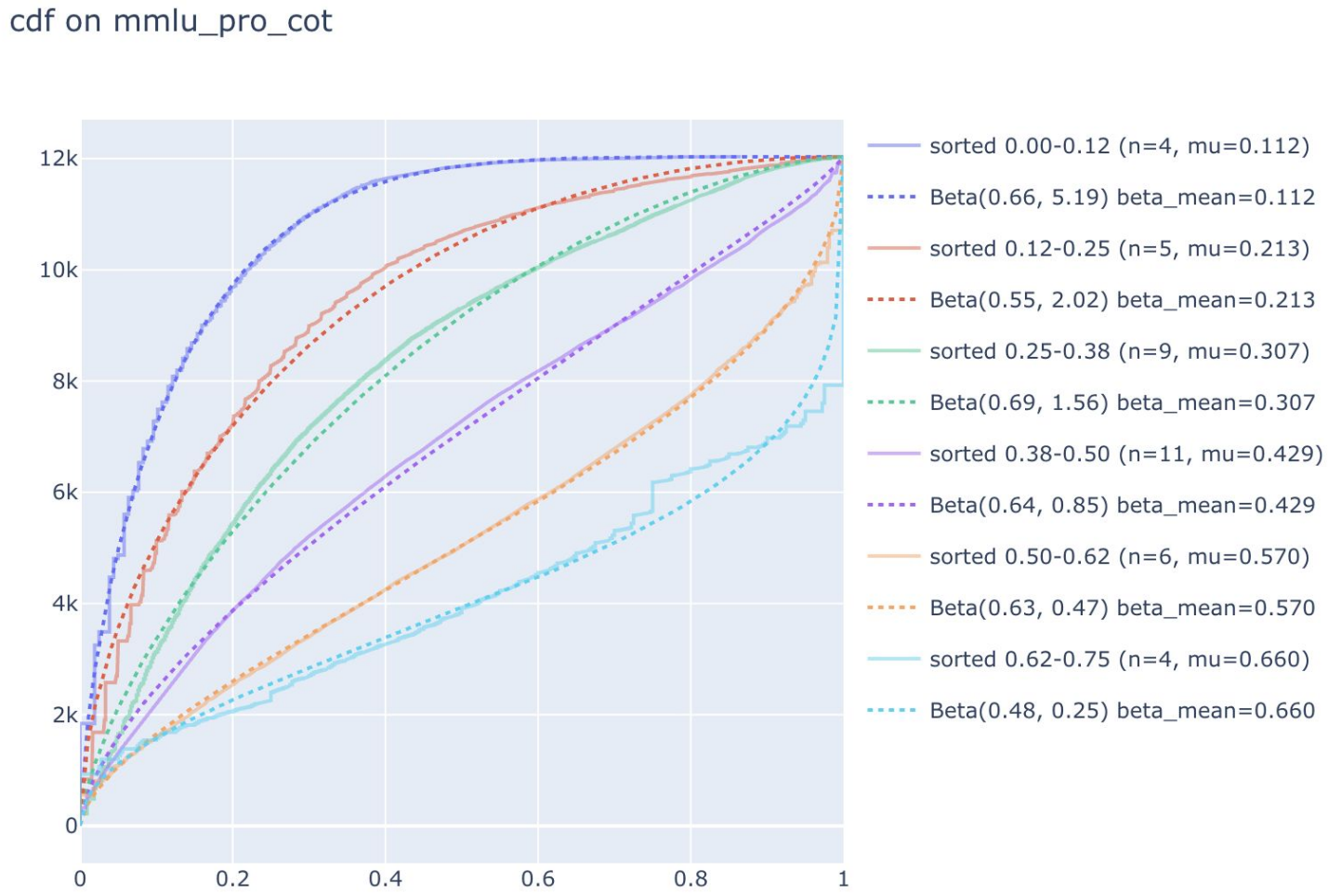
Empirical probabilities follow a Beta distribution

