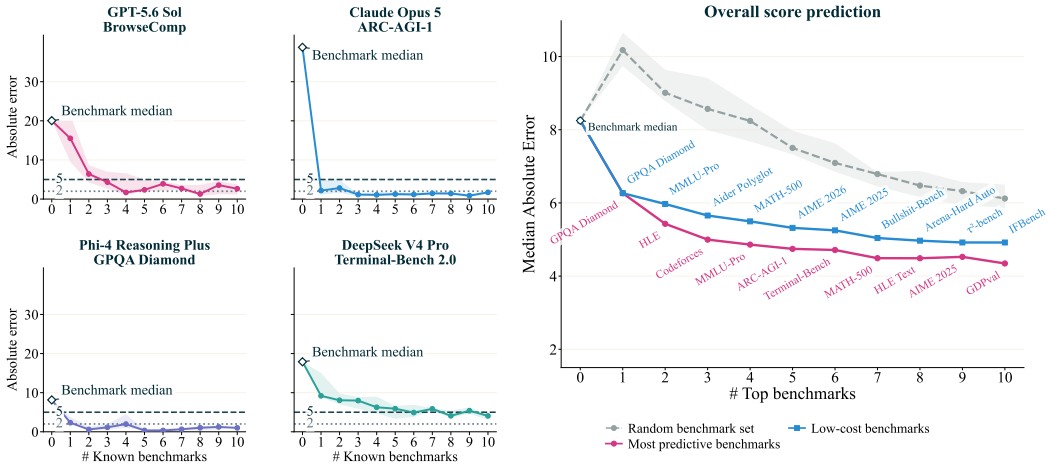
YOU DON'T NEED TO RUN EVERY EVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

A modern model release reports scores on 40+ benchmarks and the same evaluations were run many more times before it: to track training progress, compare design choices, and select the checkpoint for the release. But do we need to run every eval? We compile a public score matrix of 84 frontier models on 133 benchmarks (2,604 cells, 23.3% filled) and find it is approximately rank-2: two factors capture most of how a model's scores vary across benchmarks. We confirm this in two ways: (i) scores hidden from the matrix are best recovered using two factors, and (ii) two factors already explain over 90% of the variation among models on the benchmarks they share. Building on this, we design BENCHPRESS: a logit-space rank-2 matrix completion method with a median absolute error of 4.6 points on held-out scores. Using BENCHPRESS, we find a subset of five benchmarks {GPQA-D, HLE, Codeforces, MMLU-Pro, ARC-AGI-1} that can recover the rest of a model's public scorecard with a median absolute error of 4.74 points. For a tighter evaluation budget, a cheaper set {GPQA-D, MMLU-Pro, Aider Polyglot, MATH-500, AIME 2026} can predict a model's remaining evals with a median absolute error of 5.32 points. Finally, we identify what affects prediction reliability and combine these factors with predictor disagreement to quantify when predictions can be trusted. Code and data are available at https://anonymous.4open.science/r/benchpress-iclr2027.



(a) Accuracy of predicting one (model, benchmark) score.

(b) Reporting overall score-prediction error.

Figure 1: **BENCHPRESS predicts unseen benchmark scores from known benchmark scores. Left:** For each of four model–benchmark examples, we randomly reveal k of the same model's other scores and hide the rest, including the target. The y-axis shows the absolute error for the highlighted target score. Error drops sharply after only a few scores from the same model are revealed. **Right:** A complementary setting that mimics how a practitioner would run BENCHPRESS in practice. A fixed set of k benchmarks is chosen as the probe set, and every model is evaluated on whichever probe scores it has observed; BENCHPRESS predicts the rest of each model's scores, and we report pooled error on these predicted cells. With only five benchmark probes selected on our score matrix by the procedure in Sec. 4.1, pooled MedAE drops to 4.74 score points (5.32 when restricted to a lower inference cost list). See Sec. A.1 for the detailed experiment setting.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

LLM evaluation is expensive, and its cost is continuing to rise. A frontier model release now routinely reports scores on dozens of benchmarks: Qwen3.7-Max reports 41 language benchmark rows (Qwen Team, 2026), and Kimi K3 reports 45 benchmark rows (Moonshot AI, 2026). But the scores reported at release are only a small fraction of the evaluations run during model development. During model development, researchers repeatedly run the same benchmarks to compare checkpoints and model variants trained under different settings. After release, users run additional evaluations to compare models for their own applications. These evaluations can cost thousands of dollars and take several days. This raises a question: *Do we always need to run every evaluation, or can we use scores we already have to predict those we have not measured?*

To answer this question, we make the contributions summarized below.

1. **We compile a public score matrix and show that it is effectively rank-2.** We collect scores from public sources, canonicalize near-duplicate model variants and benchmark configurations, and filter out models and benchmarks with insufficient observations to obtain an 84×133 matrix with 2,604 observed entries (23.3%). Sec. 2 reports two independent diagnostics of this rank-2 structure: rank-sweeping Soft-Impute matrix completion minimizes held-out prediction error at rank 2, and SVDs of the largest fully-observed submatrices show that two factors explain more than 90% of variance.
2. **We build BENCHPRESS, a benchmark score predictor.** In Sec. 3, we evaluate seven feature transforms and twelve prediction methods, finding that the best full-coverage score predictor is a rank-2 alternating least squares (ALS) matrix-completion method in logit space (Koren et al., 2009). It predicts every missing model–benchmark cell, reaching 4.6 score-point median absolute error on held-out entries at 100% coverage.
3. **We show what BENCHPRESS enables for model evaluation.** (i) In Sec. 4.1, we *select compact probe sets* that recover a model's scorecard under an evaluation budget: even when restricted to a low-cost benchmark allowlist, five probes lead to pooled MedAE of 5.32 score points. (ii) In Sec. 4.2, we *verify ranking preservation*: allowing a five-point margin on the true scores, completed scores by BENCHPRESS preserve 92.1% of pairwise model rankings on the same benchmark. (iii) In Sec. 4.3, we *stress-test predictions on newly released models*: across 27 target models, each predicted from a training matrix that predates its release, five seed scores lead to median absolute error of 6.76 points on the unrevealed scores.
4. **We characterize when predictions should be trusted.** In Sec. 5, we first identify what makes a score easier or harder to predict: target-model and target-benchmark coverage, the availability of similar peer models and neighboring benchmarks, and recency of training anchors. We combine these factors with disagreement among several score predictors to estimate the reliability of each BENCHPRESS prediction. This produces a trust probability and a conformally calibrated 90% prediction interval.

Figure 1 previews BENCHPRESS on four individual model–benchmark examples and across all models in the score matrix. In both cases, as more benchmark scores are observed, BENCHPRESS quickly reduces prediction error.

Related work. Parts of this idea are not new. (i) Prior work has found low-dimensional structure in benchmark scores (Burnell et al., 2023a; Ilić & Gignac, 2024; Ruan et al., 2024; Burnham, 2025), suggesting that some scores may be inferred from others. (ii) Matrix factorisation, collaborative filtering, and latent skill models already use this idea for performance prediction (Schram et al., 2023; Zhang et al., 2024; Polo et al., 2025; Fisch et al., 2026). Some approaches predict broader performance from a fixed evaluation panel (Aitchison et al., 2023; Ruan et al., 2024). However, a new model may have public scores on a different subset of benchmarks, so these fixed-panel methods can require new evaluations rather than reuse whatever scores are already available. (iii) Collections such as HELM (Liang et al., 2023) and Every Eval Ever (Batzner et al., 2026) provide public scores for studying this prediction problem. But treating different settings of the same model, such as different reasoning efforts, as separate rows can let a predictor copy a nearly identical row rather than learn relationships across benchmarks.

Our work brings these ingredients together to predict LLM benchmark scores in practical settings while addressing the limitations above. We curate a frontier-model matrix with one representative



Figure 2: Observation pattern of the collected model-benchmark matrix, sorted by coverage. Dark cells are observed scores; only 23.3% of entries are filled.

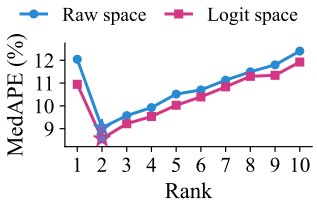


Figure 3: Held-out MedAPE vs. rank for raw- and logit-space Soft-Impute matrix completion. Both curves bottom out at rank 2.

# Bench.	# Models	Stable rank	Var (top 1)	Var (top 2)
4	42	1.11	90.4%	97.6%
7	11	1.28	78.2%	91.2%
10	7	1.16	86.5%	95.0%
13	6	1.12	89.6%	95.9%

Table 1: Complete submatrix SVD analysis across different trade-off points between benchmark coverage and model count. "Var (top k)" denotes the cumulative variance explained by the first k singular components.

setting per model and canonical benchmark configurations to avoid this shortcut. On this matrix, BENCHPRESS uses low-rank matrix completion to predict a model's unobserved benchmark scores from whichever scores are available, without a fixed input panel or predetermined prediction targets. This allows existing public scores to be used directly, without instance-level data or model access; our recommended probe sets are optional ways to collect additional scores for higher predictability. Detailed discussion on related work is provided in Sec. F.

Scope and caveats. Our claims should be read within three limits. *(i) Public-score heterogeneity:* the matrix mixes vendor-reported and third-party scores that may differ in reasoning effort, sampling temperature, prompting, judge model, evaluation harness, and benchmark version. BENCHPRESS therefore predicts what this public matrix would extrapolate to rather than what a controlled re-evaluation under a single configuration would yield. *(ii) Snapshot dependence:* the main experiments use a score matrix with a data cutoff of April 30, 2026; Sec. C.5 reports results on a later score matrix (data cutoff August 26, 2026) and on score matrices from other sources. *(iii) Score inferability:* our analysis identifies benchmark *scores* that are currently inferable from others, not benchmarks whose *existence* is unnecessary. Running a benchmark can still show where a model fails, whether it may have seen the test data before, and how it performs on new kinds of data. Benchmark results also influence what model developers choose to improve.

2 THE SCORE MATRIX AND ITS GEOMETRY

Now we ask: given a collection of existing LLM benchmark scores, can we predict the missing ones from a small subset? Our starting point is a *score matrix* with models on one axis and benchmarks on the other, populated from publicly available evaluations. Sec. 2.1 describes how each cell is sourced and introduces the resulting matrix, while Sec. 2.2 reveals that the score matrix is effectively rank-2. Throughout, M denotes the number of models, B denotes the number of benchmarks, s_{mb} denotes the observed score of model m on benchmark b, and $\hat{s}_{mb}$ denotes its prediction.

2.1 THE SCORE MATRIX

Data collection. Our data collection proceeds in four steps: we crawl model-provider materials for a seed set of recent frontier releases, recurse through competitor baselines mentioned in those sources, sweep primary leaderboards, and filter the resulting raw matrix to a dense analysis subset. When the same model-benchmark cell is reported by multiple sources, we keep the highest-priority value (release blog, system card, technical report, Hugging Face model card, then primary leaderboards), breaking ties by recency, and we fix one canonical configuration per model. Every retained value carries its source URL and evaluation-setting metadata, and roughly half come from a source outside the model's own provider, such as a competitor's release materials or a leaderboard. Details are provided in Sec. B.1.

After curation, the score matrix (data cutoff April 30, 2026) contains 2,604 observed entries out of 11,172 cells (23.3% fill rate). Figure 2 shows the sparsity pattern: popular models and benchmarks are well-covered, but the lower-right corner is almost entirely empty.

Benchmarks. The 133 benchmarks span all major LLM evaluation axes: agentic tasks and tool use, math, coding, multimodal and vision, long context, instruction following, knowledge and QA, reasoning, hallucination and factuality, science, composite indices, human preference, safety, and other specialized categories.

Models. The 84 models span 13 providers: OpenAI (20), Google (12), Anthropic (11), Alibaba/Qwen (11), DeepSeek (9), Meta (6), Zhipu AI (4), Moonshot AI (3), xAI (3), MiniMax (2), Cohere (1), ByteDance (1), and Mistral (1). Among models with annotated type, 51 are reasoning models (chain-of-thought) and 31 are non-reasoning. Of the 84 models, 40 are open-weight and 44 are closed. Where parameter counts are disclosed, they range from 1B (e.g., Gemma 3 1B (Google, 2025)) to 1.6T (e.g., DeepSeek-V4-Pro (DeepSeek-AI, 2026)).

Full benchmark and model details are provided in Sec. B.2.

2.2 Rank-2 Geometry

Evidence for effective rank-2. If model capabilities lie in a low-dimensional space, the score matrix should be predictable from a low-rank completion. The operational question is therefore not only whether observed scores can be compressed, but which rank best predicts held-out benchmark scores. We establish rank 2 through two lines of evidence: *(i)* rank-2 matrix completion minimizes held-out prediction error in raw-score and logit-score spaces (Figure 3), and *(ii) Singular Value Decomposition* (SVD) of fully-observed submatrices shows matching rank-2 geometry.

Evidence 1: Rank-2 completion minimizes held-out prediction error. We use Soft-Impute (Mazumder et al., 2010), a standard matrix-completion method that alternates between filling missing entries and taking a low-rank SVD approximation. We sweep its rank in raw-score and logit-transformed score spaces; the latter linearizes percentage scores before standardization. We evaluate on held-out entries using Median Absolute Percentage Error (MedAPE), the median of absolute percentage errors $|\text{predicted} - \text{true}|/|\text{true}| \times 100\%$. Figure 3 shows the same pattern in both score spaces: held-out error is minimized at rank 2 and rises for higher ranks. Sec. B.3 shows that rank 2 also remains the best-performing choice in logit space when the newest models are held out as a block and predicted from older models.

Evidence 2: Complete submatrices show matching rank-2 geometry. Recall that the best rank-r approximation retains the top r singular values, and the *stable rank* $\|M\|_F^2/\sigma_1^2$ measures effective dimensionality: values near 1 mean that one component dominates. We mean-center each benchmark column before computing SVD, so that the leading component reflects directions of model variation rather than the shared average score level. We then take the largest fully-observed model subsets available at several benchmark-coverage levels and compute the SVD of each submatrix. Table 1 shows the same pattern across these different shapes: the spectra are dominated by a single direction, and two components explain more than 90% of the variance in every submatrix.

Taken together, the held-out completion sweep gives the operational reason to use rank 2, and the fully-observed SVDs show the matching geometry behind that choice:

Finding 1: The collected 84×133 model-benchmark matrix is effectively rank-2.

Interpretation of the top two factors. On the rank-2 completed, column-standardized matrix, the first factor tracks general capability: all 133 benchmarks load on it with the same sign, and its per-model scores match each model's mean z-scored observed score at Spearman 0.893. The second factor separates domains, with long-context and coding suites on one side and agentic-search and competition-math on the other; Sec. B.3 lists the models and benchmarks at both ends of each factor.

3 BenchPress: A Low-rank Benchmark Score Predictor

Because the held-out rank sweep in Sec. 2.2 selects rank 2, we now ask: can we turn this structure into a predictor for missing benchmark scores?

3.1 From Matrix Completion to BenchPress

Methods. We compare seven feature transforms and twelve prediction methods. *Feature transforms:* identity, log, logit, arcsinh, square root, probit, and quantile. *Prediction methods:* benchmark mean; model mean; Benchmark-KNN (Bench-KNN); Model-KNN; per-benchmark regression (BenchReg);

Table 2: Top-15 transform × method configurations ranked independently by held-out test error. # is per-metric rank; values are shown as metric value followed by coverage in parentheses, e.g. 7.8 (100%). All results use standardization and report the median over 10 seeds × 3 folds. Pink highlights mark the Logit Bias ALS configuration adopted as BENCHPRESS's main score predictor.

MedAPE (%) ↓					MedAE ↓				
#	Transform	Method	Hyperparameter	Value	#	Transform	Method	Hyperparameter	Value
1	Probit	ModelReg	$R^2_{\min}$=0.2, k=7	7.7 (82%)	1	Logit	Bias ALS	λ=0.01, r=2	4.62 (100%)
2	Probit	ModelReg	$R^2_{\min}$=0.1, k=5	7.7 (74%)	2	Probit	Bias ALS	λ=0.1, r=2	4.62 (100%)
3	Probit	ModelReg	$R^2_{\min}$=0.3, k=7	7.7 (82%)	3	**Logit**	**Bias ALS**	λ=0.1, r=2	**4.63 (100%)**
4	Probit	BenchReg	$R^2_{\min}$=0.2, k=7	7.7 (85%)	4	Probit	BenchReg	$R^2_{\min}$=0.3, k=7	4.64 (84%)
5	Probit	ModelReg	$R^2_{\min}$=0.1, k=7	7.7 (83%)	5	Logit	BenchReg	$R^2_{\min}$=0.3, k=7	4.66 (84%)
6	Probit	ModelReg	$R^2_{\min}$=0.3, k=5	7.7 (74%)	6	Quantile	Bias ALS	λ=0.1, r=2	4.66 (100%)
7	Probit	ModelReg	$R^2_{\min}$=0.2, k=5	7.7 (74%)	7	Probit	ModelReg	$R^2_{\min}$=0.3, k=5	4.66 (74%)
8	Probit	BenchReg	$R^2_{\min}$=0.3, k=7	7.8 (84%)	8	Probit	BenchReg	$R^2_{\min}$=0.2, k=7	4.66 (85%)
9	Logit	ModelReg	$R^2_{\min}$=0.3, k=5	7.8 (74%)	9	Probit	ModelReg	$R^2_{\min}$=0.2, k=5	4.67 (74%)
10	Probit	BenchReg	$R^2_{\min}$=0.1, k=7	7.8 (86%)	10	Probit	ModelReg	$R^2_{\min}$=0.1, k=5	4.67 (74%)
11	Logit	BenchReg	$R^2_{\min}$=0.1, k=7	7.8 (86%)	11	Logit	BenchReg	$R^2_{\min}$=0.1, k=7	4.67 (86%)
12	**Logit**	**Bias ALS**	λ=0.1, r=2	**7.8 (100%)**	12	Probit	ModelReg	$R^2_{\min}$=0.2, k=3	4.67 (60%)
13	Logit	BenchReg	$R^2_{\min}$=0.3, k=7	7.8 (84%)	13	Probit	ModelReg	$R^2_{\min}$=0.3, k=3	4.67 (60%)
14	Logit	BenchReg	$R^2_{\min}$=0.2, k=7	7.8 (85%)	14	Probit	ModelReg	$R^2_{\min}$=0.1, k=3	4.67 (60%)
15	Probit	Bias ALS	λ=0.1, r=2	7.8 (100%)	15	Logit	BenchReg	$R^2_{\min}$=0.2, k=7	4.67 (85%)

per-model regression (ModelReg); Soft-Impute (Mazumder et al., 2010); non-negative matrix factorization (NMF) (Lee & Seung, 1999); probabilistic matrix factorization (PMF) (Mnih & Salakhutdinov, 2008); nuclear norm minimization (Candès & Recht, 2009); bias-decomposed alternating least squares (ALS) (Koren et al., 2009); and a multilayer perceptron (MLP). Details are provided in Sec. C.1.

Experiment setting. We evaluate all 84 combinations of the seven feature transforms and twelve prediction methods. For each model, we randomly split its known benchmark scores into 3 disjoint folds; each fold hides one third, trains on the remaining two thirds (plus all other models' data), and predicts the hidden scores, over 10 seeds (26,040 test predictions per pair). Each pair applies the feature transform and column standardization, runs the prediction method, and inverts both steps to return original-scale predictions. Hyperparameters are selected independently for each pair by nested cross-validation on validation cells split from each fold's training data, so all reported errors are measured on held-out folds that take no part in selection; Sec. C.2 lists the hyperparameter grid.

Metrics. Using the notation from Sec. 2.1, we measure prediction quality with two score-error metrics: *(i) median absolute percentage error* (MedAPE ↓), computed within each held-out fold as the median of $|\hat{s} - s|/|s| \times 100\%$ and then summarized by the median over folds, and *(ii) median absolute error* (MedAE ↓), computed analogously from $|\hat{s} - s|$ in raw score points. We use median-based score-error metrics because the error distribution is heavy-tailed: a handful of near-zero scores inflate percentage errors and a few large misses would dominate the mean; the figures and reported numbers below likewise summarize per-record errors by their median. We also report *coverage*, the fraction of held-out entries for which a method produces a finite prediction, because some methods (e.g., regression) cannot predict when insufficient correlated data exists.

Results. Table 2 ranks the best-performing configurations by held-out test error under the two score-error metrics. Several patterns emerge: (i) BenchReg and ModelReg dominate the top score-error entries, but their coverage is not always complete: some cells are left blank rather than predicted. (ii) Among methods that predict every missing cell, Logit Bias ALS is the strongest and remains very close to the best regression entries. (iii) The leading configurations are not separated by a large qualitative gap: related logit/probit variants and regularization choices give similar performance. We adopt Logit Bias ALS with $\lambda = 0.1$ and rank 2 as BENCHPRESS's default score predictor because it has the lowest validation error among the full-coverage configurations in Sec. C.2. That subsection also reports the full 7×12 transform × method grid. Sec. B.3 sweeps the rank of BENCHPRESS from 0 to 10: pooled MedAE falls from 5.17 with offsets only to 4.59 at rank 2, and ranks 3 to 6 differ from rank 2 by less than 0.07 points, within model-bootstrap intervals.

Finding 2: Logit-transformed, bias-decomposed alternating least squares (ALS) matrix completion (Koren et al., 2009), with rank 2 and regularization 0.1, gives near-best score-prediction accuracy while predicting every missing model–benchmark score.

BENCHPRESS: Point Prediction Recipe

Given a partially observed model–benchmark score matrix, BENCHPRESS predicts missing cells as follows:
1. Transform percentage scores with the logit transform; leave non-percentage scores on their native scale.
2. Standardize each benchmark column using the observed training entries.
3. Fit bias-decomposed alternating least squares (ALS) matrix completion (Koren et al., 2009) with a global level, model offsets, benchmark offsets, and a rank-2 residual interaction ($\lambda = 0.1$).
4. Invert the standardization and feature transform to return predictions on the original benchmark scale.

3.2 BENCHPRESS VS. LLM-BASED BENCHMARK SCORE PREDICTORS

We compare BENCHPRESS with a five-shot GPT-5.5 predictor queried separately for each target cell, with and without public model and benchmark identities.

On the held-out cells of Sec. 3.1, we prompt GPT-5.5 with five peer examples per target cell: the models in that fold's training matrix that observe the target benchmark, share at least five visible benchmarks with the target model, and have the highest Pearson correlation to it over shared visible scores. The *informed* condition keeps the real model and benchmark names; the *blind* condition anonymizes them while preserving the scores and peer-example structure. Details are provided in Sec. C.3. Table 3 shows that the informed prompt is a strong baseline and confirms that a frontier LLM can often predict held-out benchmark scores from nearest-peer examples. But this is not the same capability as score prediction from the matrix alone: names give the LLM access to public model reputations, benchmark semantics, and possibly memorized leaderboard facts. The blind condition removes that channel and is therefore the diagnostic comparison. There, the LLM is close to BENCHPRESS but not a cheaper or more reliable replacement.

Finding 3: A frontier LLM can predict benchmark scores when real names are visible, but that advantage depends on model and benchmark memory. In the blind setting, BENCHPRESS has comparable accuracy and is more scalable because it fits the score matrix once rather than querying an LLM over target cells.

4 WHAT BENCHPRESS ENABLES FOR MODEL EVALUATION

With the default score predictor fixed, we now evaluate what BENCHPRESS enables across realistic model-evaluation tasks. Three questions guide this section. **First**, Sec. 4.1 asks which probe benchmarks a practitioner should run under a fixed evaluation budget so that BENCHPRESS best recovers the model's scorecard. **Second**, Sec. 4.2 asks whether BENCHPRESS's predicted scores preserve same-benchmark model rankings well enough that practitioners can use them to compare models. **Third**, Sec. 4.3 asks whether BENCHPRESS can still produce useful predictions from a small seed evaluation for a brand-new model released after the matrix was assembled. Unless otherwise stated, all analyses in this section use the default BENCHPRESS score predictor from Sec. 3.1.

4.1 BUDGETED SCORECARD RECOVERY

The rank-2 structure identified in Sec. 2.2 suggests that many benchmark scores can be inferred from carefully chosen evidence. We therefore ask: if a practitioner can run only a few benchmarks on a new model, which scores should be measured, and which scores can BENCHPRESS infer from them? We compare three selection strategies: cost-unaware greedy selection, where any benchmark may be chosen; cost-aware greedy selection, where probes must come from the low-cost allowlist; and a random baseline that applies one global random benchmark ordering to every target model. Both greedy variants build the probe set one benchmark at a time: at each iteration we try every remaining candidate, temporarily add it to the current probe set, and keep the candidate that minimizes pooled MedAE over every observed model-benchmark cell; reported errors are computed only on the unrevealed cells. Details are provided in Sec. D.1. Figure 1b shows how pooled MedAE decreases as the number of probe benchmarks grows, which indicates the following finding.

Finding 4: With only five probe benchmarks, BENCHPRESS predicts the remaining score profile to a pooled MedAE of 4.74; restricting probes to the low-cost allowlist still reaches 5.32.