Once performance is good, then the bimodal ~ Beta(a, 1-a)



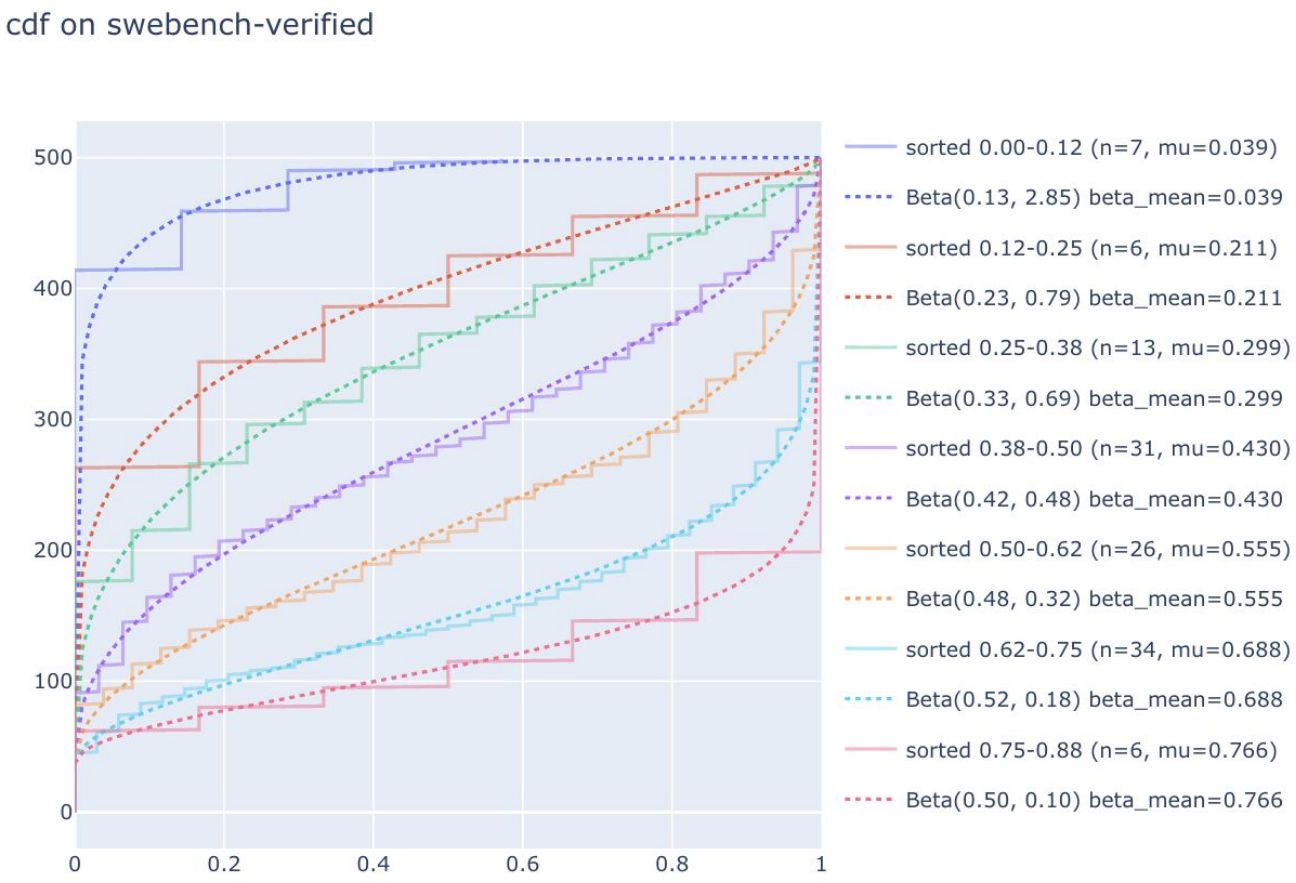


More beta


cdf on mmlu_pro_cot
12k
10k
8k
6k
4k
2k
0
0
0.2
0.4
0.6
0.8
1
sorted 0.00-0.12 (n=4, mu=0.112)
Beta(0.66, 5.19) beta_mean=0.112
sorted 0.12-0.25 (n=5, mu=0.213)
Beta(0.55, 2.02) beta_mean=0.213
sorted 0.25-0.38 (n=9, mu=0.307)
Beta(0.69, 1.56) beta_mean=0.307
sorted 0.38-0.50 (n=11, mu=0.429)
Beta(0.64, 0.85) beta_mean=0.429
sorted 0.50-0.62 (n=6, mu=0.570)
Beta(0.63, 0.47) beta_mean=0.570
sorted 0.62-0.75 (n=4, mu=0.660)
Beta(0.48, 0.25) beta_mean=0.660

Betas

only binary preds



The failure: why we cannot just boost the signal

My original goal is to see if I can make current benchmarks less noisy

- Still just getting 1 bit of information per hard example.
- Model performance is very noisy and unreliable
  - indirect evidence of memorization when a bad model solves a hard problem
  - unless it's all open problems, there could always be memorization. "we deduplicate all n-grams overlap"
- Modelling difficulty and error of each problem failed to get much signal. Item Response Model failed to do better than baseline.
- Other's attempt to correct errors did not lead to any measurable improvements
  - Subjectively poor quality dataset (MBPP) has better signal to noise than subjectively higher quality dataset (HumanEval+).
  - HumanEval+, MBPP+ is not any better than the original

Attempting to boost signal to noise by filtering, reweighting problems is not promising (because the models are the bigger sources of inconsistency than the benchmarks)