Table 3: **LLM prediction depends on names.** GPT-5.5 predicts each held-out score from the target model's known scores and five similar models' scores, with real (✓) or anonymized (✗) model and benchmark names.

Predictor	Names	MedAPE	MedAE
GPT-5.5	✓	5.86	3.50
GPT-5.5	✗	7.89	4.70
BENCHPRESS	✗	**7.77**	**4.63**

Table 4: **Pairwise ranking preservation.** Fraction of model pairs whose predicted scores keep the true order, among pairs whose true scores differ by at least the margin (points); median across benchmarks.

Margin	BENCHPRESS	Model mean
0	83.8%	78.3%
1	86.3%	80.2%
2	88.0%	82.1%
5	92.1%	86.7%

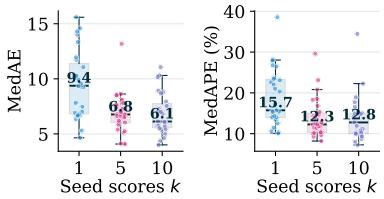


Figure 4: **Predicting newly released models under a prespecified temporal window.** Each dot is one target model; boxes show the median and interquartile range across 27 targets.

Sec. D.1 further shows that probes selected on part of the models still beat random probes on the other models, and that both the choice of probes and the rank-2 term contribute to the accuracy.

4.2 PRESERVING MODEL RANKINGS

The practical value of score prediction is not only numerical accuracy, but whether the predictions support the same evaluation decisions: when two models differ meaningfully on the same benchmark, does BENCHPRESS preserve which model is better? We reuse the holdout setting of Sec. 3.1 and, for each same-benchmark model pair with at least one held-out score, check whether the completed order matches the true order. We define *margin-aware pairwise accuracy* as the fraction of these pairs ordered correctly, counting only pairs whose true scores differ by at least a margin (0, 1, 2, or 5 points), since a small prediction error can flip a near-tied pair; Table 4 reports its median across benchmarks. Details are provided in Sec. D.2. We derive the following finding.

Finding 5: For same-benchmark model pairs separated by at least five score points, BENCHPRESS's predicted scores yield the correct ordering 92.1% of the time.

4.3 PREDICTING NEWLY RELEASED MODELS

The evaluations so far hide cells from the same matrix used to fit BENCHPRESS, so every model in the test set has already contributed training signal elsewhere in the matrix. Real deployment is stricter: when a new model is released, the matrix was assembled before that release and contains no information about the new model. We therefore ask whether BENCHPRESS can still produce useful scores for a brand-new model from the historical matrix plus a small seed evaluation on that model.

Experiment setting. We evaluate the 27 models released from the post-DeepSeek-R1 reasoning era through GPT-5.1 that have more than 20 observed benchmark scores, a temporal window chosen using only release metadata and matrix coverage before inspecting prediction errors. For each target model, we train BENCHPRESS on only the models released before the target's release date, reveal $k \in \{1, 5, 10\}$ of the target's observed benchmark scores, and predict the rest, repeating each setting over 10 random seeds and reporting the median; revealed cells are excluded from the metric. Details are provided in Sec. D.3.

Results. Two patterns stand out across the 27 targets in Figure 4. First, even with release-date cutoffs, revealing a small seed set sharply reduces prediction error: the median target drops from 9.37 MedAE at k=1 to 6.76 at k=5 and 6.12 at k=10. On the same cells, the logit-space model mean fit to the same training matrix and seed scores gives 8.59, 7.49, and 7.69: one seed score is too few to fit the rank-2 term, while from k=5 on BENCHPRESS is more accurate for 19 of the 27 targets. The corresponding MedAPE drops from 15.73% to 12.27% at k=5 and is 12.76% at k=10. Second, the distribution narrows once five seed scores are revealed, showing that the gain is not driven by only a few easy releases. In Sec. D.3, a prospective check against a later score matrix (data cutoff August 26, 2026) gives consistent errors: fit only on the main score matrix and evaluated on its 133 benchmarks, BENCHPRESS predicts the 178 newly reported scores of existing models with a MedAE of 4.44 points (logit-space model mean: 4.86) and, given whichever of the five cost-unaware greedy probes they report, the remaining scores of the 40 of 45 newly added models that report at least one probe

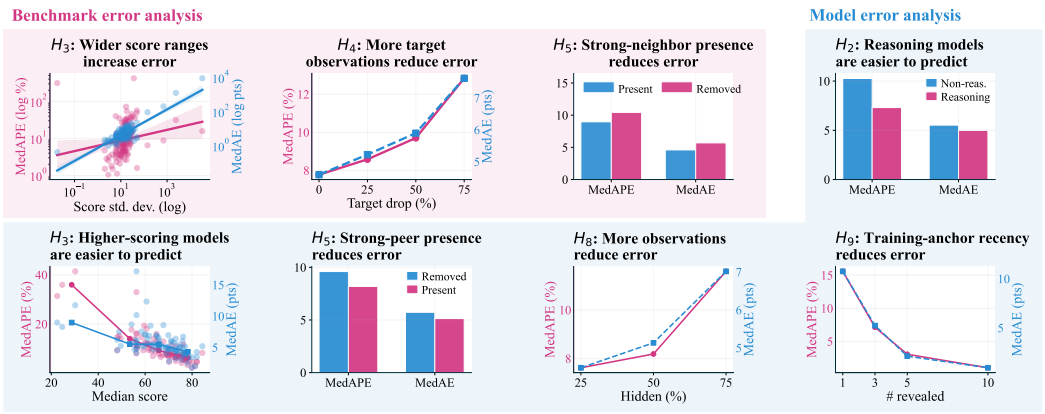


Figure 5: **Prediction-error patterns for benchmark and model analyses.** The top-left block shows the jointly supported benchmark-side effects: H3 score spread, H4 target coverage, and H5 strong-neighbor presence. The top-right and bottom blocks show the jointly supported model-side effects: H2 model type, H3 score level, H5 strong-peer presence, H8 observation count, and H9 training-anchor recency.

with a MedAE of 6.88 points, against 6.95 for the logit-space model mean and 10.15 for benchmark medians.

Finding 6: Across 27 target models in a pre-specified temporal window, five seed scores give BENCHPRESS a median absolute error of 6.76 points on the unrevealed scores, and ten seeds tighten this to 6.12 points.

5 WHEN TO TRUST BENCHPRESS'S PREDICTIONS

The practical question is not only whether BENCHPRESS can fill in missing scores, but when those filled-in scores are safe to use. This section explains when to trust the default BENCHPRESS score predictor. We first identify benchmark- and model-side factors associated with prediction quality, then use those signals together with predictor disagreement to estimate prediction reliability.

5.1 WHAT AFFECTS PREDICTION RELIABILITY

The natural next question is where prediction error comes from. Some sources of error may be benchmark-side: sparse observations, weak benchmark neighbors, or score distributions that are intrinsically hard to interpolate. Others may be model-side: sparse model rows, weak peers, provider effects, scale, or recency. We test seven benchmark-side and nine model-side hypotheses, each phrased as a no-effect claim about prediction quality measured by MedAPE and MedAE. We use Spearman rank correlation tests for observational hypotheses and paired Wilcoxon signed-rank tests for interventions, rejecting a no-effect hypothesis when $p < 0.01$. Small p-values indicate evidence against no effect; failure to reject does not establish that a factor has no effect. More experiment details are provided in Sec. E.1.

Benchmark analysis. What determines whether a benchmark is easy or hard for BENCHPRESS to predict? We test seven hypotheses split into two families. H1 to H3 concern low-rank fit, score level, and score spread; H4 to H7 concern target coverage, strong-neighbor presence, strong-neighbor support, and same-category evidence.

Three hypotheses are rejected under both metrics (Figure 5, top-left). Rejecting H3 (score spread) means benchmarks with wider score ranges across models are harder to predict. Rejecting H4 (target coverage) and H5 (strong-neighbor presence) means a benchmark is easier to predict when it has many observed model scores, and when at least one strongly correlated neighbor remains in the training matrix. The remaining hypotheses (H1 low-rank fit, H2 score level, H6 strong-neighbor support, H7 same-category evidence) are not rejected under both metrics.

Model analysis. Symmetrically, what determines whether a model is easy or hard for BENCHPRESS to predict? Per-model prediction error varies by $\sim$40$\times$ across the 84 models in our matrix, so this matters in practice: harder-to-predict models warrant less trust in their point estimates. We test nine

hypotheses split into three families. H1 to H4 concern model size, model type, score level, and low-rank fit; H5 to H8 concern strong-peer presence, strong-peer support, same-provider evidence, and observation count; H9 tests training-anchor recency.

Five hypotheses are rejected under both metrics (Figure 5, top-right and bottom). Reasoning models are easier to predict than non-reasoning ones (H2), and higher-scoring models are easier than lower-scoring ones (H3). Among the ablations, models with at least one strongly correlated peer in the matrix are easier to predict (H5), models with more observed benchmark scores are easier (H8), and the temporal experiment shows that prediction quality on newer models improves when the training matrix contains more recent anchors rather than only the oldest third (H9). The H5 result matches what Zhang et al. (2025b) find: predictions are only reliable when the target model performs similarly to the models the predictor was fit on. The remaining hypotheses (H1 model size, H4 low-rank fit, H6 strong-peer support, H7 same-provider evidence) are not rejected under both metrics.

Because we run 32 tests across both families (16 hypotheses under two metrics), we control the false discovery rate with the Benjamini-Hochberg correction (Benjamini & Hochberg, 1995) described in Sec. E.1: at $\alpha = 0.01$, all eight jointly-supported factors stay significant under both metrics.

Finding 7: A predicted score is most trustworthy when both sides of the matrix are well supported: the target benchmark has many observations and correlated neighbors, the target model has many observed scores and correlated peers, and the training matrix contains recent models near the target.

5.2 Estimating Prediction Reliability

To identify safe-to-use predictions before running a benchmark, we compare three reliability estimators for the fixed Logit Bias ALS predictor. They differ only in their features: (i) the *ensemble-spread reliability estimator* uses only disagreement among score predictors; (ii) the *matrix-support reliability estimator* instead reuses the model- and benchmark-side signals from Sec. 5.1; and (iii) the *hybrid reliability estimator* uses both feature groups in one model. All three use the same fold-internal model selection over a linear ridge baseline and small ReLU MLPs, so the risk score for a hidden cell is learned from other cells, not from its own error. Larger risk scores indicate less reliable predictions. Sec. E.2 calibrates the same risk score into a trust probability and a conformal prediction interval; throughout this subsection we report MedAE only, since the risk-coverage curve is measured on the absolute score-point scale.

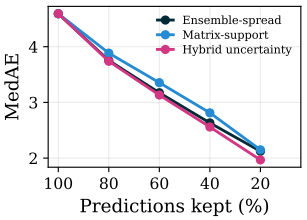


Figure 6: **Risk scores pick out accurate predictions.** When we keep only the lowest-risk predictions, MedAE falls as fewer are kept, from 4.6 to about 2 points.

Figure 6 plots MedAE as we retain only the held-out cells with the lowest predicted risk. At 20% retained, the hybrid estimator achieves a selective MedAE of 1.97 score points, outperforming both single-feature variants. We therefore use the **hybrid reliability estimator** as BenchPress's reliability layer.

Finding 8: A hybrid reliability estimator uses ensemble spread and matrix support to identify low-risk score predictions before running the benchmark.

6 Limitations and Future Work

First, BenchPress does not predict every score equally well: Sec. 5.1 shows that accuracy depends on support from observed scores and correlated neighbors. For a model with few scores or no correlated peers, published metadata such as architecture, model size, and training data could supply the similarity that scores cannot. *Second*, because BenchPress predicts only aggregate scores, it inherits benchmark noise and misses within-benchmark structure that instance-level outcomes could capture. *Third*, the matrix covers only text and vision-language benchmarks, so audio, embodied agents, and scientific simulators remain untested. *Fourth*, rank 2 describes the matrix at its data cutoff and should be rechecked as new models may add a third factor. *Fifth*, the factors tested in Sec. 5.1 are not exhaustive, and the observational ones, such as model type and score level, are correlational rather than causal. Untested attributes such as training recipe may also affect predictability.

AI Use Statement

In this work, we used generative AI tools to design and evaluate experiments, propose and refine hypotheses about what affects prediction reliability, implement methods and experiment code, and clean and reformat the collected benchmark-score records. GPT-5.5 is also a baseline score predictor in Sec. 3.2; this use is part of the study itself. We have not used generative AI tools to generate synthetic data, develop theoretical models or conceptual frameworks, formulate or prove mathematical claims, or interpret results; translation and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to search for information and identify relevant literature, create or edit software code, create or modify figures, suggest experimental parameters, draft and edit parts of the paper, and format references. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

Ethics Statement

This work uses only aggregate benchmark scores that are publicly reported in model release materials, technical reports, model cards, and leaderboards. It involves no human subjects, crowdsourced annotation, or personal data, and every released score records the URL of its source (Sec. B.1 shows the record format). The main risk is over-reliance on predicted scores. A predicted score does not replace running the benchmark, particularly for safety, fairness, or deployment-specific evaluation, and because the public matrix covers widely reported models and benchmarks more densely (Figure 2), predictions can inherit this coverage bias. Sec. 5 identifies what makes a prediction reliable and estimates, for each prediction, whether it is reliable enough to use or whether the benchmark should still be run.

Reproducibility Statement

Secs. 2.1 and B.1 describe how the score matrix is collected, canonicalized, and filtered, together with the released record format, and Sec. B.2 lists every benchmark and model in the matrix. Sec. 3.1 defines the held-out evaluation protocol, Secs. C.1 and C.2 specify every candidate transform, method, and hyperparameter grid, and each experiment in Secs. 4 and 5 states its own setting, with further details in Secs. D and E. Sec. C.3 gives the exact prompt for the LLM baseline. The score matrix and the code that reproduces every experiment and figure are available at https://anonymous.4open.science/r/benchpress-iclr2027/README.md.

References

Matthew Aitchison, Penny Sweetser, and Marcus Hutter. Atari-5: Distilling the arcade learning environment down to five games. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 421–438, 2023.

Anthropic. Claude Opus 4.7. https://www.anthropic.com/news/claude-opus-4-7, 2026.

ARC Prize. ARC Prize leaderboard. https://arcprize.org/leaderboard, 2026.

Mislav Balunović, Jasper Dekoninck, Ivo Petrov, Nikola Jovanović, and Martin Vechev. MathArena: Evaluating LLMs on uncontaminated math competitions. https://matharena.ai/, 2025.

Jan Batzner, Sree Harsha Nelaturu, Damian Stachura, et al. Every eval ever: A unifying schema and community repository for AI evaluation results. *arXiv preprint arXiv:2606.14516*, 2026.

Andrew M. Bean, Nabeel Seedat, Shengzhuang Chen, and Jonathan Richard Schwarz. Scales++: Compute efficient evaluation subset selection with cognitive scales embeddings. *arXiv preprint arXiv:2510.26384*, 2025.

Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B (Methodological)*, 57(1):289–300, 1995.

Stella Biderman, Hailey Schoelkopf, Lintang Sutawika, et al. Lessons from the trenches on reproducible evaluation of language models. *arXiv preprint arXiv:2405.14782*, 2024.

Ryan Burnell, Han Hao, Andrew R. A. Conway, and Jose Hernandez-Orallo. Revealing the structure of language model capabilities. *arXiv preprint arXiv:2306.10062*, 2023a.

Ryan Burnell et al. Rethink reporting of evaluation results in AI. *Science*, 2023b.

Greg Burnham. Benchmark scores = general capability + claudiness. Epoch AI Blog, 2025. URL https://epoch.ai/gradient-updates/benchmark-scores-general-capability-claudiness.

Emmanuel J. Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: An open platform for evaluating LLMs by human preference. In *ICML*, 2024.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro, 2026.

Epoch AI. FrontierMath. https://epoch.ai/benchmarks/frontiermath, 2026.

Adam Fisch, Daniel Deutsch, Joshua Maynez, Alekh Agarwal, Jonathan Berant, William Cohen, Amir Globerson, and Jacob Eisenstein. CollabEval: Statistically efficient collaborative model evaluation via matrix completion, 2026.

Google. Gemma 3. https://blog.google/technology/developers/gemma-3/, 2025.

Google DeepMind. Gemini 3.1 Pro model card. https://deepmind.google/models/model-cards/gemini-3-1-pro/, 2026.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *ICLR*, 2021.

Anson Ho et al. A rosetta stone for ai benchmarks. *arXiv preprint arXiv:2512.00193*, 2025.

Myles Hollander, Douglas A. Wolfe, and Eric Chicken. *Nonparametric Statistical Methods*. Wiley, Hoboken, NJ, 3rd edition, 2014.

David Ilić and Gilles E. Gignac. Evidence of interrelated cognitive-like capabilities in large language models: Indications of artificial general intelligence or achievement? *Intelligence*, 106:101858, 2024. doi: 10.1016/j.intell.2024.101858.

Alexander Kipnis et al. MetaBench: A sparse benchmark of reasoning and knowledge in large language models. In *ICLR*, 2025.

Woosung Koh, Juyoung Suk, Sungjun Han, Se-Young Yun, and Jamin Shin. Predicting LLM reasoning performance with small proxy model. In *ICLR*, 2026.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *Transactions on Machine Learning Research*, 2023.

LMArena. LMArena leaderboard. https://lmarena.ai/leaderboard, 2026.

Aviya Maimon, Amir David Nisan Cohen, Gal Vishne, Shauli Ravfogel, and Reut Tsarfaty. From benchmarks to skills: Low-rank factors for LLM evaluation. *Transactions of the Association for Computational Linguistics*, 14, 2026.

Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 11:2287–2322, 2010.

Mikhail Mirzayanov. Codeforces. https://codeforces.com/, 2009.

Andriy Mnih and Ruslan Salakhutdinov. Probabilistic matrix factorization. In *NeurIPS*, 2008.

Moonshot AI. Kimi K3. https://github.com/MoonshotAI/Kimi-K3, 2026.

Jinjie Ni, Fuzhao Xue, Xiang Yue, Yuntian Deng, Mahir Shah, Kabir Jain, Graham Neubig, and Yang You. MixEval: Deriving wisdom of the crowd from LLM benchmark mixtures. In *NeurIPS*, 2024.

OpenAI. Introducing GPT-5.5. https://openai.com/index/introducing-gpt-5-5/, 2026.

Lorenzo Pacchiardi, Konstantinos Voudouris, Ben Slater, Fernando Martínez-Plumed, José Hernández-Orallo, Lexin Zhou, and Wout Schellaert. PredictaBoard: Benchmarking LLM score predictability. In *Findings of ACL*, 2025.

Jungsoo Park, Ethan Mendes, Gabriel Stanovsky, and Alan Ritter. Look before you leap: Estimating LLM benchmark scores from descriptions. *arXiv preprint arXiv:2509.20645*, 2025.

Tejal Patwardhan et al. GDPval: Evaluating AI model performance on real-world economically valuable tasks. https://openai.com/index/gdpval/, 2025.

Yotam Perlitz, Elron Bandel, Ariel Gera, Ofir Arviv, Liat Ein-Dor, Eyal Shnarch, Noam Slonim, Michal Shmueli-Scheuer, and Leshem Choshen. Efficient benchmarking (of language models). In *Proceedings of NAACL*, pp. 2519–2536, 2024a.

Yotam Perlitz, Ariel Gera, Ofir Arviv, Asaf Yehudai, Elron Bandel, Eyal Shnarch, Michal Shmueli-Scheuer, and Leshem Choshen. Benchmark agreement testing done right: A guide for LLM benchmark evaluation. *arXiv preprint arXiv:2407.13696*, 2024b.

Felipe Maia Polo, Lucas Weber, Leshem Choshen, Yuekai Sun, Gongjun Xu, and Mikhail Yurochkin. tinybenchmarks: evaluating llms with fewer examples. In *International Conference on Machine Learning*, 2024.

Felipe Maia Polo, Seamus Somerstep, Leshem Choshen, Yuekai Sun, and Mikhail Yurochkin. Sloth: Scaling laws for LLM skills to predict multi-benchmark performance across families. In *NeurIPS*, 2025.

Qwen Team. Qwen3.7: The agent frontier. https://qwen.ai/blog?id=qwen3.7, 2026.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark. In *COLM*, 2024.

Yangjun Ruan, Chris J. Maddison, and Tatsunori B. Hashimoto. Observational scaling laws and the predictability of language model performance. In *NeurIPS*, 2024. Spotlight.

Alexander Rubinstein, Benjamin Raible, Martin Gubri, and Seong Joon Oh. DISCO: Diversifying sample condensation for efficient model evaluation. In *ICLR*, 2026.

Gayathri Saranathan, Cong Xu, Mahammad Parwez Alam, Tarun Kumar, Martin Foltin, Soon Yee Wong, and Suparna Bhattacharya. SubLIME: Subset selection via rank correlation prediction for data-efficient LLM evaluation. In *ACL*, 2025.

Viktoria Schram, Daniel Beck, and Trevor Cohn. Performance prediction via Bayesian matrix factorisation for multilingual natural language processing tasks. In *Proceedings of EACL*, pp. 1790–1801, 2023.

Terminal-Bench Team. Terminal-Bench 2.0. https://www.tbench.ai/, 2025.

Rajan Vivek, Kawin Ethayarajh, Diyi Yang, and Douwe Kiela. Anchor points: Benchmarking models with much fewer examples. In *Proceedings of EACL*, 2024.

Shaobo Wang et al. Rethinking LLM evaluation: Can we evaluate LLMs with 200× less data? In *ICLR*, 2026.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Siddartha Naidu, et al. LiveBench: A challenging, contamination-limited LLM benchmark. In *ICLR*, 2025.

Qinyuan Ye, Harvey Yiyun Fu, Xiang Ren, and Robin Jia. How predictable are large language model capabilities? a case study on big-bench. In *Findings of EMNLP*, 2023.

Guanhua Zhang, Ricardo Dominguez-Olmedo, and Moritz Hardt. Train-before-Test harmonizes language model rankings. *arXiv preprint arXiv:2507.05195*, 2025a.

Guanhua Zhang, Florian E. Dorner, and Moritz Hardt. How benchmark prediction from fewer data misses the mark. *arXiv preprint arXiv:2506.07673*, 2025b.

Qiyuan Zhang, Fuyuan Lyu, Xue Liu, and Chen Ma. Collaborative performance prediction for large language models. In *EMNLP*, 2024.

Jin Peng Zhou, Christian K. Belardi, Ruihan Wu, Travis Zhang, Carla P. Gomes, Wen Sun, and Kilian Q. Weinberger. On speeding up language model evaluation. In *ICLR*, 2025.

Appendix

The appendix is organized as supplements to the main text. Sec. A supplements Sec. 1 with the experiment setting for Figure 1. Sec. B supplements Sec. 2 with data-collection provenance, full benchmark and model catalogs, and additional evidence for the low-rank structure. Sec. C supplements Sec. 3 with the full candidate-method specification, comparisons to observational scaling laws and independent score matrices, and the additional LLM baseline prompt template. Sec. D supplements Sec. 4 with budgeted scorecard recovery, ranking preservation, and temporal-deployment details. Sec. E supplements Sec. 5 with prediction-error and prediction-reliability details. Sec. F reviews related work on efficient evaluation and benchmark score prediction.

Contents

A Supplemental to Sec. 1: Introduction . . . 15
- A.1 Experiment Setting for Figure 1 . . . 15
- A.2 Later Score Matrix (Data Cutoff August 26, 2026) . . . 15

B Supplemental to Sec. 2: The Score Matrix and Its Geometry . . . 15
- B.1 Data Collection . . . 15
- B.2 The Final Score Matrix . . . 19
- B.3 Rank-2 Geometry . . . 23

C Supplemental to Sec. 3: BENCHPRESS: A Low-rank Benchmark Score Predictor . . . 24
- C.1 Candidate Methods . . . 24
- C.2 From Matrix Completion to BENCHPRESS . . . 28
- C.3 BENCHPRESS vs. LLM-Based Benchmark Score Predictors . . . 31
- C.4 Comparison to Observational Scaling Laws . . . 32
- C.5 BENCHPRESS on Other Score Matrices . . . 33

D Supplemental to Sec. 4: What BENCHPRESS Enables for Model Evaluation . . . 33
- D.1 Budgeted Scorecard Recovery . . . 33
- D.2 Preserving Model Rankings . . . 38
- D.3 Predicting Newly Released Models . . . 39

E Supplemental to Sec. 5: When to Trust BENCHPRESS's Predictions . . . 41
- E.1 What Affects Prediction Reliability . . . 41
- E.2 Estimating Prediction Reliability . . . 48

F Related Work . . . 52

A Supplemental to Sec. 1: Introduction

A.1 Experiment Setting for Figure 1

Left panel (per-cell error). We pick four highlighted cells: (GPT-5.6 Sol, BrowseComp), (Claude Opus 5, ARC-AGI-1), (Phi-4 Reasoning Plus, GPQA Diamond), and (DeepSeek-V4-Pro, Terminal-Bench 2.0). For each cell and each $k \in \{1, \ldots, 10\}$, we (i) hide the target cell, (ii) keep k scores sampled uniformly at random from the same model's other observed cells and hide the rest of its row, (iii) feed the resulting masked matrix to BenchPress, and (iv) record the absolute error on the held-out target cell. We repeat over 10 seeds (base seed 42); the line is the per-cell median and the shaded band is the 25–75 percentile range. The diamond at k=0 marks the benchmark-median baseline (no BenchPress). GPT-5.6 Sol and Claude Opus 5 were released in July 2026, so their cells use the later score matrix described in Sec. A.2, with a data cutoff of August 26, 2026; the other two cells use the main score matrix (data cutoff April 30, 2026). Phi-4 Reasoning Plus has 11 observed benchmarks in the main score matrix, below the 15-benchmark threshold, so we append its row to that matrix.

Right panel (overall pooled error). The right panel reuses the global probe-set setting of Secs. 4.1 and D.1: a fixed probe set of k benchmarks is chosen, every model is evaluated on whichever probe scores it has observed, and BenchPress predicts the rest of each model's observed cells. Pooled MedAE is reported on these predicted cells; revealed probe cells are excluded. The greedy curves use the cost-aware and cost-unaware MedAE orderings from Table 16; the gray random baseline draws one global random benchmark ordering per seed (10 seeds, base seed 42) and uses the corresponding prefix for every target model. The shaded band is the 25–75 percentile range across seeds.

A.2 Later Score Matrix (Data Cutoff August 26, 2026)

Two cells in the left panel of Figure 1, the method comparison in Sec. C.5, and the prospective check in Sec. D.3 use a later score matrix, collected with the same pipeline up to a data cutoff of August 26, 2026. It contains 283 models, 712 benchmarks, and 8,713 primary scores, together with 2,756 additional source-specific score candidates retained alongside the primary records. Requiring at least 15 observed benchmarks per model and eight observed models per benchmark yields a 129×253 matrix with 4,905 observed cells (15.0% filled), compared with the 84×133 matrix with 2,604 observed cells used in the main experiments.

B Supplemental to Sec. 2: The Score Matrix and Its Geometry

B.1 Data Collection

Sec. 2.1 summarizes how we crawl public model releases, technical reports, model cards, and primary leaderboards; this appendix gives the full collection procedure, the information preserved in the released data, the limitations of the resulting raw matrix, and how we canonicalize and filter it.

Collection procedure. Our data collection proceeds in four steps. *First*, we seed a queue with a small initial set of models (GPT-5.5 (OpenAI, 2026), Claude Opus 4.7 (Anthropic, 2026), Gemini 3.1 Pro (Google DeepMind, 2026), DeepSeek-V4-Pro (DeepSeek-AI, 2026), and a few other widely-discussed recent releases) and crawl every official source attached to each: the release blog, the system card, the technical report, and the Hugging Face model card. *Second*, we recurse: each source typically reports the model's own scores together with a handful of competitor baselines, so any newly mentioned model is added to the queue and crawled in turn, until no further round introduces an unvisited model. *Third*, we sweep a fixed list of primary leaderboards (MathArena (Balunović et al., 2025), ARC-Prize (ARC Prize, 2026), Terminal-Bench (Terminal-Bench Team, 2025), LMArena (LMArena, 2026), Epoch AI (Epoch AI, 2026), LiveBench (White et al., 2025)) to fill remaining gaps for models and benchmarks that vendors do not directly cover. *Fourth*, we filter the resulting raw matrix to a dense subset that supports the analyses in the rest of the paper.

When the same model-benchmark cell is reported by multiple sources we resolve conflicts as best we can: we keep the highest-priority value, with priority order release blog, then system card, then

technical report, then HuggingFace model card, then primary leaderboards, breaking ties by recency. We also fix one canonical configuration per model (typically the setting the vendor itself foregrounds, such as a specific reasoning effort level) to avoid inflating coverage with effectively duplicate rows from near-duplicate variants. Every retained value carries the URL it was sourced from (the record format is shown below), and roughly half come from a source outside the model's own provider, such as a competitor's release materials or a leaderboard. Alongside the score itself, each entry records the model's release date, provider, and canonical evaluation setting (mode, reasoning effort, sampling, judge, harness, prompt style, temperature, tool use), and each benchmark records its category, metric, problem count, and reference link, so downstream analyses can condition on release timing, evaluation regime, or benchmark type without re-crawling the sources.

Here we make explicit what information is preserved in the released data, because later analyses reuse these fields without re-crawling the original sources. The released data keeps three linked record types: one record per model, one record per benchmark, and one record per observed model–benchmark score.

Example model record


```
{
"id": "gpt-5.2",
"name": "GPT-5.2",
"provider": "OpenAI",
"release_date": "2025-12-11",
"is_reasoning": true,
"open_weights": false,
"canonical_setting": {
"mode": "thinking",
"effort": "xhigh",
"tools": "none",
"sampling": "pass@1",
"judge": "rule-based",
"harness": "official",
"prompt_style": "default"
}
}
```


Example benchmark record


```
{
"id": "aime_2025",
"name": "AIME 2025",
"category": "Math",
"metric": "% correct (pass@1)",
"num_problems": 30,
"source_url":
"https://artofproblemsolving.com/wiki/index.php/2025_AIME",
"canonical_setting": {
"version": "AIME-2025-I+II",
"metric_type": "pct",
"range": [0, 100],
"higher_is_better": true,
"multimodal_input": false,
"tools": "none"
}
}
```

Example observed cell record


```
{
"model_id": "gpt-5.2",
"benchmark_id": "aime_2025",
"score": 100.0,
"reference_url": "https://openai.com/index/introducing-gpt-5-2/",
"source_type": "official_blog",
"audit_status": "verified",
"reported_setting": {
"mode": "thinking",
"effort": "xhigh",
"tools": "none",
"sampling": "pass@1",
"judge": "rule-based",
"harness": "official"
},
"matches_canonical": true,
"candidates": [{"score": 99.0, "source_type": "model_card"}, ...]
}
```


This structure separates entity metadata from score provenance. Benchmark-level fields, such as item count and modality, support analyses that reason about the columns of the matrix. Model-level fields, such as provider, release date, and reasoning capability, support analyses that reason about the rows. Cell-level fields keep the audit trail for the actual number used in the matrix: where it came from, how the model was run, whether that setting matches the canonical setting, and which alternative values were seen but not selected as the primary score.

The raw score matrix suffers from the following problems.

- **Evaluation heterogeneity.** Scores come from vendor reports and independent third parties and may differ in prompting strategy, reasoning budget, sampling temperature, evaluation harness, benchmark version, and evaluation date. Vendor-reported scores may be optimistically biased relative to independent reproductions. Even with identical prompts, non-deterministic decoding can change scores by 1 to 3 points across runs, while different harnesses can shift the same benchmark score by more than 5 points (Biderman et al., 2024).
- **Structured missingness.** Popular models $\times$ popular benchmarks are over-represented, violating the uniform-sampling assumption underlying standard matrix-completion guarantees.

We allow these effects to remain for two reasons. First, our goal is to predict roughly where a model lands on a benchmark, not to reproduce an exact score. This is enough for a practitioner deciding whether to run the evaluation. Second, the public record is heterogeneous by nature; if we required a single controlled setting, too few scores would remain to work with. Reducing these effects still improves accuracy, so the filtering procedure below keeps one canonical setting per model and one canonical column per benchmark family, then removes rows and columns with too few observations.

Filtering. The raw matrix at this point (data cutoff April 30, 2026) contains 188 models and 316 benchmarks but only 4,493 of the 59,408 cells are filled (7.6%), with the long tail dominated by barely-observed rows and columns. This raw audit pool is useful for provenance, but it is not yet the analysis matrix: some rows or columns are alternate views of the same underlying signal. We first canonicalize these cases. For model setting variants, we keep one representative row rather than making one mode trivially predictable from another. For benchmark variants, we keep one canonical column per task family; same-scale versions may fill missing canonical cells as non-canonical measurements, while a variant on a different scale never fills a canonical cell. Variants that change the evaluation protocol, such as the item subset, tool access, input modality, harness, grader, score unit, or context length (for example, HLE Text, HLE with tools, Aider Polyglot in whole-file mode, SWE-Lancer in dollars, or MRCR v2 with eight needles), remain separate benchmarks. This yields a canonicalized pool with 181 models, 304 benchmarks, and 4,177 observed cells. The canonicalized pool is still very sparse, so we then filter to a dense subset by requiring every retained model to be observed on at least a minimum number of benchmarks, and every retained benchmark to be observed on at least a minimum number of models. Table 5 shows how the number of retained entries and the fill rate change across candidate thresholds. But there is no single definition of the best

threshold: a looser threshold keeps more data but leaves the matrix sparser, while a stricter threshold creates a denser matrix by discarding more data. We therefore use 15 observations per model and 8 per benchmark as a practical balance for all analyses in this paper, but other choices are equally reasonable. Prediction results remain similar across the full grid, with several stricter thresholds giving lower error (Table 6). The resulting matrix has 84 models $\times$ 133 benchmarks with 2,604 observed cells (23.3% fill). These protocol variants make prediction only slightly easier: keeping the most-observed member of each of 23 variant groups (95 of 133 columns) raises the held-out MedAE of BENCHPRESS on those 95 columns from 4.69 to 4.98 under the protocol of Sec. 3.1. The stored alternative source values also show how precisely a cell is known: on the 361 percentage-scale cells whose sources report at least two distinct values, the chosen and alternative values differ by a median of 3.2 points.

Table 5: **The threshold filter trades coverage for density.** Each row applies a minimum-observation requirement to model rows and benchmark columns, iterated to a fixed point. We use the bolded setting throughout.

Min. obs. per		Resulting matrix			
Model	Bench.	#Models	#Bench.	#Obs.	Fill
Raw audit pool		188	316	4,493	7.6%
10	8	130	141	3,201	17.5%
10	12	124	104	2,795	21.7%
10	16	112	70	2,246	28.6%
15	**8**	**84**	**133**	**2,604**	**23.3%**
15	12	61	81	1,811	36.7%
15	16	53	53	1,337	47.6%
20	8	49	110	1,885	35.0%
20	12	41	69	1,353	47.8%

Filtering-threshold robustness. The density filter trades matrix coverage for a more fully observed analysis block. To test whether the adopted minimum of 15 benchmark observations per model and 8 model observations per benchmark was selected because it yields favorable prediction error, we apply the fixed BENCHPRESS predictor to the complete threshold grid under the same 10-seed, three-fold per-model holdout protocol. Each row in Table 6 is evaluated on the observed cells retained by that threshold setting. Prediction error remains comparable across the grid, while several stricter filters yield lower error by discarding more sparsely observed rows and columns. The adopted setting therefore balances matrix coverage and density rather than minimizing reported error.

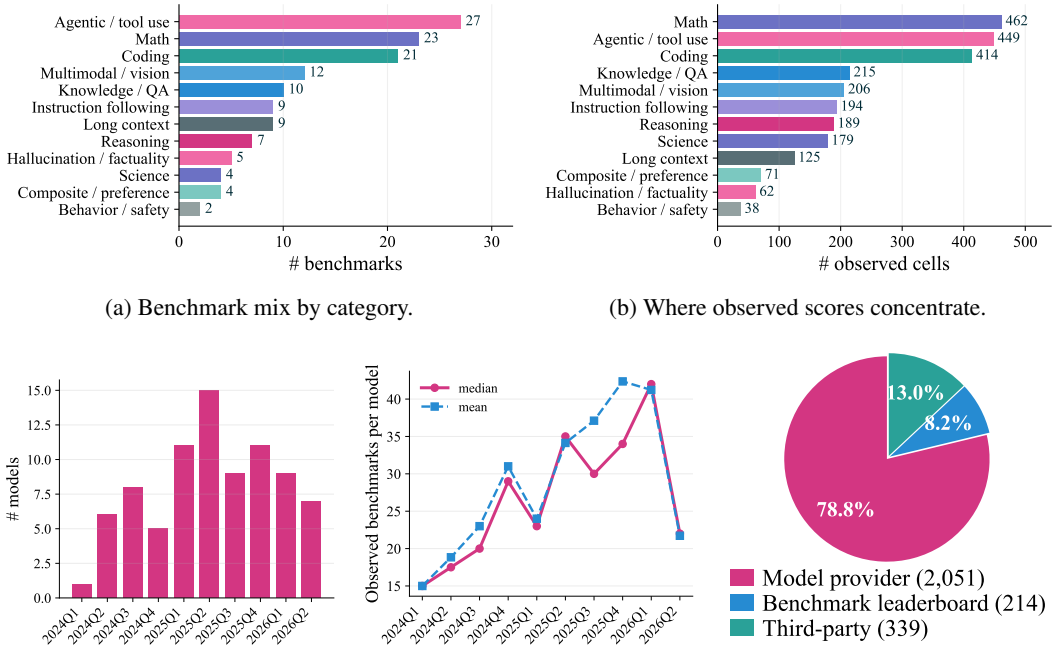
Table 6: **Prediction error remains comparable across filtering thresholds.** Each row applies the fixed logit-space rank-2 Bias ALS predictor ($\lambda = 0.1$) and reports median error under 10 seeds and three per-model folds on that row's retained observed cells. "Canonicalized, no density filter" is evaluated after model and benchmark variants are consolidated but before minimum-observation filtering; bold marks the adopted setting.

Min. obs. per		#Obs.	Fill	MedAE	MedAPE
Model	Bench.			↓	↓
Canonicalized, no density filter		4,177	7.6%	4.92	8.44%
10	8	3,201	17.5%	4.64	7.67%
10	12	2,795	21.7%	4.52	7.52%
10	16	2,246	28.6%	4.79	7.79%
15	**8**	**2,604**	**23.3%**	**4.63**	**7.77%**
15	12	1,811	36.7%	3.92	6.59%
15	16	1,337	47.6%	3.83	6.23%
20	8	1,885	35.0%	4.05	7.16%
20	12	1,353	47.8%	3.91	6.68%
20	16	Empty after filtering			

B.2 The Final Score Matrix

Sec. 2.1 introduced the benchmark score matrix as an 84×133 table with s_{mb} the score of model m on benchmark b, populated from publicly available evaluations and 23.3% filled. This appendix provides the complete benchmark and model inventories underlying that matrix, together with its composition and coverage. Figure 7 summarizes what this adopted matrix contains: a broad benchmark mix, with observed cells concentrated in math, coding, agentic/tool-use, and knowledge-oriented evaluations; a model set concentrated in recent releases, with coverage varying by release time; and score provenance dominated by model-provider materials, with the remainder split between benchmark leaderboards and third-party aggregators. Table 7 lists all 133 benchmarks with their categories, metrics, item counts, and source links. Table 8 enumerates all 84 retained models with parameter counts, reasoning capability, open-weight status, release dates, and source links. Every score in the matrix is attributed to one of these sources; the full (model, benchmark) $\rightarrow$ URL mapping is released with the accompanying repository.

Matrices used in the experiments. The main experiments use the score matrix with a data cutoff of April 30, 2026, which after filtering contains 84 models and 133 benchmarks (Figure 2); the unfiltered audit pool covers 188 models across 316 benchmarks. Sec. C.5 confirms that the same predictor remains effective on the later score matrix described in Sec. A.2, with a data cutoff of August 26, 2026, and on independently built HELM (Liang et al., 2023) and Every Eval Ever (EEE) (Batzner et al., 2026) matrices.



(a) Benchmark mix by category.

(b) Where observed scores concentrate.

(c) Model releases over time.

(d) Coverage by model release time.

(e) Source provenance of observed cells.

Figure 7: **Composition and coverage of the adopted score matrix** (84 models $\times$ 133 benchmarks, 2,604 observed cells, 23.3% fill). The matrix spans a broad mix of benchmark categories (a), with most observed cells in math, coding, agentic/tool-use, and knowledge-oriented evaluations (b). Models are concentrated in recent releases (c), and coverage varies by release time because newer models are often reported on different benchmark suites than older baselines (d). Roughly four in five scores come from model providers' materials, with the remainder split between benchmark leaderboards and third-party aggregators (e).

Table 7: **Benchmark inventory.** All 133 benchmarks in the adopted score matrix. Categories are grouped to match the main-text summary.

Category	Benchmark	Metric	Items	Link
Agentic & tool use (26)	BFCL	—	—	
	BFCL v3	—	—	
	BrowseComp	% correct	1,266	
	BrowseComp-ZH	—	1,156	
	ComplexFuncBench	%	1,000	
	CyberGym	% solved	1,507	
	DeepSearchQA (Accuracy)	%	900	
	Finance Agent v1.1	% solved	537	
	FinSearchComp-Global	%	317	
	Frames	%	824	
	GAIA (text only)	%	103	
	MCPAtlas Public	% correct (pass@1)	500	
	MCPMark	% success (pass@1)	127	
	OSWorld	% success	369	
	tau-bench Airline	% success	50	
	Tau-Bench Retail	% success	115	
	Terminal-Bench 1.0	% solved	—	
	Terminal-Bench 2.0	% solved	—	
	Toolathlon	% correct (pass@1)	108	
	Vending-Bench 2	—	15,000	
	WideSearch (item-F1)	%	200	
	xbench-DeepSearch	%	100	
	τ^2-bench Airline	% success	50	
	τ^2-bench Retail	% success	115	
	τ^2-bench Telecom	% success	114	
	τ^3-Bench	%	1,500	
Math (23)	AIME 2024	% correct (pass@1)	30	
	AIME 2025	% correct (pass@1)	30	
	AIME 2026	% correct (pass@1)	30	
	Beyond AIME	%	100	
	BRUMO 2025	% correct (pass@1)	30	
	CMIMC 2025	% correct (pass@1)	40	
	CNMO 2024	%	6	
	FrontierMath	% correct T1-3	300	
	FrontierMath Tier 4	%	48	
	GSM8K	% correct	1,319	
	HMMT Feb 2025	%	30	
	HMMT Feb 2026	% correct (pass@1)	33	
	HMMT Nov 2025	% correct	30	
	IMO-AnswerBench	—	400	
	MATH	—	12,500	
	MATH-500	% correct	500	
	MathArena Apex 2025	% correct	12	
	MathVision	% correct	3,040	
	MathVista	%	1,000	
	MGSM	exact match (%)	2,500	
	MT-AIME2024	%	1,650	
	SMT 2025	% correct (pass@1)	53	
	USAMO 2025	% of 42 points	6	
Coding (21)	Aider Polyglot (diff mode)	%	450	
	Aider Polyglot (whole mode)	%	450	
	ArtifactsBench	%	5,475	
	BigCodeBench	pass@1 %	1,140	
	Bird-SQL (Dev)	—	—	
	Codeforces Rating	Elo rating	—	
	HumanEval	pass@1 %	164	
	LiveCodeBench	pass@1 %	1,055	
	MBPP+	—	—	
	Multi-SWE-bench	%	1,632	
	MultiPL-E (average)	%	12,667	
	NL2Repo-Bench	%	104	
	OJBench	%	232	
	RepoQA	—	500	
	SciCode	% correct	338	
	SWE-bench Multilingual	% resolved	—	
	SWE-bench Pro	% resolved	731	
	SWE-bench Verified	% resolved	500	
	SWE-Lancer IC Diamond	%	198	
	SWE-Lancer IC SWE Diamond Freelance ($)	dollars	198	

Category	Benchmark	Metric	Items	Link
	Terminal-Bench Hard	%	—	
Multimodal & vision (12)	BabyVision	% accuracy	388	
	CharXiv Descriptive	% accuracy	4,000	
	CharXiv Reasoning	% accuracy	1,000	
	ERQA	%	400	
	MMMU	% correct	900	
	MMMU-Pro	% correct	3,460	
	OmniDocBench (normalized edit distance, lower is better)	edit distance (lower=better)	1,651	
	OmniDocBench 1.5	edit distance (lower=better)	1,355	
	ScreenSpot-Pro	—	1,581	
	Vibe-Eval	—	269	
	Video-MME	—	2,700	
	Video-MMMU	%	900	
Long context (9)	AA Long Context Reasoning	% correct	300	
	BrowseComp Long Context 128k	% accuracy	1,266	
	GraphWalks BFS 0-128K	%	300	
	GraphWalks parents 0-128K	%	350	
	LongBench-V2	%	503	
	MRCR v1	—	2,000	
	MRCR v2	% correct	2,400	
	OpenAI MRCR v2 (2 needle, 128k)	%	500	
	OpenAI MRCR v2 (8-needle)	%	800	
Instruction following (9)	Arena-Hard Auto	% win rate	500	
	COLLIE	%	2,080	
	IFBench	% correct	300	
	IFEval	% correct (prompt strict)	541	
	InFoBench	—	2,250	
	Internal API IF Hard	%	—	
	Multi-IF	%	13,503	
	MultiChallenge	%	273	
	MultiChallenge (o3-mini grader)	%	273	
Knowledge & QA (9)	C-Eval (Chinese)	%	12,342	
	Chinese-SimpleQA	%	3,000	
	GDPval (Artificial Analysis ELO)	score	220	
	HealthBench	%	5,000	
	MMLU-Pro	% correct	12,032	
	MMMLU	% correct	258,090	
	PopQA	—	14,267	
	SimpleQA	% correct	4,326	
	SimpleQA-Verified	% correct (pass@1)	—	
Reasoning (8)	ARC-AGI-1	% correct	400	
	ARC-AGI-2	% correct	400	
	BigBench Hard (BBH)	—	—	
	DROP	%	9,536	
	Global PIQA	—	6,283	
	HLE (Humanity's Last Exam)	% correct	2,500	
	HLE (w/ tools)	accuracy (%)	2,500	
	HLE Text	%	2,158	
Hallucination & factuality (5)	FACTS Grounding	—	1,719	
	FActScore (hallucination rate)	%	500	
	LongFact-Concepts (hallucination rate)	%	1,140	
	LongFact-Objects (hallucination rate)	%	1,140	
	TruthfulQA	—	817	
Science (4)	CritPt	% correct	70	
	GPQA Diamond	% correct	198	
	GPQA Main (full set)	—	448	
	SuperGPQA	%	26,529	
Other (7)	AA Intelligence Index	index score	12,826	
	AlpacaEval 2.0 (LC-winrate)	%	—	
	Bullshit-Bench (Clear Pushback)	% clear pushback	55	
	Chatbot Arena Elo	Elo rating	8,000	
	CLUEWSC	%	2,574	
	LiveBench	overall score	1,000	
	Safety (OLMES suite)	—	—	

Table 8: **Model inventory.** All 84 models from 13 providers. *R* = reasoning (chain-of-thought). *O* = open-weight. Parameter counts in billions; "— = undisclosed. Active parameters shown only for MoE models.