Recommend: Report on more benchmarks. Collect bigger datasets or at least attempt to collect **more information per example** (i.e. > 1 bit) on complex and expensive tasks i.e. SWE-Bench.

To be reconsidered when models are less random!

Conclusions for measuring noise

- run with multiple seeds: https://arxiv.org/abs/2406.10229v1
  - usually an underestimate, but can produce significant results. compatible to this work.
- variance of the training curve (underestimate)
- We recommend: in increasing accuracy, and decreasing convenience
  - read off our table
  - read off our figure in your accuracy range
  - Export your results and run our code, provides context
    - benchmark/leaderboard builders should share the full question level results
    - Do your own statistical tests and calculations?? fairly error prone

Other features of Eval-Arena (docs)

Noise level references and signal-to-noise ratio (this talk)

Model performance ranked by ~~Elo~~, win-rate, accuracy (best)

Tool for finding suspect examples (we found more errors / inconsistencies on MBPP+) and some ways. Explore if time allows

Finding suspicious models

Signal to noise

The noise values allows us to interpret reported results. To get a sense of the quality of the benchmark, we need signal to noise ratio.

Signal: the gain from doubling the size of model in each series

Noise: implied standard deviation by statistical testing (same as bootstrapping)

sig/noise = |A-B| / SE(A-B)