Provider	Model	B	Act.	R	O	Rel.
OpenAI	GPT-3.5 Turbo (0125)	—	—	✗	✗	2024-01
	GPT-4o (2024-05-13)	—	—	✗	✗	2024-05
	GPT-4o mini	—	—	✗	✗	2024-07
	GPT-4o (2024-11-20)	—	—	✗	✗	2024-11
	OpenAI o1 (high)	—	—	✓	✗	2024-12
	o3-mini (high)	—	—	✓	✗	2025-01
	GPT-4.5	—	—	✗	✗	2025-02
	GPT-4.1	—	—	✗	✗	2025-04
	GPT-4.1 mini	—	—	✗	✗	2025-04
	GPT-4.1 nano	—	—	✗	✗	2025-04
	o3 (high)	—	—	✓	✗	2025-04
	o4-mini (high)	—	—	✓	✗	2025-04
	GPT-5 mini	—	—	✓	✗	2025-07
	GPT-5 nano	—	—	✓	✗	2025-07
	GPT-5	—	—	✓	✗	2025-08
	gpt-oss-120B	116.8	5.1	✓	✓	2025-08
	GPT-5.1	—	—	✓	✗	2025-11
	GPT-5.2	—	—	✓	✗	2025-12
	GPT-5.4	—	—	✓	✗	2026-03
	GPT-5.5	—	—	✓	✗	2026-04
Google	Gemini 1.5 Flash	—	—	✗	✗	2024-05
	Gemini 1.5 Pro	—	—	✗	✗	2024-05
	Gemma 2 27B	27	27	✗	✓	2024-06
	Gemma 2 9B	9	9	✗	✓	2024-06
	Gemma 3 1B	—	—	✓	✓	2025
	Gemini 2.0 Flash	—	—	✗	✗	2025-02
	Gemma 3 27B	27	27	✗	✓	2025-03
	Gemini 2.5 Flash	—	—	✓	✗	2025-05
	Gemini 2.5 Pro (GA)	—	—	✓	✗	2025-06
	Gemini 3 Flash	—	—	✓	✗	2025-11
	Gemini 3 Pro	—	—	✓	✗	2025-11
	Gemini 3.1 Pro	—	—	✓	✗	2026-02
Anthropic	Claude 3.5 Sonnet (1022)	—	—	✗	✗	2024-10
	Claude 3.7 Sonnet	—	—	✓	✗	2025-02
	Claude Opus 4	—	—	✓	✗	2025-05
	Claude Sonnet 4	—	—	✗	✗	2025-05
	Claude Opus 4.1	—	—	✓	✗	2025-08
	Claude Sonnet 4.5	—	—	✓	✗	2025-09
	Claude Haiku 4.5	—	—	✓	✗	2025-10
	Claude Opus 4.5	—	—	✓	✗	2025-11
	Claude Opus 4.6	—	—	✓	✗	2026-02
	Claude Sonnet 4.6	—	—	✓	✗	2026-02
	Claude Opus 4.7	—	—	✓	✗	2026-04

Provider	Model	B	Act.	R	O	Rel.
Alibaba	Qwen2.5 72B Instruct	—	—	✗	✓	2024-09
	Qwen2.5-14B	14	14	✗	✓	2024-09
	Qwen2.5-32B-Instruct	32	32	✗	✓	2024-09
	Qwen2.5-7B-Instruct	7	7	✗	✓	2024-09
	QwQ-32B	32.8	32.8	✓	✓	2025-03
	Qwen3-235B-A22B	235	22	✓	✓	2025-05
	Qwen3-30B-A3B	30	3	✓	✓	2025-05
	Qwen3-32B	32	32	✓	✓	2025-05
	Qwen3-8B	8	8	✓	✓	2025-05
	Qwen3.5-397B-A17B	397	17	✓	✓	2026-02
	Qwen3.6-Plus	—	—	✓	✗	2026-03
DeepSeek	DeepSeek-V2-0506	—	—	✗	✓	2024-05
	DeepSeek-V2.5-0905	—	—	✗	✓	2024-09
	DeepSeek-V3	671	37	✗	✓	2025-01
	DeepSeek-R1	671	37	✓	✓	2025-01
	DeepSeek-R1-Distill-Llama-70B	70	70	✓	✓	2025-01
	DeepSeek-R1-0528	671	37	✓	✓	2025-05
	DeepSeek-V3.2	671	37	✓	✓	2025-12
	DeepSeek-V4-Flash	284	13	✓	✓	2026-04
	DeepSeek-V4-Pro	1600	49	✓	✓	2026-04
Meta	LLaMA-3.1 405B Instruct	—	—	✗	✓	2024-07
	Llama 3.1 8B Instruct	8	8	✗	✓	2024-07
	Llama 3.2 1B	—	—	✓	✓	2024-09
	Llama-3.3-70B-Instruct	70	70	✗	✓	2024-12
	Llama 4 Maverick	402	17	✗	✓	2025-04
	Muse Spark	—	—	✓	✗	2026-04
Zhipu AI	GLM-4.6	—	—	✓	✓	2025-09
	GLM-4.7	—	—	✓	✓	2025-12
	GLM-5	—	—	✓	✓	2026-03
	GLM-5.1	—	—	✓	✓	2026-04
Moonshot AI	Kimi K2	—	—	✗	✓	2025-07
	Kimi K2.5	—	—	✓	✓	2026-01
	Kimi K2.6	—	—	✓	✓	2026-04
xAI	Grok 3 Beta	—	—	✗	✗	2025-02
	Grok 4	—	—	✓	✗	2025-07
	Grok 4.1	—	—	✓	✗	2025-11
MiniMax	MiniMax-M2	—	—	✓	✓	2025-10
	MiniMax M2.1	—	—	✓	✓	2025-12
Cohere	Command A	111	—	✗	✓	2025-03
ByteDance	Doubao Seed 2.0 Pro	—	—	✓	✗	2026-02
Mistral	Ministral 8B Instruct 2410	8	8	✗	✓	2024-10

B.3 RANK-2 GEOMETRY

Rank selection under held-out models. The rank sweep in Sec. 2.2 selects each rank on the canonical per-model folds, where every model contributes some training scores. A skeptical reading is that the rank-2 minimum could come from copying a near-duplicate sibling row of the target rather than from genuine low-rank structure, and Figure 3 reports no uncertainty. We therefore repeat the sweep under a stricter split: we hold out the newest 17 models (the most recent 20% by release date) as a block, removing their rows from the training matrix entirely, and predict each held-out model on its own from the older models plus two thirds of its own observed scores, so no other recent model is available to copy from. Figure 8 plots held-out MedAPE against rank with 95% cluster-bootstrap intervals over the 17 target models.

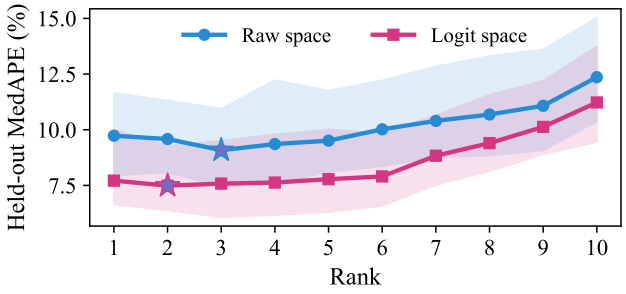


Figure 8: **Rank sweep with the newest 20% of models held out as a block.** Held-out MedAPE (%) versus Soft-Impute rank, in raw and logit score space, with 95% cluster-bootstrap intervals over the 17 held-out models; stars mark each curve's minimum. In logit space (the space BENCHPRESS uses) rank 2 remains the minimum even though no sibling of a target model is in the training matrix; in raw space ranks 2 and 3 fall within each other's interval.

Low rank continues to predict best under this split: the logit-space minimum stays at rank 2, and the raw-space curve is flat across the first three ranks, whose intervals overlap. The rank-2 geometry is therefore not an artifact of near-duplicate sibling rows.

Rank of BENCHPRESS. The sweeps above use Soft-Impute. We also sweep the rank of BENCHPRESS itself from 0, where only the global level and the model and benchmark offsets are fit, to 10, with $\lambda = 0.1$ on the same 30 folds. Pooled MedAE is 5.17 at rank 0, 4.80 at rank 1, and 4.59 at rank 2; it stays between 4.53 and 4.62 through rank 6 and rises to 5.11 at rank 10. Under a bootstrap over the 84 models (1,000 resamples), rank 2 is more accurate than rank 0 by 0.58 points (95% interval $[0.43, 0.72]$) and than rank 1 by 0.21 points ($[0.11, 0.31]$), while the intervals for ranks 3 to 6 against rank 2 all contain zero. Rank 2 is therefore the smallest rank whose error cannot be distinguished from the best rank.

Top two factors. We complete the matrix at rank 2 and take the SVD of the completed, column-standardized scores; Table 9 lists the top and bottom models and benchmarks on each of the two factors interpreted in Sec. 2.2. The first component tracks general capability: frontier models sit at the top and small or older models at the bottom. All 133 benchmarks load on it with the same sign and with similar magnitude (loadings run from 0.042 to 0.097, against 0.087 for a perfectly uniform direction), so no single benchmark or family drives it, and its per-model scores match the naive capability ranking (each model's mean z-scored observed score) at Spearman 0.893. Since this component is computed on the completed matrix, we also check it against the observed entries alone: using only the scores actually reported, 131 of 133 benchmarks correlate positively with the first-component model score. The second component instead separates domains. Its benchmark loadings take both signs, with long-context and coding suites on one side and agentic-search and competition-math on the other, and it does not order models by strength (for example, Claude 3.5 and 3.7 Sonnet rank above GPT-5.5). Its coding-heavy side matches the task-specialized components in Ruan et al. (2024, Figure 2), where PC2 emphasizes mathematical and coding benchmarks and PC3 primarily reflects programming capability. Zhang et al. (2025a) show that this second factor can

also be related to task preparation: how much a model was trained on the kind of task a benchmark measures.

Table 9: **Top and bottom models and benchmarks on each of the top two factors**, from the SVD of the rank-2 completed, column-standardized score matrix. PC1 orders models by overall capability, with frontier models at the top and small or older models at the bottom, and loads every benchmark with the same sign; PC2 does not order models by capability and instead splits benchmarks by domain.

	PC1 (general capability)	PC2 (domain lean)
Top-5 models	GPT-5.5, Claude Opus 4.7, Gemini 3.1 Pro, GPT-5.4, DeepSeek-V4-Pro	Claude 3.7 Sonnet, Claude Opus 4.1, Llama 4 Maverick, Claude Opus 4, Claude 3.5 Sonnet
Bottom-5 models	Llama 3.2 1B, Gemma 3 1B, GPT-3.5 Turbo, Gemma 2 9B, Llama 3.1 8B	Ministral 8B, Gemma 3 1B, GPT-4.1 nano, Llama 3.2 1B, GPT-5.4
Top-5 benchmarks	MultiChallenge, BFCL v3, Vibe-Eval, SWE-Lancer Freelance, Multi-IF	MRCR v1, BigCodeBench, RepoQA, MBPP+, AA Intelligence Index
Bottom-5 benchmarks	OmniDocBench 1.5, FACTS Grounding, Safety, MathArena Apex, MRCR v1	xbench-DeepSearch, HMMT Feb 2026, BrowseComp-ZH, CyberGym, AIME 2026

C Supplemental to Sec. 3: BenchPress: A Low-rank Benchmark Score Predictor

C.1 Candidate Methods

This appendix gives the formal definitions of the feature transforms and matrix-completion methods compared in Sec. 3.1.

Suppose a new model arrives with scores on k benchmarks. How do we predict its scores on the remaining benchmarks? We decompose the problem into two design choices: (i) a *feature transform* that reshapes raw scores into a space amenable to linear methods, and (ii) a *prediction method* that exploits correlations across benchmarks or models.

Feature transforms. Let $s \in [0, 100]$ denote a benchmark score. Scores are bounded percentages, and models cluster near the ceiling on easy benchmarks and near the floor on hard ones. We evaluate seven transforms that address this nonlinearity in different ways:

- *Identity.* Use scores s as-is, with no transformation.
- *Log.* Apply $\log(s + 1)$, compressing high scores and stretching low ones.
- *Logit.* Apply $\log\big(s/(100 - s)\big)$, mapping the bounded score range to an unbounded scale that symmetrically spreads apart scores near both 0 and 100.
- *Arcsinh.* Apply $\operatorname{arcsinh}(s/50)$, a smooth approximation to log that is defined at zero.
- *Square root.* Apply $\sqrt{s}$, a mild compression that reduces the influence of high scores.
- *Probit.* Apply $\Phi^{-1}(s/100)$, where Φ is the standard normal CDF. Similar to logit but with lighter tails.
- *Quantile.* Replace each score with its within-benchmark rank divided by $n + 1$, producing uniform marginals. Non-parametric but discards magnitude information.

Transforms that assume a $[0, 100]$ range (logit, probit, arcsinh, square root) are applied only to benchmarks whose metadata declares a percentage scale. Eight columns are left untransformed: the seven non-percentage benchmarks (AA Intelligence Index, Chatbot Arena Elo (Chiang et al., 2024), Codeforces rating (Mirzayanov, 2009), GDPval (Artificial Analysis ELO) (Patwardhan et al., 2025), LiveBench (White et al., 2025), SWE-Lancer IC SWE Diamond Freelance ($), and Vending-Bench 2) and HLE (w/ tools), whose metadata declares no scale. For logit and probit, scores are clipped slightly away from the endpoints before transformation to avoid infinite values. After applying the chosen transform, we standardize each benchmark column to zero mean and unit variance, so every

prediction method operates entirely in the transformed, standardized space. After prediction, we invert the pipeline in reverse order: first undo the standardization (restoring each column's stored mean and standard deviation), then apply the inverse feature transform (e.g., sigmoid for logit) to map predictions back to the original score scale. We clip the final predictions of SWE-Lancer IC SWE Diamond Freelance ($) to $[0, 200{,}000]$ and of Vending-Bench 2 to $[0, 100{,}000]$, their declared ranges, and those of percentage-scale benchmarks to $[0, 100]$; the other untransformed columns are left unconstrained. Lower-is-better columns (FActScore, LongFact-Concepts, LongFact-Objects, and OmniDocBench) are not sign-flipped; they enter as recorded and are transformed, standardized, predicted, and inverted like the other percentage-scale columns.

Prediction methods. We compare the following methods, each evaluated across multiple feature transforms:

- *Benchmark mean.* Predict each missing score as the column average. No tunable parameters.
- *Model mean.* Predict each missing score as the row average. No tunable parameters.
- *Benchmark-KNN* (*Bench-KNN*). For each missing entry, find the k benchmarks most correlated with the target benchmark and predict from the model's observed scores on those neighbors, using correlation-based weights. Hyperparameter: k (number of neighbors).
- *Model-KNN.* Find the k models closest to the target model by root-mean-square distance over shared observed benchmarks, then average their scores on the target benchmark. Hyperparameter: k.
- *Per-benchmark regression* (*BenchReg*). For each target benchmark, BenchReg selects the k most correlated predictor benchmarks, fits one univariate regression per predictor benchmark, and combines the available predictions with R^2 weights. Targets and predictor pairs with fewer than five observations are skipped. When a model lacks observations on some predictors, BenchReg uses only the observed predictors; if none are observed, the cell is left unpredicted (coverage $< 100\%$). We use an ensemble of univariate regressions rather than a single multivariate model because the number of shared observations per benchmark pair is often very small (5–12 models), making joint estimation of k coefficients prone to overfitting. Hyperparameters: $k \in \{3, 5, 7\}$, $R^2_{\min} \in \{0.1, 0.2, 0.3\}$.
- *Per-model regression* (*ModelReg*). ModelReg is the row-wise counterpart to BenchReg. For each target model, it selects the k most correlated predictor models over shared benchmarks, fits univariate regressions from each predictor model's benchmark scores to the target model's scores, and combines the resulting predictions with R^2 weights. Like BenchReg, it skips targets and predictor pairs with fewer than five observations and can leave a cell unpredicted when no usable predictor has enough shared observations. Hyperparameters: $k \in \{3, 5, 7\}$, $R^2_{\min} \in \{0.1, 0.2, 0.3\}$.
- *Soft-Impute*. Soft-Impute (Mazumder et al., 2010) iterates between SVD truncation at a chosen rank and re-imputation of missing entries until convergence. We fix the rank to 2 following the held-out rank sweep in Sec. 2.2 (no tunable hyperparameters).
- *NMF.* Non-negative matrix factorization (Lee & Seung, 1999), constraining both factors to be non-negative. Hyperparameter: rank r.
- *PMF.* Probabilistic matrix factorization (Mnih & Salakhutdinov, 2008) with Gaussian priors on both factors. Hyperparameter: rank r.
- *Nuclear norm minimization.* Convex relaxation of rank minimization (Candès & Recht, 2009): minimize the nuclear norm of the completed matrix plus a squared-error fit on observed entries, with λ trading off low rank against data fidelity. Hyperparameter: λ.
- *Bias-decomposed alternating least squares (ALS)* (Koren et al., 2009). Bias ALS is the full-coverage method that we later adopt. We fix the rank to 2 following the held-out rank sweep in Sec. 2.2; the only tunable hyperparameter is the regularization λ.
- *Neural baseline* (*MLP*). A 2-layer MLP (hidden dimension 32) with binary mask for missing entries, trained for 500 epochs. Hyperparameter: learning rate.

Full definitions of all prediction methods, including equations and fallback rules, are given below. All methods below operate on the same transformed, column-standardized matrix.

We use the following notation throughout this subsection:

- $X \in \mathbb{R}^{M \times B}$ is the transformed, column-standardized version of the adopted score matrix, with $M = 84$ models and $B = 133$ benchmarks.
- x_{mb} is the entry for model m on benchmark b in this transformed space.

- Ω is the set of observed cells.
- $\Omega_m = \{b : (m, b) \in \Omega\}$ is the set of benchmarks observed for model m.
- $\Omega^b = \{m : (m, b) \in \Omega\}$ is the set of models observed for benchmark b.
- $\bar{x}_{m\cdot} = |\Omega_m|^{-1} \sum_{b \in \Omega_m} x_{mb}$ is the observed mean for model m.
- $\bar{x}_{\cdot b} = |\Omega^b|^{-1} \sum_{m \in \Omega^b} x_{mb}$ is the observed mean for benchmark b.
- $\bar{x} = |\Omega|^{-1} \sum_{(m,b) \in \Omega} x_{mb}$ is the observed global mean.
- $\hat{x}_{mb}$ is a method's prediction for cell (m, b).
- R is the rank used by low-rank methods in this subsection.
- Throughout the paper, "correlation" refers to the Pearson correlation: for two columns a, b of length n, $\rho(a, b) = \frac{\sum_i (a_i - \bar{a})(b_i - \bar{b})}{\sqrt{\sum_i (a_i - \bar{a})^2 \cdot \sum_i (b_i - \bar{b})^2}} \in [-1, 1]$, where $\bar{a}, \bar{b}$ are the column means.
- $N_k(t; \rho)$ is a method-local top-k neighbor set. For benchmark targets, $t = b \in \{1, \ldots, B\}$ and candidate neighbors are $u \in \{1, \ldots, B\} \setminus \{b\}$; for model targets, $t = m \in \{1, \ldots, M\}$ and candidate neighbors are $u \in \{1, \ldots, M\} \setminus \{m\}$. The scoring function $\rho(t, u) \in \mathbb{R}$ ranks each eligible candidate u for target t; larger scores are preferred, so distances are used with a minus sign.

Benchmark mean. The benchmark-mean baseline predicts each missing cell from the observed mean of the target benchmark:

$$\hat{x}_{mb} = \bar{x}_{\cdot b}.$$

Model mean. The model-mean baseline predicts each missing cell from the observed mean of the target model:

$$\hat{x}_{mb} = \bar{x}_{m\cdot}.$$

Bench-KNN. Let $\Omega^{bb'} = \Omega^b \cap \Omega^{b'}$, and let $\mathrm{corr}(b, b') \in [-1, 1]$ be the Pearson correlation between benchmark columns b and b' over $\Omega^{bb'}$ when this correlation is defined. Here $N_k(b; \mathrm{corr})$ selects benchmark neighbors $b' \neq b$ using score $\rho(b, b') = \mathrm{corr}(b, b')$. For a missing cell (m, b), Bench-KNN predicts by the correlation-weighted average over $N_k(b; \mathrm{corr}) \cap \Omega_m$:

$$\hat{x}_{mb} = \frac{\sum_{b' \in N_k(b;\mathrm{corr}) \cap \Omega_m} \max(\mathrm{corr}(b, b'), 0.01) x_{mb'}}{\sum_{b' \in N_k(b;\mathrm{corr}) \cap \Omega_m} \max(\mathrm{corr}(b, b'), 0.01)}.$$

If $N_k(b; \mathrm{corr}) \cap \Omega_m$ is empty, it falls back to $\bar{x}_{\cdot b}$.

Model-KNN. Let $\Omega_{mm'} = \Omega_m \cap \Omega_{m'}$. For each model pair (m, m') with $|\Omega_{mm'}| \geq 3$, define the shared-benchmark distance function $\Delta(m, m') \in \mathbb{R}_{\geq 0}$ by

$$\Delta(m, m') = \sqrt{\frac{1}{|\Omega_{mm'}|} \sum_{b \in \Omega_{mm'}} (x_{mb} - x_{m'b})^2}.$$

Here $N_k(m; -\Delta)$ selects model neighbors $m' \neq m$ using score $\rho(m, m') = -\Delta(m, m')$. For a missing cell (m, b), Model-KNN predicts by the average over $N_k(m; -\Delta) \cap \Omega^b$:

$$\hat{x}_{mb} = \frac{1}{|N_k(m; -\Delta) \cap \Omega^b|} \sum_{m' \in N_k(m; -\Delta) \cap \Omega^b} x_{m'b}.$$

If $N_k(m; -\Delta) \cap \Omega^b$ is empty, it falls back to $\bar{x}_{\cdot b}$.

BenchReg. For each target benchmark b and predictor benchmark b', BenchReg fits a one-dimensional linear predictor $f_{bb'} : \mathbb{R} \to \mathbb{R}$ on $\Omega^{bb'} = \Omega^b \cap \Omega^{b'}$. Here $R^2(f_{bb'}) \in \mathbb{R}$ is the coefficient of determination[1] of this linear fit on the shared observations. For BenchReg, $N_k(b; R^2)$ selects

[1]We use the coefficient of determination $R^2 = 1 - \mathrm{SSE}/\mathrm{SST}$, where $\mathrm{SST} = \sum_i (y_i - \bar{y})^2$ is the total variance of the target around its mean $\bar{y}$ and $\mathrm{SSE} = \sum_i (y_i - \hat{y}_i)^2$ is the residual variance left by the fit, with y_i the values we are trying to predict (the target benchmark's observed scores across shared models in the univariate-regression case) and $\hat{y}_i$ the corresponding predicted values. $R^2 = 1$ means a perfect fit ($\mathrm{SSE} = 0$), $R^2 = 0$ means the fit does no better than predicting the constant mean $\bar{y}$, and $R^2 < 0$ means it does worse than the constant mean.

benchmark neighbors $b' \neq b$ with $|\Omega^{bb'}| \geq 5$ and $R^2(f_{bb'}) \geq R^2_{\min}$ using score $\rho(b, b') = R^2(f_{bb'})$. For a missing cell (m, b), BenchReg predicts by the R^2-weighted average over $N_k(b; R^2) \cap \Omega_m$:

$$\hat{x}_{mb} = \frac{\sum_{b' \in N_k(b;R^2)\cap\Omega_m} R^2(f_{bb'})\, f_{bb'}(x_{mb'})}{\sum_{b' \in N_k(b;R^2)\cap\Omega_m} R^2(f_{bb'})}.$$

If $N_k(b; R^2) \cap \Omega_m$ is empty, BenchReg leaves the cell unpredicted.

ModelReg. For each target model m and predictor model m', ModelReg fits a one-dimensional linear predictor $f_{mm'} : \mathbb{R} \to \mathbb{R}$ on $\Omega_{mm'} = \Omega_m \cap \Omega_{m'}$. Here $R^2(f_{mm'}) \in \mathbb{R}$ is the coefficient of determination of this linear fit on the shared benchmarks. For ModelReg, $N_k(m; R^2)$ selects model neighbors $m' \neq m$ with $|\Omega_{mm'}| \geq 5$ and $R^2(f_{mm'}) \geq R^2_{\min}$ using score $\rho(m, m') = R^2(f_{mm'})$. For a missing cell (m, b), ModelReg predicts by the R^2-weighted average over $N_k(m; R^2) \cap \Omega^b$:

$$\hat{x}_{mb} = \frac{\sum_{m' \in N_k(m;R^2)\cap\Omega^b} R^2(f_{mm'})\, f_{mm'}(x_{m'b})}{\sum_{m' \in N_k(m;R^2)\cap\Omega^b} R^2(f_{mm'})}.$$

If $N_k(m; R^2) \cap \Omega^b$ is empty, ModelReg leaves the cell unpredicted.

Soft-Impute. Soft-Impute (Mazumder et al., 2010) initializes missing cells and then alternates between a rank-R SVD projection (a hard rank-R truncation rather than singular-value soft-thresholding) and clamping the observed entries:

$$X^{(\ell+1)}_{mb} = x_{mb} \quad \text{for } (m, b) \in \Omega, \qquad X^{(\ell+1)}_{\Omega^c} = \left[\mathrm{SVD}_R(X^{(\ell)})\right]_{\Omega^c}.$$

The fixed point gives predictions $\hat{x}_{mb} = X^{(\infty)}_{mb}$.

Bias-decomposed ALS. Bias-decomposed ALS models each entry as a global level plus a model offset a_m, a benchmark offset c_b, and a rank-R residual correction $UV^\top$, with $a \in \mathbb{R}^M$, $c \in \mathbb{R}^B$, $U \in \mathbb{R}^{M\times R}$, and $V \in \mathbb{R}^{B\times R}$. The offsets and factors are fitted jointly by

$$(\mu, a, c, U, V) = \arg\min_{\mu,a,c,U,V} \sum_{(m,b)\in\Omega} \left[x_{mb} - \mu - a_m - c_b - (UV^\top)_{mb}\right]^2 + \lambda\left(\|a\|_2^2 + \|c\|_2^2 + \|U\|_F^2 + \|V\|_F^2\right),$$

where the global level μ is not regularized. Its prediction is

$$\hat{x}_{mb} = \underbrace{\mu}_{\text{global level}} + \underbrace{a_m}_{\text{model } m \text{ offset}} + \underbrace{c_b}_{\text{benchmark } b \text{ offset}} + \underbrace{(UV^\top)_{mb}}_{\text{rank-}R\text{ residual correction}}.$$

The correction satisfies $UV^\top \in \mathbb{R}^{M\times B}$ and has rank at most R because U and V each have R columns. The offsets absorb row and column effects, so the low-rank term only has to model residual model–benchmark interaction structure. Because a and c are fitted jointly on the observed entries rather than set to observed row and column means, the rank-0 variant is a two-way additive model and differs from the model mean whenever strong and weak models report different benchmarks. ALS updates (a_m, U_m) for each model and (c_b, V_b) for each benchmark in closed form via ridge regression on observed entries and sets μ to the mean residual; we ensemble-average over multiple random initializations to reduce sensitivity to local minima. The default predictor uses rank $R = 2$, $\lambda = 0.1$, and averages the completed matrices from 10 random initializations (seeds 42–51).

NMF. Non-negative matrix factorization (NMF) (Lee & Seung, 1999) first shifts each benchmark column, if needed, so that the observed entries are non-negative. Writing the shifted observed entries as $x'_{mb} \in \mathbb{R}_{\geq 0}$, it solves

$$(U, V) = \arg\min_{\substack{U\in\mathbb{R}_+^{M\times R}\\ V\in\mathbb{R}_+^{B\times R}}} \sum_{(m,b)\in\Omega} \left[x'_{mb} - (UV^\top)_{mb}\right]^2 + \lambda(\|U\|_F^2 + \|V\|_F^2),$$

then subtracts the column shifts from $UV^\top$ to obtain predictions in the original transformed space.

PMF. Probabilistic matrix factorization (PMF) (Mnih & Salakhutdinov, 2008) places Gaussian priors on both factors and uses the same factor-matrix dimensions without the non-negativity constraint:

$$(U, V) = \arg \min_{\substack{U \in \mathbb{R}^{M \times R} \\ V \in \mathbb{R}^{B \times R}}} \sum_{(m,b) \in \Omega} \left[x_{mb} - (UV^\top)_{mb}\right]^2 + \lambda(\|U\|_F^2 + \|V\|_F^2),$$

with prediction $\hat{x}_{mb} = (UV^\top)_{mb}$.

Nuclear norm minimization. The nuclear-norm baseline (Candès & Recht, 2009) solves the convex low-rank surrogate

$$Z^\star = \arg \min_{Z \in \mathbb{R}^{M \times B}} \frac{1}{2} \sum_{(m,b) \in \Omega} (Z_{mb} - x_{mb})^2 + \lambda \|Z\|_*,$$

and predicts $\hat{x}_{mb} = Z^\star_{mb}$. Here λ trades off low rank against data fidelity.

Neural baseline. Let $\tilde{x}_m \in \mathbb{R}^B$ be row m with missing entries filled by zero, and let $o_m \in \{0, 1\}^B$ be its binary observation mask. The MLP baseline trains a two-layer network f_θ with masked reconstruction loss, where $\odot$ denotes elementwise multiplication:

$$\min_\theta \sum_m \left\| o_m \odot (f_\theta(\tilde{x}_m) - \tilde{x}_m) \right\|_2^2,$$

and predicts $\hat{x}_{mb} = [f_\theta(\tilde{x}_m)]_b$, averaged over three random seeds.

C.2 From Matrix Completion to BenchPress

This appendix lists the hyperparameter grid used in Sec. 3.1 and expands its head-to-head comparison from a small selected set into the full transform $\times$ method grid, reported both as a heatmap and as a sortable table.

The full transform $\times$ method grid (Figure 9) evaluates all 84 combinations on a common experiment setting; Table 11 reports the same numbers in tabular form, sorted by MedAPE. Each (transform, method) pair follows a four-stage pipeline: (i) apply the feature transform, (ii) standardize each column, (iii) run the prediction method, (iv) invert both transforms to recover original-scale predictions. Predictions on percentage-scale benchmarks are clipped to $[0, 100]$; omitting standardization degrades most methods, especially NMF and PMF.

Hyperparameter grid. For each (transform, method) pair, Sec. 3.1 grid-searches over the following hyperparameters:

- *Benchmark Mean, Model Mean.* No tunable parameters.
- *Bench-KNN, Model-KNN.* Number of neighbors $k \in \{3, 5, 7, 10\}$.
- *BenchReg, ModelReg.* Number of predictors $k \in \{3, 5, 7\}$, minimum correlation $R^2_{\min} \in \{0.1, 0.2, 0.3\}$ (9 configurations each).
- *Soft-Impute.* No tunable hyperparameters; rank fixed at 2.
- *Bias-decomposed ALS.* Regularization $\lambda \in \{0.01, 0.1, 1.0\}$; rank fixed at 2.
- *NMF.* Rank r in $\{1, 2, 3, 5\}$.
- *PMF.* Rank r in $\{1, 2, 3, 5\}$.
- *Nuclear Norm.* Regularization $\lambda \in \{0.1, 0.5, 1.0, 5.0\}$.
- *MLP.* Learning rate $\in \{10^{-4}, 10^{-3}, 10^{-2}\}$; architecture fixed at 2 layers with hidden dimension 32 and 500 training epochs.

Model selection. Sec. 3.1 selects each pair's hyperparameter on validation cells rather than on the held-out folds it reports, using nested cross-validation. The outer folds are unchanged. Within each outer fold we draw a second per-model partition of the training cells and use two of its three parts as validation cells, so the inner training set retains roughly the same fraction of the outer training cells that the outer training set retains of the matrix. Every candidate configuration is fit on the inner training cells and scored on the validation cells, and the configuration with the lowest validation MedAPE is then evaluated on the untouched outer test cells. Table 10 ranks the 203 configurations

that achieve 100% coverage on the outer held-out cells. Within this full-coverage set, the pipeline, seeds, metrics, and aggregation match Table 2; only the scored cells change from outer test to inner validation. Table 11 and Figure 9 report each pair under this nested selection, and where the winning hyperparameter varies across outer folds they report the one selected most often.

Unit and tail errors. Restricted to the 24,720 held-out cells on percentage-scale benchmarks, BENCHPRESS has a MedAE of 4.37 and the logit-space model mean 6.97. Over all 26,040 held-out cells, the 90th percentile of absolute error is 19.73 for BENCHPRESS and 27.69 for the logit-space model mean, and 24.9% and 39.5% of predictions miss by more than 10 points. With five probes, the corresponding tail fractions are 25.6% versus 40.6% for cost-unaware greedy and 29.1% versus 42.4% for cost-aware greedy; on the new models of Sec. D.3 they are 33.8% versus 38.6%.

Table 10: Top-15 transform × method configurations ranked independently by validation error among the 203 configurations with 100% coverage on the outer held-out cells. The full-coverage filter is applied before ranking. Within this set, the prediction pipeline, seeds, metrics, and aggregation match Table 2; each configuration is fit on inner training cells and scored on inner validation cells. # is per-metric rank; values are shown as metric value followed by validation coverage, rounded to the nearest whole percent, in parentheses. All results use standardization and report the median over 10 seeds × 3 outer folds × 2 inner validation folds. Pink highlights mark the Logit Bias ALS configuration adopted as BENCHPRESS's main score predictor.

Validation MedAPE (%) ↓					Validation MedAE ↓				
#	Transform	Method	Hyperparameter	Value	#	Transform	Method	Hyperparameter	Value
1	**Logit**	**Bias ALS**	λ=0.1, r=2	**8.9 (100%)**	1	**Logit**	**Bias ALS**	λ=0.1, r=2	**5.20 (100%)**
2	Probit	Bias ALS	λ=0.1, r=2	9.0 (100%)	2	Probit	Bias ALS	λ=0.1, r=2	5.29 (100%)
3	Quantile	Bias ALS	λ=0.1, r=2	9.0 (100%)	3	Quantile	Bias ALS	λ=0.1, r=2	5.39 (100%)
4	Logit	Bias ALS	λ=0.01, r=2	9.3 (100%)	4	Logit	Bias ALS	λ=0.01, r=2	5.39 (100%)
5	Quantile	Bias ALS	λ=0.01, r=2	9.3 (100%)	5	Quantile	Bias ALS	λ=0.01, r=2	5.45 (100%)
6	Logit	Bias ALS	λ=1.0, r=2	9.4 (100%)	6	Probit	Bias ALS	λ=0.01, r=2	5.53 (100%)
7	Probit	Bias ALS	λ=0.01, r=2	9.4 (100%)	7	Logit	Bias ALS	λ=1.0, r=2	5.62 (100%)
8	Probit	Bias ALS	λ=1.0, r=2	9.5 (100%)	8	Probit	Bias ALS	λ=1.0, r=2	5.78 (100%)
9	Identity	Bias ALS	λ=0.1, r=2	9.5 (100%)	9	Logit	Soft-Impute	r=2	5.86 (100%)
10	Identity	Bias ALS	λ=1.0, r=2	9.8 (100%)	10	Probit	Soft-Impute	r=2	5.88 (100%)
11	Quantile	Bias ALS	λ=1.0, r=2	9.8 (100%)	11	Quantile	Bias ALS	λ=1.0, r=2	5.89 (100%)
12	Quantile	Soft-Impute	r=2	9.9 (100%)	12	Quantile	Soft-Impute	r=2	5.93 (100%)
13	Probit	Soft-Impute	r=2	10.0 (100%)	13	Identity	Bias ALS	λ=0.1, r=2	5.95 (100%)
14	Logit	Soft-Impute	r=2	10.0 (100%)	14	Square root	Bias ALS	λ=0.1, r=2	6.09 (100%)
15	Identity	Bias ALS	λ=0.01, r=2	10.0 (100%)	15	Identity	Bias ALS	λ=0.01, r=2	6.14 (100%)

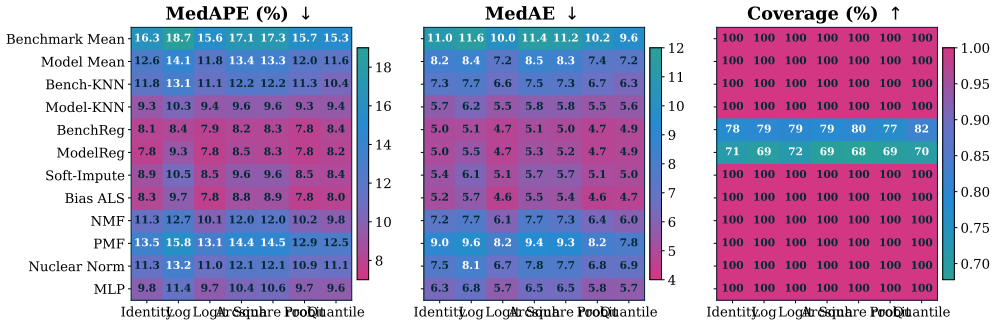


Figure 9: Full transform × method grid (7 transforms, 12 methods) across the metrics in Sec. 3: MedAPE, MedAE, and coverage. Each cell reports the hyperparameter selected on nested validation cells, evaluated on the outer test cells as the median over 10 seeds × 3 folds. All methods operate in standardized space. Green = better.

Table 11: Full transform × method grid: all 84 pairs of feature transform and prediction method from Sec. 3.1, sorted by MedAPE. Each row reports the hyperparameter selected on nested validation cells (Table 10); where that choice varies across outer folds, the one selected most often is shown. Errors are measured on the original score scale on the outer test cells as the median over 10 seeds × 3 folds.

#	Transform	Method	Hyperparameter	MedAPE (%) ↓	MedAE ↓	Cov.
1	Logit	ModelReg	$R^2_{\min}$=0.3, k=3	7.75	4.69	72%
2	Logit	Bias ALS	λ=0.1, r=2	7.77	4.63	100%
3	Probit	Bias ALS	λ=0.1, r=2	7.80	4.64	100%
4	Probit	ModelReg	$R^2_{\min}$=0.3, k=3	7.81	4.71	69%
5	Identity	ModelReg	$R^2_{\min}$=0.2, k=3	7.83	5.03	71%
6	Probit	BenchReg	$R^2_{\min}$=0.3, k=7	7.84	4.73	77%
7	Logit	BenchReg	$R^2_{\min}$=0.3, k=7	7.95	4.69	79%
8	Quantile	Bias ALS	λ=0.1, r=2	8.00	4.73	100%
9	Identity	BenchReg	$R^2_{\min}$=0.3, k=7	8.10	4.96	78%
10	Quantile	ModelReg	$R^2_{\min}$=0.3, k=3	8.17	4.90	70%
11	Arcsinh	BenchReg	$R^2_{\min}$=0.3, k=7	8.23	5.06	79%
12	Identity	Bias ALS	λ=0.1, r=2	8.25	5.17	100%
13	Square root	ModelReg	$R^2_{\min}$=0.2, k=3	8.27	5.17	68%
14	Square root	BenchReg	$R^2_{\min}$=0.3, k=5	8.29	4.99	80%
15	Quantile	BenchReg	$R^2_{\min}$=0.3, k=7	8.35	4.88	82%
16	Log	BenchReg	$R^2_{\min}$=0.3, k=7	8.37	5.08	79%
17	Quantile	Soft-Impute	r=2	8.42	5.04	100%
18	Arcsinh	ModelReg	$R^2_{\min}$=0.3, k=3	8.46	5.29	69%
19	Logit	Soft-Impute	r=2	8.49	5.10	100%
20	Probit	Soft-Impute	r=2	8.53	5.08	100%
21	Arcsinh	Bias ALS	λ=0.1, r=2	8.84	5.47	100%
22	Identity	Soft-Impute	r=2	8.89	5.40	100%
23	Square root	Bias ALS	λ=0.1, r=2	8.95	5.36	100%
24	Probit	Model-KNN	k=10	9.26	5.50	100%
25	Log	ModelReg	$R^2_{\min}$=0.3, k=3	9.28	5.54	69%
26	Identity	Model-KNN	k=10	9.33	5.69	100%
27	Logit	Model-KNN	k=10	9.40	5.51	100%
28	Quantile	Model-KNN	k=10	9.41	5.61	100%
29	Arcsinh	Soft-Impute	r=2	9.55	5.74	100%
30	Square root	Soft-Impute	r=2	9.57	5.74	100%
31	Quantile	MLP	lr=0.001	9.59	5.74	100%
32	Square root	Model-KNN	k=10	9.62	5.79	100%
33	Arcsinh	Model-KNN	k=10	9.63	5.83	100%
34	Logit	MLP	lr=0.001	9.65	5.73	100%
35	Log	Bias ALS	λ=0.1, r=2	9.68	5.72	100%
36	Probit	MLP	lr=0.001	9.70	5.78	100%
37	Quantile	NMF	r=1	9.76	5.95	100%
38	Identity	MLP	lr=0.001	9.84	6.30	100%
39	Logit	NMF	r=1	10.12	6.10	100%
40	Probit	NMF	r=1	10.21	6.36	100%
41	Log	Model-KNN	k=10	10.30	6.21	100%
42	Arcsinh	MLP	lr=0.01	10.42	6.52	100%
43	Quantile	Bench-KNN	k=10	10.44	6.29	100%
44	Log	Soft-Impute	r=2	10.53	6.10	100%
45	Square root	MLP	lr=0.001	10.57	6.47	100%
46	Probit	Nuclear	λ=5.0	10.95	6.82	100%
47	Logit	Nuclear	λ=5.0	10.97	6.71	100%
48	Quantile	Nuclear	λ=5.0	11.10	6.88	100%
49	Logit	Bench-KNN	k=10	11.15	6.65	100%
50	Identity	NMF	r=1	11.26	7.22	100%
51	Probit	Bench-KNN	k=10	11.27	6.69	100%
52	Identity	Nuclear	λ=5.0	11.34	7.48	100%
53	Log	MLP	lr=0.01	11.35	6.76	100%
54	Quantile	Model-Mean	—	11.62	7.21	100%
55	Identity	Bench-KNN	k=10	11.76	7.29	100%

continued on next page

Table 11 – continued from previous page

#	Transform	Method	Hyperparameter	MedAPE (%) ↓	MedAE ↓	Cov.
56	Logit	Model-Mean	—	11.83	7.22	100%
57	Probit	Model-Mean	—	11.95	7.38	100%
58	Square root	NMF	r=1	11.96	7.32	100%
59	Arcsinh	NMF	r=1	11.97	7.70	100%
60	Arcsinh	Nuclear	λ=5.0	12.05	7.79	100%
61	Square root	Nuclear	λ=5.0	12.14	7.67	100%
62	Square root	Bench-KNN	k=10	12.16	7.33	100%
63	Arcsinh	Bench-KNN	k=10	12.17	7.45	100%
64	Quantile	PMF	r=5	12.54	7.82	100%
65	Identity	Model-Mean	—	12.61	8.19	100%
66	Log	NMF	r=1	12.70	7.66	100%
67	Probit	PMF	r=5	12.91	8.19	100%
68	Logit	PMF	r=3	13.07	8.16	100%
69	Log	Bench-KNN	k=10	13.09	7.65	100%
70	Log	Nuclear	λ=5.0	13.20	8.10	100%
71	Square root	Model-Mean	—	13.33	8.29	100%
72	Arcsinh	Model-Mean	—	13.36	8.53	100%
73	Identity	PMF	r=5	13.54	9.04	100%
74	Log	Model-Mean	—	14.14	8.44	100%
75	Arcsinh	PMF	r=3	14.42	9.36	100%
76	Square root	PMF	r=3	14.53	9.26	100%
77	Quantile	Bench-Mean	—	15.26	9.65	100%
78	Logit	Bench-Mean	—	15.56	10.04	100%
79	Probit	Bench-Mean	—	15.70	10.21	100%
80	Log	PMF	r=3	15.85	9.61	100%
81	Identity	Bench-Mean	—	16.29	11.04	100%
82	Arcsinh	Bench-Mean	—	17.06	11.40	100%
83	Square root	Bench-Mean	—	17.28	11.18	100%
84	Log	Bench-Mean	—	18.72	11.56	100%

C.3 BENCHPRESS VS. LLM-BASED BENCHMARK SCORE PREDICTORS

BENCHPRESS predicts from the observed score matrix alone. Another natural question is whether a frontier LLM can predict a benchmark score directly from the target model, the target benchmark, and a few nearest-peer examples. This is a per-cell LLM predictor: each target score is queried separately, and the prompt may expose public model and benchmark identities.

Experiment setting. We use the same held-out cells as the method-comparison experiment in Sec. 3.1. For each target cell (model, benchmark), we select peer examples using only the training matrix for that fold. A candidate peer model must have an observed score on the target benchmark and share at least five visible benchmarks with the target model. Among eligible peers, we choose the five models with the highest Pearson correlation to the target model over shared visible scores. The prompt then asks GPT-5.5 to predict the target model's score on the target benchmark from these five peer examples. We consider two scenarios. In the *informed* condition, the prompt keeps the real model and benchmark names. In the *blind* condition, model and benchmark identifiers are anonymized within the prompt, while the scores and peer-example structure are preserved. The informed condition tests whether a frontier LLM can exploit public model and benchmark semantics; the blind condition tests whether the numerical peer structure alone is enough. We score both conditions on the same held-out cells as BENCHPRESS using MedAPE and MedAE. The exact prompt template is given below.

Prompt. The LLM diagnostic in Sec. 3.2 uses no separate system prompt: the API call passes `system_message=None`, and all task instructions are contained in the user message. The user prompt is generated once per batch of target cells. In the named condition, model and benchmark fields use the real names and the benchmark line also includes the benchmark scale. In the blind condition, those fields are replaced by local labels such as `Target model q0`, `Benchmark A`, and `Peer model q0-1`; scores and the five-shot peer-example structure are unchanged. Here `query_id` is the local identifier for a target cell within the batch, such as `q0`; it is used only to match the returned JSON value to the corresponding query.

Five-shot LLM user prompt template

```
You are estimating benchmark results before running expensive
evaluations.
Each query gives compact known scores for a target model and five
nearest peer-model examples.
Make a quick numerical estimate from the nearest peers; do not
explain or show calculations.
Return ONLY valid JSON mapping each query id to a numeric score, e.g.
{"q0": 72.5}.

Query {query id, e.g. q0}
Target model: {target model name or blind target label}
Target known scores: [{benchmark label: score}, ...]
Estimate the target model's score on: {target benchmark name and
scale, or blind benchmark label}
Nearest peer examples:
Example 1: model={peer model name or blind peer label};
shared_scores=[{benchmark label: score}, ...]; {target benchmark
label} score: {peer target score}
...
Example 5: model={peer model name or blind peer label};
shared_scores=[{benchmark label: score}, ...]; {target benchmark
label} score: {peer target score}
```

C.4 Comparison to Observational Scaling Laws

Observational Scaling Laws (OSL) (Ruan et al., 2024) construct low-dimensional capability measures from a fixed block of standard benchmarks and fit a separate predictor for each selected downstream capability, including under weak-to-strong extrapolation. BenchPress instead considers the general setting in which any available benchmark scores may be observed and any remaining score may be the prediction target.

This difference shows up in how much data each method needs. OSL's released matrices are nearly complete: benchmarks miss under 5% of their entries and every training benchmark carries at least ten model scores, so its per-benchmark regressions always see dense columns. Our matrix is 76.7% missing, and some benchmarks are observed for as few as two models, so a per-benchmark regression frequently has fewer training models than free parameters. OSL's largest completion task predicts two agentic targets for twenty-two models (their Section 4.2), whereas ours predicts 133 benchmarks for 84 models.

To run OSL on our matrix, we supply the dense capability block it assumes in two ways: (i) restrict to the eight most-observed benchmark columns and impute their remaining gaps before fitting OSL's capability regression; and (ii) impute the full 76.7%-missing matrix before the same fit. Both variants and BenchPress are evaluated with the same 30-fold protocol and the same evaluation harness, and Table 12 reports the median error over each method's finite predictions on the held-out cells. Even the denser variant (i) trails BenchPress, and imputing the full sparse matrix (ii) degrades OSL further.

Table 12: Applying OSL (Ruan et al., 2024) to our score matrix, against BenchPress. OSL needs a densely scored capability block, supplied either by (i) the eight most-observed benchmark columns or (ii) imputing the full matrix. Median error is computed by the shared evaluation harness on each method's finite predictions under the same 30-fold protocol; lower is better.

Method	MedAPE (%) ↓	MedAE ↓
OSL, densest-8 block (i)	9.85	5.94
OSL, full-matrix impute (ii)	21.87	12.17
BenchPress	**7.77**	**4.63**

One way to read the gap is that BenchPress draws on more of the scores a model already has. OSL summarizes each model with a few capability values fitted on one fixed benchmark block, so observed

scores outside that block do not directly enter its prediction, whereas BENCHPRESS conditions each estimate on all of that model's observed cells. On a sparse public matrix, whose coverage is uneven across models and benchmarks, this may be part of why BENCHPRESS does better here.

The gap does not make OSL a weaker method on its own terms. Where a genuinely stronger model must be predicted from weaker models alone, with no comparably strong peer already in the matrix, OSL's scaling formulation is the better-matched tool; that regime lies outside the general completion problem BENCHPRESS targets.

C.5 BENCHPRESS ON OTHER SCORE MATRICES

Third-party score matrices. To check that BENCHPRESS is not specific to our curated matrix, we rebuild two independent score matrices, one from the public HELM leaderboard (Liang et al., 2023) and one from Every Eval Ever (EEE) (Batzner et al., 2026), using the same construction rules and the same predictor. Each source is curated into a BENCHPRESS-compatible matrix, restricted to its percentage-scale benchmarks and filtered to the paper's observation thresholds, giving an 83×207 HELM matrix (9,776 cells) and a 653×498 EEE matrix (24,879 cells). Table 13 scores BENCHPRESS on both against per-benchmark-median and global-median baselines under the paper's per-model holdout: BENCHPRESS is the most accurate under both metrics on both matrices, matching the pattern on our own matrix.

Table 13: BENCHPRESS on two independently built score matrices, curated from the HELM leaderboard (83×207, 9,776 cells) and Every Eval Ever (653×498, 24,879 cells), against per-benchmark-median and global-median baselines under the paper's per-model holdout; lower is better. BENCHPRESS is the most accurate on both.

	HELM		EEE	
Predictor	MedAE ↓	MedAPE ↓	MedAE ↓	MedAPE ↓
BENCHPRESS	**2.87**	**10.67%**	**2.89**	**5.02%**
Per-benchmark median	4.12	15.31%	7.00	11.24%
Global median	19.31	46.60%	18.25	23.18%

Later score matrix (data cutoff August 26, 2026). We also rerun the full 329-configuration method comparison on the later score matrix of Sec. A.2 using the same filtering thresholds and 10-seed, three-fold per-model holdout. After filtering, it contains 129 models, 253 benchmarks, and 4,905 observed cells. Table 14 shows that the adopted Logit Bias ALS predictor remains fully covered and ranks sixth by MedAPE and first by MedAE. The lowest-MedAPE configuration is only slightly more accurate but covers 79% of held-out cells, so the larger matrix supports the same choice of Logit Bias ALS as the full-coverage predictor.

D SUPPLEMENTAL TO SEC. 4: WHAT BENCHPRESS ENABLES FOR MODEL EVALUATION

This section provides additional details for the model-evaluation analyses in Sec. 4. Sec. D.1 supplements Sec. 4.1 with its full experiment setting, a pairwise-redundancy diagnostic, the selected probe sets and their error curves, and more exhaustive probe-selection checks. Sec. D.2 gives the full experiment setting and results discussion of Sec. 4.2 and reports an auxiliary shortlist-recovery metric and ranking-aware probe selection. Sec. D.3 gives the full experiment setting, the per-target table, and a prospective check on a later score matrix for Sec. 4.3.

D.1 BUDGETED SCORECARD RECOVERY

The rank-2 structure identified in Sec. 2.2 suggests that benchmark scores contain substantial shared information, rather than 133 independent measurements. Direct evidence supports this: Sec. E.1.1 shows that most benchmarks can be predicted from the rest with low error, and Sec. D.1 shows

Table 14: Top-15 transform and method configurations on the later score matrix (data cutoff August 26, 2026) after applying the paper's observation thresholds (129×253, 4,905 observed cells). The two halves rank configurations independently by median absolute percentage error (MedAPE) and median absolute error (MedAE); each value is followed by prediction coverage in parentheses. All results report the median over 10 seeds and three folds. Pink highlights mark the Logit Bias ALS configuration adopted as BENCHPRESS's main score predictor.