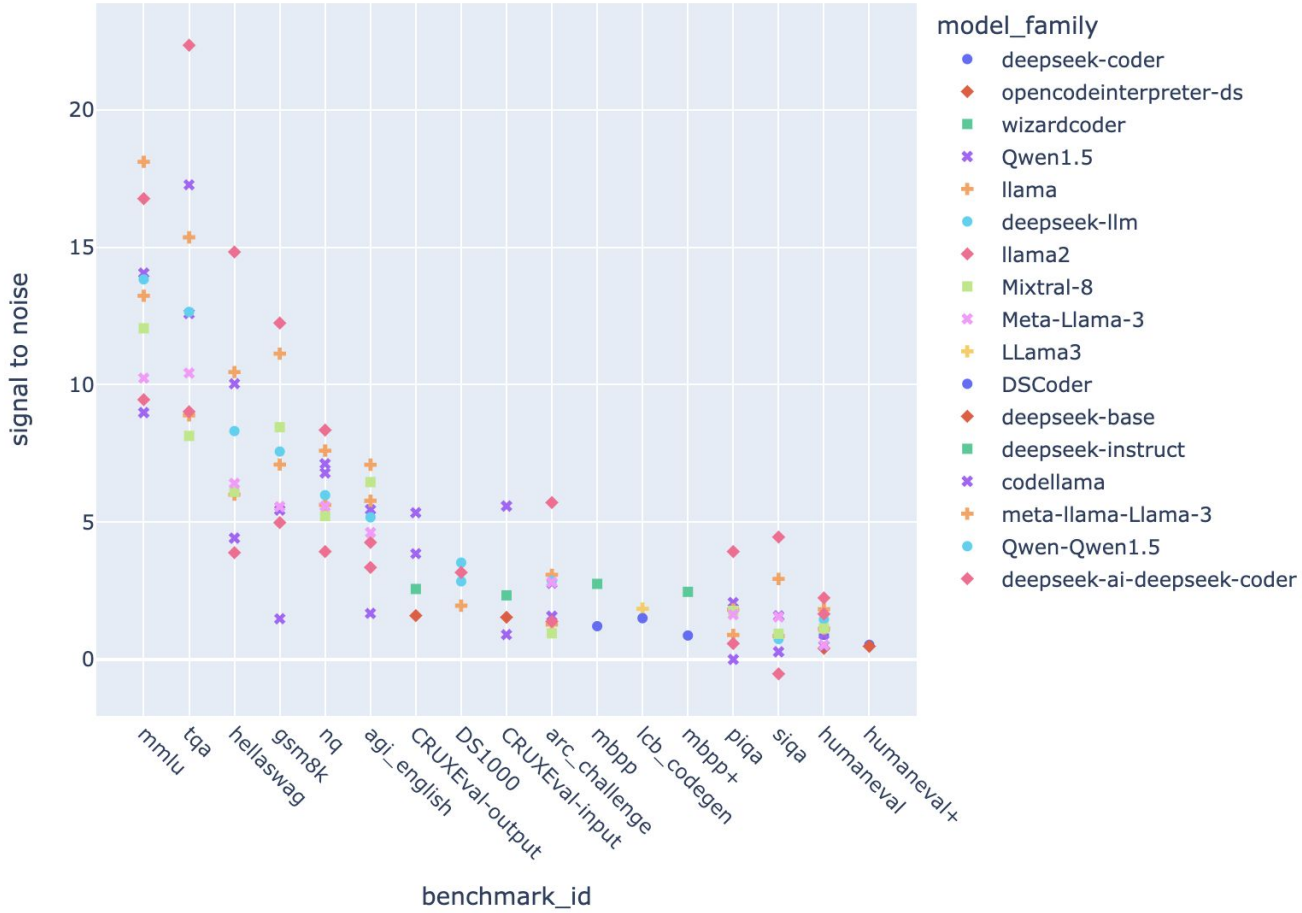
sig/noise < 2: cannot reliably measure the improvement from doubling model size. True on most **code benchmarks!** examples of inversion

Simple multiple choice benchmarks have much better signal to noise ratio than complex generation benchmarks

benchmark_id	size	sig/noise
mmlu	14042	13.2
tqa	11313	12.6
gsm8k	1319	7.1
hellaswag	10042	6.4
nq	3610	5.9
agi_english	2546	5.1
CRUXEval-output	800	3.2
DS1000	1000	3.0
arc_challenge	1165	2.7
mbpp	378	1.9
CRUXEval-input	800	1.9
piqa	1838	1.7
lcb_codegen	400	1.6
mbpp+	378	1.6
humaneval	164	1.1
siqa	1954	0.93
humaneval+	164	0.50

signal-to-noise

by model class as well



When everyone is too busy to look into the details of the eval because it's too complicated and 100k tokens, bad things can happen

https://github.com/SWE-bench/SWE-bench/issues/465

@UniverseFly found it, several others from the team investigated, as mentioned in the issue

Paired comparisons