MedAPE (%) ↓					**MedAE ↓**				
#	Transform	Method	Hyperparameter	Value	#	Transform	Method	Hyperparameter	Value
1	Probit	BenchReg	$R^2_{\min}$=0.3, k=7	7.6 (79%)	1	**Logit**	**Bias ALS**	λ=0.1, r=2	**4.43 (100%)**
2	Logit	BenchReg	$R^2_{\min}$=0.3, k=7	7.6 (79%)	2	Probit	Bias ALS	λ=0.1, r=2	4.47 (100%)
3	Probit	BenchReg	$R^2_{\min}$=0.2, k=7	7.7 (80%)	3	Logit	BenchReg	$R^2_{\min}$=0.3, k=7	4.49 (79%)
4	Logit	BenchReg	$R^2_{\min}$=0.2, k=7	7.7 (80%)	4	Logit	Bias ALS	λ=0.01, r=2	4.51 (100%)
5	Probit	BenchReg	$R^2_{\min}$=0.1, k=7	7.7 (81%)	5	Logit	BenchReg	$R^2_{\min}$=0.2, k=7	4.54 (80%)
6	**Logit**	**Bias ALS**	λ=0.1, r=2	**7.7 (100%)**	6	Probit	BenchReg	$R^2_{\min}$=0.3, k=7	4.55 (79%)
7	Probit	Bias ALS	λ=0.1, r=2	7.7 (100%)	7	Logit	BenchReg	$R^2_{\min}$=0.1, k=7	4.56 (80%)
8	Logit	BenchReg	$R^2_{\min}$=0.3, k=5	7.8 (72%)	8	Probit	BenchReg	$R^2_{\min}$=0.2, k=7	4.56 (80%)
9	Quantile	Bias ALS	λ=0.1, r=2	7.8 (100%)	9	Probit	BenchReg	$R^2_{\min}$=0.1, k=7	4.56 (81%)
10	Logit	BenchReg	$R^2_{\min}$=0.1, k=7	7.8 (80%)	10	Probit	Bias ALS	λ=0.01, r=2	4.57 (100%)
11	Logit	BenchReg	$R^2_{\min}$=0.1, k=5	7.8 (74%)	11	Quantile	Bias ALS	λ=0.1, r=2	4.61 (100%)
12	Logit	Bias ALS	λ=0.01, r=2	7.8 (100%)	12	Probit	Bias ALS	λ=1.0, r=2	4.62 (100%)
13	Probit	Bias ALS	λ=0.01, r=2	7.8 (100%)	13	Quantile	Bias ALS	λ=0.01, r=2	4.64 (100%)
14	Logit	BenchReg	$R^2_{\min}$=0.2, k=5	7.8 (73%)	14	Logit	Bias ALS	λ=1.0, r=2	4.65 (100%)
15	Identity	BenchReg	$R^2_{\min}$=0.2, k=7	7.8 (79%)	15	Logit	BenchReg	$R^2_{\min}$=0.3, k=5	4.66 (72%)

that almost every benchmark column has at least one strongly correlated peer. Many scores can therefore plausibly be inferred from a small amount of carefully chosen evidence. We therefore ask the operational version of the same question: if a practitioner can run only a few benchmarks on a new model, which scores should be measured, and which scores can BENCHPRESS infer from them?

Experiment setting. We simulate a practitioner who evaluates each target model on a fixed probe set and then asks BENCHPRESS to complete the rest of that model's public scorecard. For a target model, only the probe columns remain visible in its row; all other models keep their observed rows. The selection objective evaluates every observed model-benchmark cell at every budget, with revealed probe cells counted as exact predictions, so the denominator is fixed across probe-set sizes. Except for the selection objectives in Table 17, all reported errors exclude the revealed probe cells: they are computed only on the remaining observed cells of each target model, which BENCHPRESS predicts from the masked matrix. Thus the curves measure how much of the current score matrix can be reconstructed from a small number of selected probes; the held-out model validation below evaluates selected probes on models not used for selection.

Starting from an empty probe set, we build a ten-benchmark set greedily. At each step, we try every remaining candidate benchmark, temporarily add it to the current probe set, evaluate pooled error on the fixed universe, and keep the candidate with the lowest error. We compare two greedy probe-set methods against a random baseline:

- **Cost-unaware greedy:** any benchmark can be selected as the next probe.
- **Cost-aware greedy:** candidates are restricted by the low-cost allowlist.
- **Random baseline:** we run 10 seeds. Each seed draws one global random benchmark ordering, and each budget uses the corresponding prefix for every target model. The plotted line is the mean across seeds; the shaded band is the 25th–75th percentile range.

Results. The greedy procedure yields four ten-probe sets (Table 16), one for each combination of *objective* (MedAPE or MedAE) and *candidate pool* (any benchmark, or the low-cost allowlist). All four sets lean heavily on reasoning- and math-oriented benchmarks (GPQA Diamond, ARC-AGI, MATH-500, multiple AIME and HMMT contests, and similar): reasoning is a dominant axis of variation in the score matrix, so these benchmarks supply the cleanest signal for BENCHPRESS to triangulate the rest of a model's profile. Figure 10 (MedAPE) and Figure 1b (MedAE) trace the

corresponding error curves as the probe budget grows from one to ten. Two qualitative trends emerge. First, cost-aware greedy tracks the cost-unaware curve closely at every budget; restricting probes to the low-cost allowlist costs surprisingly little, because the cleanest reasoning-axis probes are already low-cost. Second, both greedy curves clearly separate from the random baseline at every budget (at five probes, 4.74 versus 7.50 MedAE): *which* benchmarks are chosen matters far more than how many. A more exhaustive probe-selection analysis below shows that greedy probe selection is cost-effective: its performance matches or differs only marginally from more exhaustive search.

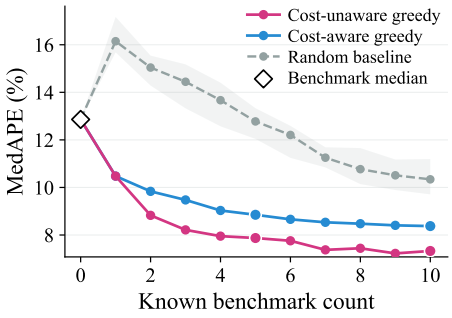


Figure 10: **MedAPE during probe-set construction.** Pooled MedAPE on the unrevealed cells decreases as selected benchmark scores are revealed.

Held-out model validation. We split the 84 models at random (seed 42) into 59 selection models and 25 held-out models with 721 observed cells, and rerun greedy selection with the MedAE objective on the selection rows only. Each held-out model is then treated as a new model: BENCHPRESS sees the selection rows and that model's probe scores, but no other held-out row, and predicts the model's remaining observed cells. The random baseline uses the same split and 10 seeds. On the held-out models, cost-aware greedy reaches a MedAE of 6.59, 5.23, and 5.54 at k=1, 5, and 10, against 10.37, 7.22, and 5.91 for the random baseline (median across seeds), and it is lower than the random baseline at every budget from one to ten. Cost-unaware greedy, rerun on the same selection models, reaches 6.59, 5.32, and 4.26 at k=1, 5, and 10, and is also lower than the random baseline at every budget.

Probe choice and the rank-2 term. We rerun the fixed probe orderings with two changes to separate the choice of probes from the rank-2 term. First, a most-reported-first ordering ranks benchmarks by the number of models that report them, over all benchmarks or over the low-cost allowlist, and keeps BENCHPRESS unchanged. Second, an offsets-only variant sets the rank of BENCHPRESS to zero on the greedy orderings, so each target is predicted from model and benchmark offsets fit to the same revealed scores. At five probes, cost-unaware greedy reaches a MedAE of 4.74 (4.35 at ten), against 5.73 (5.17) for most-reported-first and 5.41 (5.18) for offsets only on the same probes. Cost-aware greedy reaches 5.32 (4.92), against 5.50 (4.99) for most-reported-first within the allowlist and 5.81 (5.73) for offsets only. With a single probe, offsets only is slightly more accurate (5.92 versus 6.27), as in the temporal setting of Sec. D.3. A coverage-matched random baseline separates the choice of probes from the number of cells they reveal: 10 random five-benchmark sets per pool whose revealed cells are within 10% of the greedy set's (250 cells for cost-unaware greedy, 219 for cost-aware greedy) reach a median MedAE of 5.71 (interquartile range 5.60 to 5.94) over all benchmarks and 5.58 (5.47 to 5.76) within the allowlist, against 4.74 and 5.32 for greedy; at ten probes the gap is 5.18 versus 4.35 and 5.25 versus 4.92.

Beyond these results, this appendix supplements Sec. 4.1 in three ways. First, a pairwise-redundancy diagnostic explains why a small probe set can recover most of the matrix: the typical benchmark already has another benchmark column that predicts it nearly perfectly. Second, we list the selected probe sets and note how to read them. Third, a more exhaustive probe-selection analysis asks whether greedy's computational simplicity comes at a material accuracy cost, comparing it with exact enumeration over the low-cost allowlist and exact search after pruning in the unrestricted setting.

Widespread redundancy across benchmarks. Before choosing a probe set, we first ask whether many benchmark columns are redundant enough that a small probe set could plausibly recover the rest. For every ordered pair of benchmarks (b, b'), we collect the scores s_{mb} and $s_{mb'}$ of all models m evaluated on both ($n \geq 5$ shared models required), apply a logit transform followed by per-column z-scoring, fit a univariate linear regression in this transformed space, and invert the transform to obtain predicted raw scores $\hat{s}_{mb}$. For each target benchmark b, we identify its *best predictor benchmark*, the predictor b' that maximizes the absolute Pearson correlation, and report the signed correlation, MedAE, and MedAPE. Of the 133 benchmarks, 132 have at least one neighbor pair with ≥ 5 shared models; 1 is excluded for insufficient overlap. Among these 132, 127 have a best-neighbor absolute correlation ≥ 0.85 (129 reach ≥ 0.80), and the median best-neighbor absolute correlation is 0.97. Table 15 lists the five most and five least predictable benchmarks. The most predictable pair, the LongFact Concepts and LongFact Objects hallucination-rate benchmarks, achieves a correlation of 0.997. At the other extreme, Safety (OLMES suite) has the weakest best-neighbor correlation (0.62), followed by MRCR v1 (0.68).

Table 15: Five most and least predictable benchmarks identified by pairwise linear regression in logit + z-score space. Rows are selected by absolute Pearson correlation between the target and its best predictor benchmark; the Corr. column reports the signed value.

Bench.	Best neighbor	Corr.	MedAPE ↓	MedAE ↓
Highly predictable (highest correlation)				
LongFact-Concepts	LongFact-Objects	.997	5.4%	0.37
LongFact-Objects	LongFact-Concepts	.997	7.9%	0.33
Beyond AIME	BRUMO 2025	.994	1.9%	1.56
BRUMO 2025	Beyond AIME	.994	0.3%	0.29
MathVista	MMMU	.992	3.9%	2.64
Hard to predict (lowest correlation)				
Safety (OLMES suite)	HumanEval	.622	8.1%	5.81
MRCR v1	SimpleQA	.676	6.7%	4.86
HealthBench	GraphWalks parents 0–128K	-.789	4.9%	2.75
OmniDocBench 1.5	GDPval (Artificial Analysis ELO)	.815	131.9%	2.42
ComplexFuncBench	tau-bench Airline	.846	27.1%	11.41

Caveat: high cross-category correlation does not imply semantic similarity. Some cross-category pairs appear surprisingly predictable; for example, GDPval (Artificial Analysis ELO) and WideSearch have correlation 0.99. This does not mean GDP-style task performance predicts search-agent performance. The regression is fit on only 5 models that have scores on both benchmarks, all of which are frontier models whose scores are dominated by a single general-capability axis: whichever model is strongest overall tends to score highest on both. With so few data points and so little capability diversity, nearly any two benchmarks will correlate. These inflated cross-category correlation values reflect sample composition, not a meaningful relationship between the benchmarks.

Finding 9: Most benchmark scores are inferable from one well-chosen peer, reflecting shared variation across the matrix.

Selected probe sets. Table 16 lists the ten-probe sets selected in Sec. 4.1 for each objective and candidate pool. The low-cost candidate pool is a fixed list of 25 benchmarks considered inexpensive to run and specified before selection: Vibe-Eval; τ^2-bench Telecom; Bullshit-Bench (Clear Pushback); AIME 2026; AlpacaEval 2.0 (LC-winrate); AIME 2025; τ^2-bench Airline; BRUMO 2025; HMMT Nov 2025; IFBench; MathArena Apex 2025; CMIMC 2025; HMMT Feb 2026; SMT 2025; AA Long Context Reasoning; GPQA Diamond; τ^2-bench Retail; Aider Polyglot (whole mode); CritPt; Arena-Hard Auto; SciCode; MATH-500; HMMT Feb 2025; Aider Polyglot (diff mode); and MMLU-Pro.

Remark 1 (Near-duplicate probes from the same benchmark family). Greedy occasionally selects two versions of the same benchmark family (e.g., two AIME contests in the low-cost MedAE run, ARC-AGI-1 and ARC-AGI-2 in the unrestricted MedAPE run). This is not a sign that the second version adds a new axis; it is noise averaging on an axis with no good substitute. In the low-cost setting, once the highest-signal reasoning probes (GPQA Diamond, MMLU-Pro, Aider Polyglot, MATH-500) are exhausted, the remaining low-cost candidates are mostly near-duplicate math contests. In the

Table 16: **Top-10 probe sets selected by each objective.** Each row uses the same set of observed model–benchmark cells and the same greedy procedure; only the objective and candidate pool change.

Objective	Candidate pool	Top-10 prefix
MedAPE	Any benchmark	GPQA Diamond; HLE; AIME 2024; MMLU-Pro; ARC-AGI-1; ARC-AGI-2; Aider Polyglot (diff mode); LiveCodeBench; Terminal-Bench 2.0; SWE-bench Verified
	Low-cost benchmarks	GPQA Diamond; MMLU-Pro; Aider Polyglot (diff mode); MathArena Apex 2025; HMMT Nov 2025; Bullshit-Bench (Clear Pushback); MATH-500; AIME 2025; Arena-Hard Auto; IFBench
MedAE	Any benchmark	GPQA Diamond; HLE; Codeforces Rating; MMLU-Pro; ARC-AGI-1; Terminal-Bench 2.0; MATH-500; HLE Text; AIME 2025; GDPval (Artificial Analysis ELO)
	Low-cost benchmarks	GPQA Diamond; MMLU-Pro; Aider Polyglot (diff mode); MATH-500; AIME 2026; AIME 2025; Bullshit-Bench (Clear Pushback); Arena-Hard Auto; τ^2-bench Airline; IFBench

unrestricted MedAPE setting, ARC-AGI captures a frontier-reasoning axis with very low absolute scores, where small errors translate into large percentage errors, so a second ARC-AGI instance is the highest-marginal way to reduce error on that axis.

Remark 2 (Comparability with Sec. 3.1). The setting differs from the one-third per-model holdout: only the target model's non-probe cells are masked, so BENCHPRESS sees richer context and the endpoints are not directly comparable to Sec. 3.1.

Remark 3 (Snapshot-specific probe sets). For a different matrix or snapshot, this construction should be rerun: the current probe set is a snapshot-specific recommendation, not a permanent benchmark list.

Pruned exhaustive probe selection. The budgeted scorecard recovery experiment in Sec. 4.1 selects probe sets greedily. Greedy selection is simple and fast, but it does not certify that the selected five probes are the best possible subset. We therefore run two exhaustive checks, one where exact enumeration is feasible and one where the unrestricted search space first has to be pruned.

- *Cost-aware exact exhaustive search.* The low-cost allowlist has only 25 candidate probes, so exact enumeration is feasible without pruning. We search all $\binom{25}{5} = 53{,}130$ five-probe subsets to test whether the cost-aware greedy prefix misses a better cheap combination.
- *Cost-unaware pruned exhaustive search.* The unrestricted setting has 133 candidate probes, making exact search over all $\binom{133}{5}$ five-probe subsets too large. We therefore build a top-30 candidate pool from the full ten-step MedAE greedy trajectory: at each step, every remaining candidate is ranked by the pooled MedAE it would achieve if added next, and ranks are averaged across steps. This pruning keeps the full MedAE greedy top-10 prefix, reduces the exact search to $\binom{30}{5} = 142{,}506$ subsets, and lets us test whether the greedy five-probe solution is preserved when the final subset is selected exactly.

The results are reported in Table 17. In the unrestricted setting, pruned exhaustive search returns the same five probes as greedy. In the low-cost setting, exact exhaustive search improves slightly over greedy, but the gain is small; the greedy probe sets are already close to optimal under both policies.

Table 17: **Five-probe MedAE selections.** Cost-unaware greedy already matches the best five-probe set found by exact search over the pruned top-30 universe. In the low-cost allowlist, exhaustive search slightly improves over cost-aware greedy. Errors are the selection objective, which counts revealed probe cells as exact predictions.

Policy	Selection method	Five-probe set	MedAE↓	MedAPE↓
Cost-unaware	Greedy	GPQA Diamond; HLE; Codeforces Rating; MMLU-Pro; ARC-AGI-1	3.926	6.588
	Pruned exhaustive	GPQA Diamond; HLE; Codeforces Rating; MMLU-Pro; ARC-AGI-1	3.926	6.588
Cost-aware	Greedy	GPQA Diamond; MMLU-Pro; Aider Polyglot (diff mode); MATH-500; AIME 2026	4.549	7.596
	Exact exhaustive	τ^2-bench Telecom; MathArena Apex 2025; GPQA Diamond; Aider Polyglot (diff mode); MMLU-Pro	4.465	7.412

D.2 PRESERVING MODEL RANKINGS

The practical value of score prediction is not only numerical accuracy, but whether the predictions support the same evaluation decisions. Here the operational question is simple: when two models differ meaningfully on the same benchmark, does BENCHPRESS preserve which model is better?

Experiment setting. We reuse the holdout setting of Sec. 3.1 (10 seeds, three folds per model) and complete each benchmark leaderboard with true scores on seen cells and BENCHPRESS predictions on held-out cells. Because adjacent leaderboard slots are often separated by tiny gaps that a small prediction error can flip, we evaluate margin-aware pairwise ordering rather than exact ranks. An auxiliary evaluation of whether the predicted leaderboard recovers the true top-ranked models, together with ranking-aware probe selection, is reported below. For each benchmark, we form all same-benchmark model pairs with at least one held-out cell, keep those whose true score gap is at least the row's margin, and report the median across benchmarks of

$$\text{pairwise accuracy} = \frac{\#\{\text{comparable pairs whose completed order matches the true order}\}}{\#\{\text{comparable pairs}\}}.$$

The margin-0 row includes every non-tied pair and is most sensitive to near-ties; larger margins focus on clearer model differences.

Results. The margin-aware results in Table 4 show that score-prediction errors rarely overturn meaningful ordering decisions. The margin-0 row is lower because it includes many near-tied model pairs where either ordering is fragile. At a two-point score margin, BENCHPRESS achieves 88.0% pairwise ranking accuracy across 531,498 comparable pairs. When the true score gap is at least five points, pairwise ranking accuracy rises to 92.1%. On the same pairs, the logit-space model mean reaches 78.3%, 80.2%, 82.1%, and 86.7% at margins 0, 1, 2, and 5, and the logit-space benchmark mean 59.2%, 60.0%, 61.2%, and 64.4%. The numbers of comparable pairs at margins 0, 1, 2, and 5 are 589,830, 561,283, 531,498, and 454,090. Restricting each benchmark leaderboard to the models held out in the same fold, so that every scored pair compares two predicted scores, leaves 117,990, 112,117, 106,202, and 90,810 pairs; on these pairs BENCHPRESS reaches 81.3%, 82.8%, 84.4%, and 88.1%, and the logit-space model mean 76.7%, 78.0%, 79.2%, and 83.3%. We additionally plot pairwise ranking accuracy as a probe budget grows from one to ten benchmarks (Figure 11); both informativeness-greedy and cost-aware probe sets stay well above the random baseline, and the underlying greedy probe-set selection is detailed below.

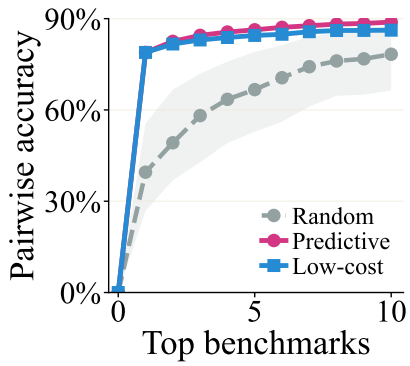


Figure 11: **Overall ranking preservation.** Pairwise ranking accuracy as the probe budget grows.

Auxiliary metric: shortlist recovery. Sec. 4.2 reports same-benchmark pairwise ranking accuracy as the main ranking metric. As an auxiliary view, we also measure shortlist recovery. For each benchmark and held-out fold, we complete the full observed leaderboard by keeping seen scores fixed and replacing held-out cells with BENCHPRESS predictions. We then compare the completed top fraction with the true top fraction on that same observed leaderboard. Because the predicted and true shortlists have the same size, the overlap rate is the fraction of true top models recovered

Table 18: **Auxiliary shortlist recovery.** Recovery is median benchmark-level overlap; slots count top-fraction positions.

Setting	Recovery	Slots
Top 10%	72.4%	9,515
Top 20%	79.3%	17,154
Top 30%	83.9%	25,091

by the predicted shortlist. Table 18 reports two summaries of this overlap: *recovery* computes top-fraction recovery separately for each benchmark and then reports the median across benchmarks, while *shortlist slots* counts selection positions rather than unique models, so a benchmark-fold group contributing four top-20% positions contributes four slots.

Probe selection for ranking preservation. We also run a probe-selection diagnostic that optimizes the ranking metric directly. This greedy search evaluates the same set of observed model–benchmark cells as Sec. 4.1, but scores each candidate prefix by same-benchmark pairwise ranking accuracy with a five-point score margin. Probe cells are revealed exactly and remain in the denominator, so the question is which known benchmark scores most improve the completed leaderboard. Table 19 reports two top-10 prefixes selected by this ranking-aware objective: a cost-unaware search that may choose any benchmark, and a cost-aware search restricted by the low-cost allowlist. The cost-aware constraint costs only a small amount of ranking accuracy at ten probes (86.2% versus 88.9%) while selecting a more practical benchmark set.

Step	Cost-unaware probe	Accuracy	Cost-aware probe	Accuracy	Comparable pairs
1	GPQA Diamond	78.9%	GPQA Diamond	78.9%	27,245
2	BrowseComp	82.5%	MMLU-Pro	81.7%	27,245
3	Codeforces Rating	84.5%	AIME 2025	83.0%	27,245
4	DROP	85.6%	Bullshit-Bench (Clear Pushback)	83.8%	27,245
5	ScreenSpot-Pro	86.3%	HMMT Feb 2026	84.5%	27,245
6	GraphWalks parents 0–128K	87.1%	MATH-500	84.9%	27,245
7	AIME 2024	87.7%	AlpacaEval 2.0 (LC-winrate)	85.7%	27,245
8	MGSM	88.2%	HMMT Feb 2025	86.1%	27,245
9	NL2Repo-Bench	88.4%	Aider Polyglot (whole mode)	86.1%	27,245
10	AlpacaEval 2.0 (LC-winrate)	88.9%	Arena-Hard Auto	86.2%	27,245

Table 19: **Top-10 probe sets selected for ranking preservation.** Both greedy searches optimize same-benchmark pairwise ranking accuracy at a five-point score margin. The cost-unaware search may choose any benchmark; the cost-aware search is restricted by the low-cost allowlist. At each step, the selected probe prefix is evaluated on the same fixed universe of observed cells; revealed probe cells are exact and unrevealed observed cells are predicted by BENCHPRESS.

D.3 PREDICTING NEWLY RELEASED MODELS

Experiment setting. We evaluate an intermediate segment of the release timeline, chosen using only release metadata and matrix coverage before inspecting prediction errors. This choice avoids two uninformative extremes: very early targets leave too few older models in the training matrix, while the latest releases would be predicted from almost the full snapshot rather than a meaningfully historical matrix. Concretely, we use models from the post-DeepSeek-R1 reasoning era through GPT-5.1, and keep only models with more than 20 observed benchmark scores. The coverage threshold ensures that, after revealing up to ten seed scores, each target still has enough hidden cells for a meaningful per-model error estimate. This yields 27 target models across the recent reasoning-era release window. For each target model, we train BENCHPRESS on only the models released before the target's release date, so the predictor sees no information about the new release beyond what we explicitly reveal. The cutoff applies to model rows: older models keep all their scores, including scores reported after the target's release, because the matrix does not record when each score was reported. We then reveal $k \in \{1, 5, 10\}$ of that target model's observed benchmark scores and predict the rest, repeating each setting over 10 random seeds and reporting the median. Revealed cells are excluded from the metric,

hidden cells with finite predictions enter MedAPE and MedAE, and hidden cells without a finite deployment prediction are dropped.

Per-target results. The main text summarizes the temporal deployment stress test as a distribution across target models. Table 20 lists the per-target results.

Table 20: **Full temporal deployment results.** For each target model, Obs. is the number of observed benchmark scores in the final matrix and Train is the number of older models available before the target's release date. Each k column reveals that many seed scores from the target model and predicts the rest; revealed seed scores are excluded from both metrics, and numbers are medians across 10 random seeds. The last row scores the logit-space model mean, fit to the same training matrix and seed scores, on the same hidden cells.

			$k = 1$		$k = 5$		$k = 10$	
Target model	Obs.	Train	MedAPE	MedAE	MedAPE	MedAE	MedAPE	MedAE
DeepSeek-R1	40	23	23.7	13.3	20.8	8.6	14.8	8.8
o3-mini (high)	34	25	20.3	9.2	11.7	8.0	11.2	7.9
Grok 3 Beta	21	27	12.5	9.4	12.3	8.5	16.6	8.9
Claude 3.7 Sonnet	25	28	26.5	14.6	23.1	13.2	18.9	10.3
GPT-4.5	28	29	14.2	6.3	11.5	6.8	12.8	6.1
QwQ-32B	23	30	14.0	10.2	10.1	8.5	14.7	11.1
Llama 4 Maverick	22	33	10.6	6.9	10.0	6.6	7.3	5.4
GPT-4.1	48	34	15.4	7.5	8.2	4.1	10.3	4.3
GPT-4.1 mini	43	34	15.7	6.7	11.8	5.1	15.1	6.1
GPT-4.1 nano	35	34	38.5	9.6	29.6	6.0	34.4	8.7
o3 (high)	44	37	22.6	10.4	14.3	7.5	11.9	7.0
o4-mini (high)	45	37	23.1	11.1	14.7	7.3	14.9	6.8
Qwen3-235B-A22B	34	39	10.3	7.0	11.4	7.0	10.2	7.3
Gemini 2.5 Flash	43	43	28.0	11.7	9.3	6.6	10.0	5.8
Claude Opus 4	23	44	28.0	11.9	18.5	6.6	17.7	5.7
Claude Sonnet 4	39	44	26.8	15.6	18.3	7.6	22.3	7.5
DeepSeek-R1-0528	24	46	22.0	14.0	10.3	6.8	9.4	6.0
Gemini 2.5 Pro (GA)	65	47	13.4	6.8	13.9	5.8	9.9	5.5
Grok 4	24	48	17.1	9.9	11.9	8.1	14.4	10.1
GPT-5 mini	30	49	11.5	6.6	12.5	6.2	8.7	5.0
GPT-5 nano	30	49	14.0	4.6	12.3	4.1	10.8	4.6
Kimi K2	45	51	15.6	7.8	16.8	7.0	14.9	6.1
GPT-5	67	52	23.5	14.3	10.1	5.4	10.3	5.6
GLM-4.6	27	55	14.4	6.7	12.7	6.0	15.0	7.1
Claude Sonnet 4.5	79	56	21.0	9.3	15.3	7.8	15.1	7.0
MiniMax-M2	28	57	10.1	5.3	9.7	5.2	8.0	4.0
GPT-5.1	37	59	15.3	10.3	10.3	6.5	9.3	5.4
Median			15.7	9.4	12.3	6.8	12.8	6.1
Median, logit-space model mean			14.9	8.6	14.0	7.5	13.5	7.7

Prospective check on a later score matrix. We also test BENCHPRESS prospectively: we fit it on the main score matrix (data cutoff April 30, 2026) and score it against the later score matrix of Sec. A.2 (data cutoff August 26, 2026), restricted to the benchmarks of the main matrix. The 2,604 cells observed in both matrices keep the same value in 99.7% of cases. First, the later matrix reports 178 scores of main-matrix models on main-matrix benchmarks that are missing from the main matrix; BENCHPRESS predicts them with a MedAE of 4.44 points (MedAPE 6.39%), against 4.86 for the logit-space model mean and 9.55 for the logit-space benchmark mean. These errors lean positive: BENCHPRESS over-predicts 65% of the 178 cells, with a median signed error of +2.48 points (58% and +1.26 for the logit-space model mean). Second, for each of the 45 models added in the later matrix, we append its row to the main matrix, reveal its scores on the first k benchmarks of a MedAE ordering from Table 16 or of a random ordering, and predict its remaining scores on main-matrix

benchmarks. A model enters a budget if it reports at least one revealed benchmark and at least five other main-matrix benchmarks. Table 21 reports pooled errors on the unrevealed cells, together with the errors of the logit-space model mean and of main-matrix benchmark medians on the same cells. On these new models the logit-space model mean is as accurate as BENCHPRESS: in each greedy row, BENCHPRESS has the lower MedAE on 42% to 59% of targets, and a target bootstrap 95% interval for the difference in median per-target MedAE contains zero.

Table 21: **Prospective prediction of models added in the later score matrix (data cutoff August 26, 2026).** BENCHPRESS is fit on the main score matrix (data cutoff April 30, 2026) plus the revealed probe scores of one new model at a time. MedAE is pooled over unrevealed cells; the last two columns score the logit-space model mean, given the same revealed scores, and main-matrix benchmark medians on the same cells. Random rows pool the unrevealed cells of 10 random orderings, whose target sets differ.

Probe ordering	k	Targets	Cells	MedAE	Model mean	Benchmark median
Cost-unaware greedy	1	29	483	8.41	6.58	10.20
	3	33	497	6.81	6.19	10.30
	5	40	541	6.88	6.95	10.15
	10	41	502	6.38	6.13	9.30
Cost-aware greedy	1	29	483	8.41	6.58	10.20
	3	36	529	8.18	7.12	10.10
	5	37	517	8.08	7.70	10.00
	10	37	485	7.00	7.64	9.61
Random	1		1,227	7.87		10.50
	3		2,522	8.70		10.80
	5		2,935	8.50		10.80
	10		4,033	7.29		10.20

E SUPPLEMENTAL TO SEC. 5: WHEN TO TRUST BENCHPRESS'S PREDICTIONS

This section provides additional details for the trust analyses in Sec. 5. Sec. E.1 supplements Sec. 5.1 with expanded prediction-error diagnostics. Sec. E.2 spells out the reliability estimators used in Sec. 5.2.

E.1 WHAT AFFECTS PREDICTION RELIABILITY

This subsection provides extended experimental analysis that complements the prediction-error analysis in Sec. 5.1. Sec. E.1.1 extends the benchmark analysis of Sec. 5.1 with the hypothesis definitions, their test results, and the full benchmark-side hypothesis grid. Sec. E.1.2 extends the model analysis of Sec. 5.1 with the hypothesis definitions, their test results, and the full model-side hypothesis grid.

Methodology. In what follows, we propose a set of hypotheses about why BENCHPRESS mispredicts certain cells: seven targeting benchmark-side factors and nine targeting model-side factors. Each hypothesis is phrased as a no-effect claim: a candidate factor is not associated with BENCHPRESS's prediction quality, measured by MedAPE and MedAE.

We assess each one with a standard statistical hypothesis test that returns a p-value: assuming the no-effect hypothesis were true, this is the probability that random chance alone would produce data deviating from "no effect" by at least as much as ours. A small p means such data would be extremely unlikely under the hypothesis, so the observation is inconsistent with the hypothesis and we reject it; a large p means our data is well within what random chance could produce, so we have no grounds to reject.

Throughout this section we treat $p < 0.01$ as our rejection threshold: when $p < 0.01$ we reject the hypothesis, i.e., the data provide strong evidence that the factor *does* matter; otherwise we fail to

reject (which may either mean the factor truly has no effect, or that we lack the sample size to detect one). Depending on the shape of the hypothesis, we use one of two tests:

- *Spearman rank correlation test.* We use this for observational hypotheses, where each benchmark or model contributes a measured feature and its BENCHPRESS prediction error. Spearman tests whether higher feature values are monotonically associated with higher or lower error, while being less sensitive to outliers than raw-value correlation.
- *Paired Wilcoxon signed-rank test.* We use this for intervention-style hypotheses, where the same benchmark or model is evaluated under a baseline setting and an ablated setting. The paired design controls for inherent target difficulty, and the rank-based test is more reliable than a paired t-test for our heavy-tailed error shifts.

Spearman rank correlation tests. Sec. 5.1 uses two hypothesis-test families. For observational hypotheses, each target contributes a single pair (x_i, y_i): a feature x_i that we measure but cannot intervene on (e.g. a benchmark's inherent rank-2 reconstruction quality, or a model's median observed score), and its BENCHPRESS prediction error y_i. We use the Spearman rank correlation test to ask whether targets with a higher feature value tend to have higher or lower prediction error. We measure this association via the *rank* of each value, its position in the sorted ordering of its column, where the smallest value has rank 1 and the largest has rank n (this sense of "rank" is unrelated to the matrix-rank quantity used in Sec. 2.2). Using ranks rather than raw values makes the test sensitive to monotonic relationships, not only linear ones, and more robust to heavy-tailed errors and outliers. Concretely, we replace each column by its within-column ranks and compute the Pearson correlation between the two ranked columns; the result $\rho \in [-1, 1]$ is the Spearman correlation. Intuitively, $\rho > 0$ means targets with a higher feature value tend to also have higher prediction error, $\rho < 0$ means the opposite, and $|\rho|$ measures how consistently the ranking holds. The p-value asks: if there were truly no monotonic association ($\rho_{\text{true}} = 0$), how often would we see a sample correlation at least as extreme as ρ? It is computed from the t-approximation $t = \rho\sqrt{(n-2)/(1-\rho^2)}$, which under H_0 approximately follows a Student-t distribution with $n-2$ degrees of freedom (Hollander et al., 2014). The approximation is reliable when n is at least a few dozen (all our targets satisfy $n \geq 25$). Since we test for any deviation from $\rho = 0$, the p-value doubles the upper-tail probability, $p = 2\big(1 - F_{t_{n-2}}(|t|)\big)$, where $F_{t_{n-2}}$ is the CDF of the Student-t_{n-2} distribution. The intuition is: $|t|$ increases with both $|\rho|$ and the sample size n, so the p-value gets smaller only when the correlation is meaningful and backed by enough targets.

Paired Wilcoxon signed-rank tests. For intervention-style hypotheses, each target contributes a pair of errors $(y_i^{\text{baseline}}, y_i^{\text{intervention}})$, measured on the same target under two different settings (e.g. BENCHPRESS trained on the original matrix vs. on a matrix where every benchmark highly correlated with the target has been masked out). We use the paired Wilcoxon signed-rank test to ask whether the intervention shifts each target's error in a consistent direction. Because the comparison is within-target, each target serves as its own control, removing the effect of inherent target difficulty. We form per-target differences $\Delta_i = y_i^{\text{intervention}} - y_i^{\text{baseline}}$ and ask whether their median is zero (i.e. the intervention has no typical effect on prediction error). The Wilcoxon signed-rank test ranks $|\Delta_i|$ from smallest to largest, denotes the rank of pair i by R_i, and uses as its statistic the sum of ranks for positive differences, $W^+ = \sum_{i:\,\Delta_i > 0} R_i$ (in the rare case of any $\Delta_i = 0$, that pair is dropped before ranking). Under H_0 (the distribution of Δ is symmetric about 0), W^+ has mean $n(n+1)/4$ and variance $n(n+1)(2n+1)/24$, and the standardized statistic $z = (W^+ - \mathbb{E}[W^+])/\sqrt{\mathrm{Var}(W^+)}$ is approximately standard normal for $n \gtrsim 25$ (Hollander et al., 2014). Since we test for any deviation from $\mathrm{median}\,\Delta = 0$, the p-value doubles the upper-tail probability, $p = 2\big(1 - \Phi(|z|)\big)$, where Φ is the standard normal CDF. The same intuition applies: $|z|$ grows with both the magnitude of the per-target shift and the sample size n, and only the combination of a substantial intervention effect and enough paired targets drives $|z|$ large enough, and the p-value small enough, to rule out chance. We use Wilcoxon rather than a paired t-test because Δ is heavy-tailed and not approximately Gaussian across targets.

Multiple-testing correction. Across the two hypothesis tables we test 16 hypotheses in total: seven benchmark-side (H1–H7 in Table 22) and nine model-side (H1–H9 in Table 23). Each hypothesis is tested under both of our error metrics, MedAPE and MedAE, and we report a p-value for each, so the two tables together contain $16 \times 2 = 32$ tests. Because this is a multiple-comparison setting,

we assess significance against a Benjamini-Hochberg false-discovery-rate correction (Benjamini & Hochberg, 1995) over all 32 p-values, which raises the bar in proportion to the number of tests. Ordering the p-values and comparing the k-th smallest against $(k/32)\,\alpha$ at $\alpha = 0.01$, the procedure rejects the 17 tests with $p \leq 0.004$. These include both the MedAPE and MedAE tests for each of the eight factors we report as jointly supported (benchmark-side H3, H4, H5 and model-side H2, H3, H5, H8, H9), so all eight remain significant under both metrics after correction; no hypothesis that was not already rejected at $p < 0.01$ becomes significant.

E.1.1 BENCHMARK ANALYSIS

This appendix defines the benchmark-side hypotheses tested in Sec. 5.1 and reports two extensions of that analysis: the full 7×2 hypothesis $\times$ metric grid, and a per-benchmark predictability ranking that names which benchmarks are easiest and hardest for BENCHPRESS to predict.

Evaluation settings. We test seven hypotheses split into two families. H1–H3 probe benchmark-intrinsic features (low-rank fit, score level, score spread), each evaluated by univariate Spearman correlation between the feature and BENCHPRESS's per-benchmark error across all 133 targets. H4–H7 probe data availability and structural overlap with other benchmarks, each evaluated by a paired hide-half ablation: we intervene on the training matrix and compare prediction quality against an unintervened baseline (paired Wilcoxon over benchmarks).

Hypotheses and test results. The seven benchmark-side hypotheses of Sec. 5.1 are defined below; Table 22 reports their test results.

- *H1 Low-rank fit. Intuition:* BENCHPRESS fills scores from a simple two-factor pattern, so a benchmark whose scores already follow that pattern is easier to reconstruct. *Hypothesis:* a benchmark's column R^2 under the rank-2 SVD reconstruction is not associated with how well it can be predicted. *Feature:* column R^2 under the rank-2 reconstruction of the standardized, zero-imputed score matrix. *Test:* Spearman.
- *H2 Score level. Intuition:* on a very easy (or very hard) benchmark almost every model scores near the top (or bottom), so the scores bunch together and are easy to guess. *Hypothesis:* the overall score level (i.e., difficulty) of a benchmark is not associated with how well it can be predicted. *Feature:* median observed score per benchmark. *Test:* Spearman.
- *H3 Score spread. Intuition:* if all models score about the same on a benchmark, its scores are nearly flat and easy to guess; when scores are spread out, there is more room to be wrong. *Hypothesis:* the spread of scores across models on a benchmark is not associated with how well it can be predicted. *Feature:* standard deviation of observed scores per benchmark. *Test:* Spearman.
- *H4 Target coverage. Intuition:* the more of a benchmark's own scores we have already observed, the more the predictor has to work from when filling the rest. *Hypothesis:* reducing the amount of training evidence for a target benchmark does not change its prediction error. *Intervention:* for each target benchmark we first split its observed cells in half: one half is held out for evaluation and the other half remains available for training; we then compare the full training half against a version where three quarters of those training cells are removed. *Test:* paired Wilcoxon.
- *H5 Strong-neighbor presence. Intuition:* if another benchmark's scores track the target's closely, we can read the target off that neighbor; remove the neighbor and that shortcut disappears. *Hypothesis:* masking strongly correlated neighbor benchmarks does not change a target benchmark's prediction error. *Intervention:* for each target benchmark, mask every neighbor benchmark whose Pearson correlation with the target is at least 0.85 on shared models, then rerun BENCHPRESS on the target's held-out cells. *Test:* paired Wilcoxon.
- *H6 Strong-neighbor support. Intuition:* even with such a neighbor, the prediction can only be as good as the number of models the two benchmarks were both scored on. *Hypothesis:* reducing overlapping evidence from the strongest neighbor does not change a target benchmark's prediction error. *Intervention:* for each target benchmark, identify its strongest neighbor, keep only the models scored by both benchmarks, and compare the full shared-evidence condition against a version where three quarters of those overlapping neighbor cells are removed. *Test:* paired Wilcoxon.
- *H7 Same-category evidence. Intuition:* if a benchmark is predictable only because it shares a topic (e.g., other math benchmarks) with its neighbors, dropping all same-topic benchmarks should break it; if it stays accurate, topic was not what mattered. *Hypothesis:* masking same-

category benchmarks does not change a target benchmark's prediction error. *Intervention:* mask all same-category benchmarks during training (43 benchmarks $\times$ 10 seeds). *Test:* paired Wilcoxon.

Table 22: **Which features predict per-benchmark prediction quality?** H1–H3 use the Spearman rank correlation test across target benchmarks; H4–H7 use paired Wilcoxon signed-rank tests on hide-half ablations. The p-value is the probability of seeing an effect at least this large by chance if the listed (no-effect) hypothesis were true; smaller p means stronger evidence against it. We reject a hypothesis when $p < 0.01$ (bold), i.e., the data provide strong evidence that the factor does matter. Pink rows are rejected under both MedAPE and MedAE and are visualized in Figure 5.

Hypothesis	MedAPE ↓ (p-value)	MedAE ↓ (p-value)
H1 Low-rank fit	$p = 0.150$	$p = 0.036$
H2 Score level	$p < 0.001$	$p = 0.054$
H3 Score spread	$p < 0.001$	$p < 0.001$
H4 Target coverage	$p < 0.001$	$p < 0.001$
H5 Strong-neighbor presence	$p < 0.001$	$p < 0.001$
H6 Strong-neighbor support	$p = 0.209$	$p = 0.035$
H7 Same-category evidence	$p = 0.832$	$p = 0.725$

Table 22 reports the test results: three benchmark-side hypotheses are rejected under both error metrics (H3 score spread, H4 target coverage, and H5 strong-neighbor presence), and Figure 5 visualizes the corresponding effects.

Three hypotheses are rejected under both metrics, i.e., these factors do affect prediction quality. Rejecting H3 (score spread) means benchmarks with wider score ranges across models are harder to predict. Rejecting H4 (target coverage) and H5 (strong-neighbor presence) means a benchmark is easier to predict when it has many observed model scores, and when at least one strongly correlated neighbor remains in the training matrix. The remaining hypotheses (H1 low-rank fit, H2 score level, H6 strong-neighbor support, H7 same-category evidence) are not rejected under both metrics.

Full hypothesis $\times$ metric grid. Figure 5 in the main text visualizes the benchmark-side patterns that pass the joint-support criterion (H3, H4, and H5). For completeness, Figure 12 expands this to the full 7×2 grid: every active benchmark-side hypothesis (H1–H7) against both score-error metrics, using the same correlational and ablation setups as Table 22.

Per-benchmark predictability. We apply a direct cell-holdout test for each benchmark column. For each model we randomly hide half of its observed scores and predict them via the BENCHPRESS predictor from Sec. 3.1; we then aggregate the test-cell errors by benchmark column. This is repeated over 10 random seeds for stability.

Figure 13 shows the per-benchmark MedAPE for the evaluated benchmark columns. Roughly 71% (35/49) of benchmarks fall below the 15% MedAPE threshold, indicating they are inferable with limited additional information loss.

E.1.2 MODEL ANALYSIS

This appendix defines the model-side hypotheses tested in Sec. 5.1 and mirrors the benchmark-side extensions: the full 9×2 hypothesis $\times$ metric grid, and a per-model predictability ranking that names which models are easiest and hardest for BENCHPRESS to predict.

Evaluation settings. For each model we run the same hide-half evaluation (10 random splits with the full Logit Bias ALS pipeline) and aggregate the held-out predictions into per-model MedAPE and MedAE.

We test nine hypotheses split into three families. H1–H4 probe model-intrinsic features (size, type, score level, low-rank fit), each evaluated by univariate Spearman correlation against per-model error (H1 uses the $n = 25$ models with disclosed parameter counts; H2–H4 use all 84). H5–H8 probe data

availability and overlap with other models, each evaluated by a paired hide-half ablation that modifies the training matrix and measures the change in error. H9 tests temporal generalization via a rolling simulation: train only on older models and predict newer ones.

Hypotheses and test results. The nine model-side hypotheses of Sec. 5.1 are defined below; Table 23 reports their test results.

- *H1 Model size. Intuition:* larger models may behave more consistently, so they could be easier to predict from similar models. *Hypothesis:* model size is not associated with how well a model can be predicted. *Feature:* parameter count for the 25 models with disclosed sizes. *Test:* Spearman.
- *H2 Model type. Intuition:* reasoning and non-reasoning models score in noticeably different ways, so a model is easier to predict when others of its own kind are present. *Hypothesis:* whether a model is a reasoning model is not associated with how well it can be predicted. *Feature:* binary reasoning vs. non-reasoning indicator among models with annotated type. *Test:* Spearman.
- *H3 Score level. Intuition:* very strong or very weak models stand out from the pack, which could make them easier or harder to predict. *Hypothesis:* the overall score level (i.e., capability) of a model is not associated with how well it can be predicted. *Feature:* per-model median observed score. *Test:* Spearman.
- *H4 Low-rank fit. Intuition:* a model whose scores already follow BENCHPRESS's simple two-factor pattern is, by construction, one it can describe well. *Hypothesis:* a model's row R^2 under the rank-2 SVD reconstruction is not associated with how well it can be predicted. *Feature:* row-level R^2 under the rank-2 reconstruction of the standardized, zero-imputed score matrix. *Test:* Spearman.
- *H5 Strong-peer presence. Intuition:* if another model scores almost the same as the target across benchmarks, we can copy its scores; remove that peer and the copy is gone. *Hypothesis:* masking strongly correlated peer models does not change a target model's prediction error. *Intervention:* for each target model, mask all peer models whose Pearson correlation with the target is at least 0.95 on shared benchmarks, then rerun BENCHPRESS on the target's hide-half cells. *Test:* paired Wilcoxon.
- *H6 Strong-peer support. Intuition:* even with such a peer, the prediction can only be as good as the number of benchmarks the two models were both scored on. *Hypothesis:* reducing overlapping evidence from the strongest peer does not change a target model's prediction error. *Intervention:* for each target model and hide-half split, identify the strongest peer model (highest $|r|$, requiring $|r| \geq 0.95$), restrict to benchmarks observed by both the target and that peer, and drop nested prefixes $f \in \{0, 0.25, 0.5, 0.75\}$ of those overlapping peer cells before rerunning BENCHPRESS on the target's held-out cells; we compare $f{=}0$ against $f{=}0.75$. *Test:* paired Wilcoxon.
- *H7 Same-provider evidence. Intuition:* models from one provider (e.g., all GPT versions) are often built alike and score alike, so removing them tests whether that family resemblance is what made the model predictable. *Hypothesis:* masking same-provider variants does not change a target model's prediction error. *Intervention:* mask all same-provider rows (e.g. all GPT variants when predicting a GPT model) and rerun BENCHPRESS on the target's hide-half cells. *Test:* paired Wilcoxon.
- *H8 Observation count. Intuition:* the more of a model's own scores we have already seen, the more the predictor has to work from when filling the rest. *Hypothesis:* reducing the amount of training evidence for a target model does not change its prediction error. *Intervention:* compare the standard hide-half split against a more severe split that hides three quarters of each model's observed scores (Figure 5 shows the full trajectory across the evaluated hide fractions). *Test:* paired Wilcoxon.
- *H9 Training-anchor recency. Intuition:* a new model is easier to place when the models we train on include recent ones close to it in time and ability. *Hypothesis:* the recency of the training matrix is not associated with how well newly released models can be predicted. *Intervention:* sort all 84 models by release date and split into oldest, middle, and newest thirds; train the BENCHPRESS score predictor using only the oldest third or only the middle third, reveal a small number of benchmark scores for each newest-third target model, and predict the rest. The table reports the condition with three revealed benchmarks. *Test:* paired Wilcoxon.

Table 23 reports the full results, and Figure 5 visualizes the model-side hypotheses that are rejected under both error metrics: H2, H3, H5, H8, and H9.

Five hypotheses are rejected under both metrics, i.e., these factors do affect prediction quality. Reasoning models are easier to predict than non-reasoning ones (H2), and higher-scoring models are easier than lower-scoring ones (H3). Among the ablations, models with at least one strongly

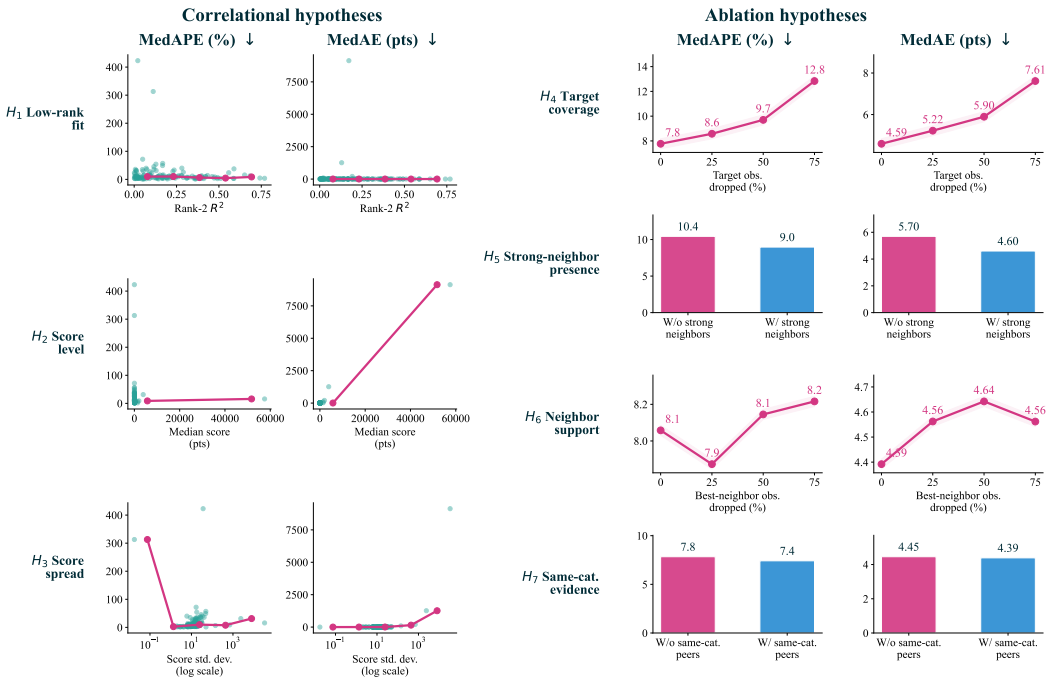


Figure 12: **All seven active benchmark-level hypotheses against both score-error metrics.** The left block shows H1–H3 (correlational hypotheses) and the right block shows H4–H7 (ablations). Columns within each block are MedAPE (↓) and MedAE (↓). Correlational rows show scatter + binned trend; ablation rows show line plots across drop fractions (H4, H6) or paired bars at the headline intervention (H5 with $|r| \geq 0.85$ peers; H7 with same-category peers).

Table 23: **What makes a model easy or hard to predict?** H1–H4 use the Spearman rank correlation test; H5–H8 use paired Wilcoxon signed-rank tests on hide-half ablations; H9 compares older vs. more recent training data. The p-value is the probability of seeing an effect at least this large by chance if the listed (no-effect) hypothesis were true; smaller p means stronger evidence against it. We reject a hypothesis when $p < 0.01$ (bold), i.e., the data provide strong evidence that the factor does matter. Pink rows are rejected under both MedAPE and MedAE and are visualized in Figure 5.

Hypothesis	MedAPE ↓ (p-value)	MedAE ↓ (p-value)
H1 Model size ($\log_{10}$ params)	$p = 0.101$	$p = 0.263$
H2 Model type	$p < 0.001$	$p = 0.003$
H3 Score level	$p < 0.001$	$p < 0.001$
H4 Low-rank fit	$p = 0.389$	$p = 0.042$
H5 Strong-peer presence	$p < 0.001$	$p = 0.004$
H6 Strong-peer support	$p = 0.033$	$p = 0.309$
H7 Same-provider evidence	$p = 0.011$	$p = 0.094$
H8 Observation count	$p < 0.001$	$p < 0.001$
H9 Training-anchor recency	$p = 0.002$	$p = 0.002$

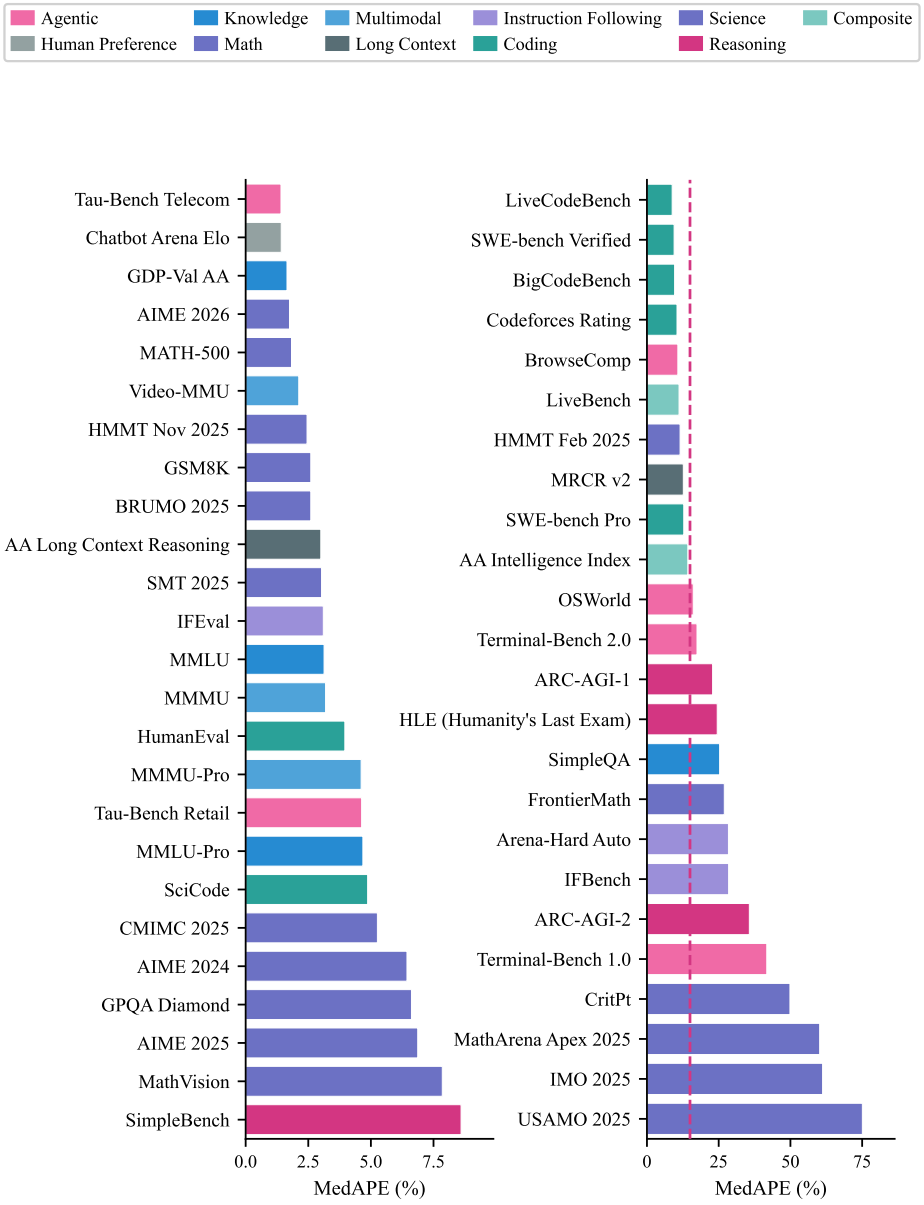


Figure 13: **Per-benchmark predictability.** For each model, half of observed scores are held out and predicted via BENCHPRESS; errors are aggregated by benchmark column (10 seeds). Benchmarks below the 15% MedAPE threshold (dashed red line) are well-predicted by others. Color = benchmark category.

correlated peer in the matrix are easier to predict (H5), models with more observed benchmark scores are easier (H8), and the temporal experiment shows that prediction quality on newer models improves when the training matrix contains more recent anchors rather than only the oldest third (H9). The H5 result matches what Zhang et al. (2025b) find: predictions are only reliable when the target model performs similarly to the models the predictor was fit on. The remaining hypotheses (H1 model size, H4 low-rank fit, H6 strong-peer support, H7 same-provider evidence) are not rejected under both metrics.

Full hypothesis × metric grid. Figure 5 in the main text visualizes five representative model-level hypotheses (H2, H3, H5, H8, H9) in the top-right and bottom blocks. For completeness, Figure 14 expands this to the paper-facing setting for each of the nine hypotheses (H1–H9) against both score-error metrics, using the same correlational, ablation, and temporal setups as Table 23.

Per-model predictability. Mirroring the per-benchmark probe in Sec. E.1.1, we apply the same half-per-model holdout but aggregate the test-cell errors by *model row* instead of benchmark column. For each model we randomly hide half of its observed scores and predict them via the BENCHPRESS predictor from Sec. 3.1, repeating over 10 random seeds for stability.

Figure 15 shows the per-model MedAPE for the 84 evaluated models. Roughly 88% (74/84) fall below the 15% MedAPE threshold and the median per-model MedAPE is 7.6%, indicating that most models are inferable from the rest of the matrix at limited additional information loss.

Figure 16 shows the same values as a distribution rather than a ranking: most models sit in a narrow band, and the models past the threshold form a thin right tail rather than a second mode.

E.2 ESTIMATING PREDICTION RELIABILITY

Sec. 5.2 adds a reliability estimator to the default BENCHPRESS score predictor from Sec. 3.1. This appendix spells out the three reliability estimators used there. If a predicted score will be used to decide whether to skip an expensive evaluation, the useful question is not only "what score would this model get?", but also "how much should we trust this prediction?" We therefore train reliability estimators that assign each cell predicted by the default BENCHPRESS score predictor a risk score, where larger risk means a less reliable prediction, and compare them by how well they identify safe-to-use predictions before the benchmark is run.

Reliability estimators. Each estimator is a small model trained on held-out cells from the training folds to predict, from features available *before* running the benchmark, how far off the Logit Bias ALS point estimate will be on a cell. All three models solve the same task: for a hidden model–benchmark cell, predict how large the absolute error of the fixed point prediction is likely to be. During training, the target is the held-out absolute error after a $\log(1 + x)$ transform. During evaluation, the reliability estimator may use the training matrix, the fixed Logit Bias ALS prediction, and auxiliary predictions computed from the training fold, but it never sees the hidden score itself.

Ensemble-spread reliability estimator. The ensemble-spread model asks whether plausible score predictors agree on the same hidden cell. If many plausible predictors disagree, this estimator should assign higher risk. It builds two stacks of alternative point predictions. The first stack measures local sensitivity of the selected score predictor: the three Logit Bias ALS configurations in the Sec. 3.1 grid with rank 2 and $\lambda \in \{0.01, 0.1, 1.0\}$. The $\lambda = 0.1$ member is the selected BENCHPRESS score predictor, and the other two show how much the prediction moves under the adjacent regularization strengths in the grid. The second stack measures disagreement with other strong full-coverage predictors. We sort transform–method configurations by median percentage error in Table 11, require coverage at least 99.9%, remove the selected Logit Bias ALS predictor, and keep the first 12 remaining configurations. In the checked-in run, these are Probit Bias ALS, Quantile Bias ALS, Identity Bias ALS, Quantile Soft-Impute, Logit Soft-Impute, Probit Soft-Impute, Arcsinh Bias ALS, Square root Bias ALS, Identity Soft-Impute, Logit Model-KNN, Probit Model-KNN, and Identity Model-KNN. For each prediction stack, we record four spread summaries: standard deviation, median absolute deviation, central 80% span, and the distance between the selected Logit Bias ALS prediction and the stack median. These eight nonnegative features are transformed with $\log(1 + x)$ before split-local standardization.

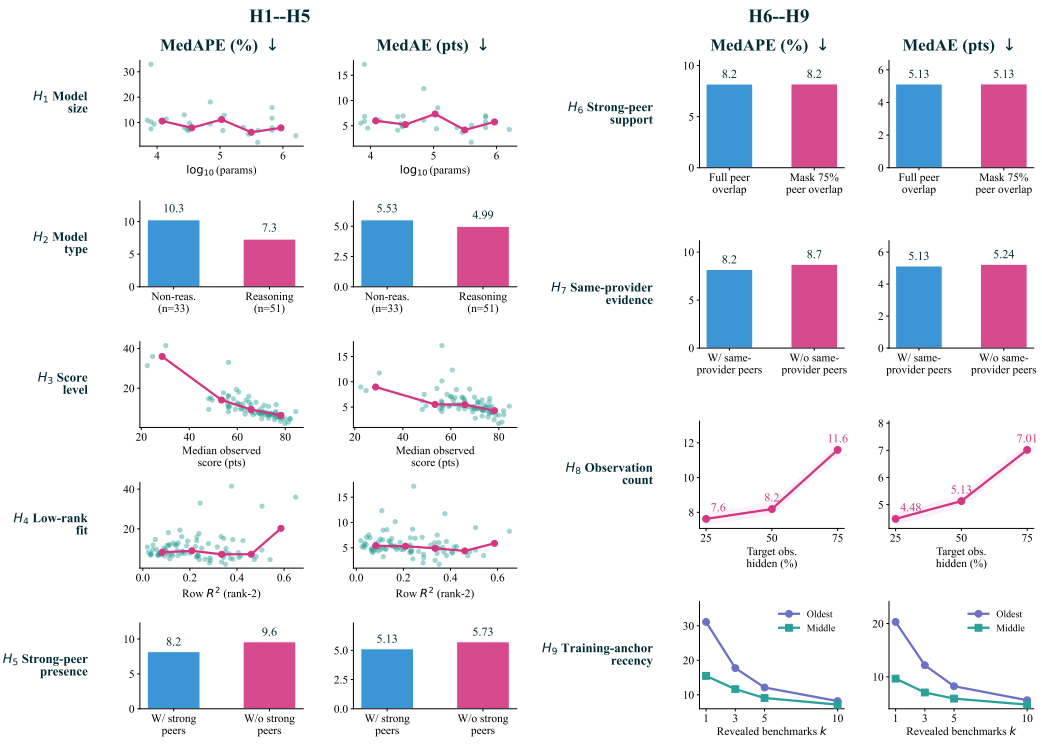


Figure 14: **Selected settings for all nine model-level hypotheses against both score-error metrics.** The left block shows H1–H5 and the right block shows H6–H9. Columns within each block are MedAPE (↓) and MedAE (↓). H1–H4 are correlational rows (H2 grouped bars), H5–H8 are ablations, and H9 is temporal. Ablation rows show paired bars at the headline intervention (H5: $|r| \geq 0.95$ peers; H6: 75% strongest-peer overlap mask; H7: same-provider evidence) or a line across hide fractions (H8). H9 compares oldest vs. middle training anchors for the displayed revealed-benchmark counts $k \in \{1, 3, 5, 10\}$; secondary H9 settings with $k = 8$ and $k = 15$ are not plotted in this figure.

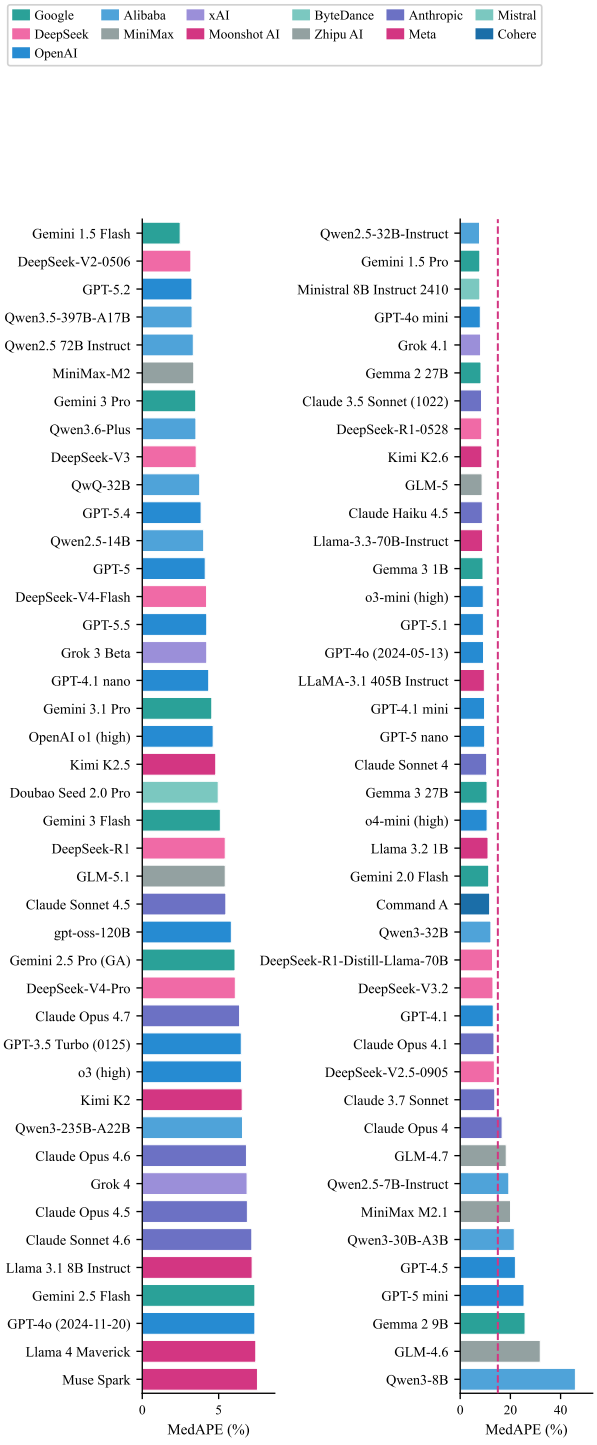


Figure 15: **Per-model predictability.** For each model, half of observed scores are held out and predicted via BENCHPRESS; errors are aggregated by model row (10 seeds). Models below the 15% MedAPE threshold (dashed red line) are well-predicted by others. Color = provider.

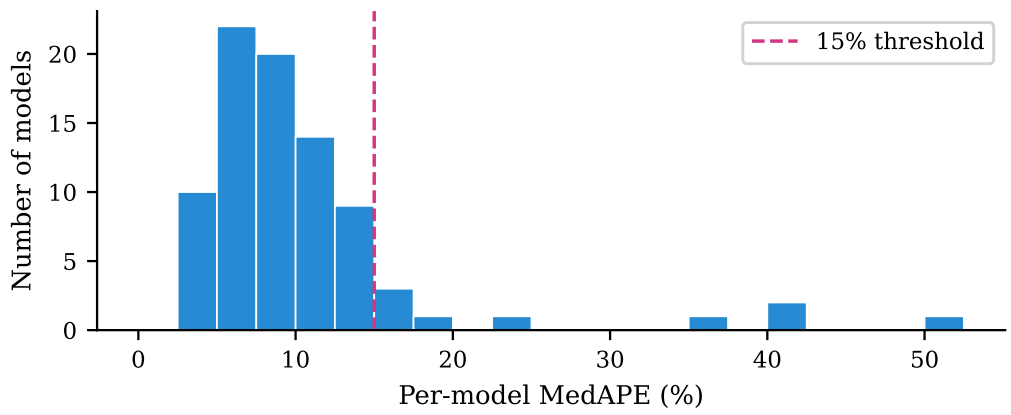


Figure 16: **Distribution of per-model predictability.** The per-model MedAPE values of Figure 15, binned across the evaluated models. Dashed line: the 15% threshold.

Matrix-support reliability estimator. The matrix-support model ignores alternative predictors and uses only evidence available in the observed score matrix. This estimator asks whether the cell is easy to infer from observed scores on correlated peer models and benchmarks. For the target model, it records the number of observed benchmark scores and the median observed score. For the target benchmark, it records the number of observed model scores, the median observed score, and the standard deviation of observed scores. It also records the strongest peer model for the target model and the strongest neighboring benchmark for the target benchmark, where "strongest" means highest absolute Pearson correlation over shared observed scores in the training matrix. The peer-model features are its absolute correlation with the target model and the number of shared observed benchmarks. The benchmark-neighbor feature is its absolute correlation with the target benchmark. We do not include benchmark-neighbor overlap because the stricter H6 ablation in Sec. 5.1 does not support it as a joint benchmark-side reliability factor. These features reuse the signals from Sec. 5.1, in the same order as the hypothesis tables. From the benchmark side: median score (H2), score spread (H3), observation count (H4), and strongest-neighbor correlation (H5). From the model side: median score (H3), strongest-peer correlation (H5), strongest-peer overlap (H6), and observation count (H8). Benchmark-side median score did not reach joint significance, while model-side median score did; we keep the benchmark-side score as a low-cost control.

Hybrid reliability estimator and calibration. The hybrid reliability estimator concatenates the ensemble-spread features and the matrix-support features, then uses the same risk-model selection procedure as the two single-signal models. It can learn cases where predictor disagreement is enough to flag risk, cases where sparse structural support is enough to flag risk, and cases where both signals reinforce each other. Before fitting any of the three learned reliability estimators, each feature column is clipped at zero, transformed with $\log(1+x)$, and standardized using only the training split. Each cell appears in one fold per seed, so for every evaluated fold, candidate risk models are trained on the other folds after removing every cell that also appears in the evaluated fold, and the architecture is selected inside those training folds from a linear ridge model with zero hidden layers, one ReLU MLP layer of 16 units, one ReLU MLP layer of 32 units, or two ReLU MLP layers of 64 and 32 units. Concretely, the remaining training cells are split by cell for architecture selection: cells whose index (model index times 133 plus benchmark index) is divisible by 5 form the inner validation set and the rest form the inner training set, giving roughly a 4:1 split in which all copies of a cell fall on the same side. If this modulo-5 split leaves too few validation or training cells, we fall back to cell index modulo 3, giving roughly a 2:1 split. Across the 30 evaluated folds, the hybrid estimator selected ridge, 16, 32, and 64/32 hidden units in 3, 12, 5, and 10 folds; the ensemble-spread estimator selected them in 21, 3, 1, and 5 folds; the matrix-support estimator selected 16, 32, and 64/32 hidden units in 11, 8, and 11 folds. After the architecture is selected, the MLP variants use ReLU activations, Adam, ℓ_2 penalty 10^{-3}, learning rate 3×10^{-3}, a separate 15% early-stopping validation fraction within the fitting routine, 25 no-improvement iterations, at most 500 iterations, and deterministic seeds derived

from base seed 42. Each fitted model outputs a risk score, where larger values mean less reliable point predictions. For display, we calibrate this ordering into a trust probability: the probability that predictions with similar risk fall within a chosen number of score points of the reported score. The display calibration bins held-out cells by hybrid risk, estimates the empirical within-tolerance probability in each bin, enforces a monotone nonincreasing map from risk to trust probability, and interpolates this map for displayed cells. For prediction intervals, we apply the same leave-fold-out conformal wrapper to each reliability estimator: on all folds except the evaluated fold, after again removing the evaluated fold's cells, take the 90th percentile of $|\hat{s} - s|/r$, multiply the evaluated fold's risk score r by that scale, and center the resulting 90% interval at the fixed Logit Bias ALS point prediction $\hat{s}$. Table 24 reports the resulting interval widths at three coverage levels. The hybrid 90% intervals cover 89.9% of held-out cells; on the lowest-risk third of cells they cover 88.4% with a median width of 15.4 points, and on the highest-risk third they cover 91.1% with a median width of 51.6 points.

Table 24: **Conformal interval widths** (score points) at three coverage levels; lower is sharper. The hybrid row is shaded.

Method	80%	90%	95%
Ensemble spread	19.24	27.32	35.93
Matrix support	21.47	31.48	42.34
Hybrid	**19.85**	**27.96**	**37.01**

Evaluation setting. All reliability estimators are evaluated on the same held-out folds as the point-prediction comparison; for every hidden score, they may use the training matrix, the fixed Logit Bias ALS point prediction, and auxiliary quantities derived from the training fold, but not the held-out value. Once the reliability estimator outputs a risk score, the main-text evaluation treats it as a ranking signal: predictions with lower risk should have lower realized error. We evaluate reliability ranking with a risk-coverage curve: sort cells from lowest to highest risk and plot MedAE after keeping the most trusted 100%, 80%, 60%, 40%, or 20% of cells. The risk-coverage curve in Sec. 5.2 therefore asks whether the risk score can identify predictions that are safe enough to use for triage, while flagging predictions that should still trigger benchmark runs.

Results. When keeping only the most-trusted 20% of cells, the hybrid estimator lowers selective MedAE to 1.97 score points, beating both single-feature variants (Figure 6); at 40% and 60% kept, its MedAE is 2.56 and 3.13. We therefore use the **hybrid reliability estimator** as the reliability layer: it takes both ensemble spread and matrix support into account, and assigns each prediction a risk score that identifies whether the prediction is safe to use.

Interval width versus number of observed scores. The widths above condition on each model's full training split. To isolate how an interval narrows as a single benchmark accumulates observations, we run a controlled variant that varies only the number of revealed scores. We keep the 23 benchmarks with at least 30 observed model scores, hold out the same 10 scores per benchmark for evaluation, and reveal $k \in \{5, 10, 15, 20\}$ of the remaining scores in that column as conditioning evidence while hiding the rest of the column and keeping all other columns observed. We predict the 10 held-out cells with the fixed Logit Bias ALS predictor and form leave-one-benchmark-out conformal 80% intervals at each k. Table 25 shows that empirical coverage stays near the 80% target while the median interval width falls from 32.3 to 23.0 score points and the median absolute error drops from 5.8 to 4.7 as more scores are revealed.

F Related Work

Collected score matrix. HELM (Liang et al., 2023) and Every Eval Ever (EEE) (Batzner et al., 2026) aggregate scores across many models and benchmarks. For our study of score predictability, we canonicalize near-duplicate model variants and benchmark configurations before filtering by observation coverage. Otherwise, scores from nearly identical settings can make prediction artificially easy, obscuring whether scores can be predicted across models and benchmarks. Sec. B.1 describes

Table 25: **Prediction intervals sharpen as more scores are observed.** Restricting to benchmarks with at least 30 observed model scores, holding out the same 10 evaluation cells, and revealing k conditioning scores per benchmark. Coverage is the empirical fraction of held-out cells inside the 80% interval; width is the median interval width in score points.

Revealed scores k	MedAE	80% coverage	Median width
5	5.83	0.799	32.28
10	4.96	0.798	25.32
15	4.93	0.800	23.32
20	4.69	0.798	23.04

this curation, and Sec. C.5 evaluates BENCHPRESS on matrices independently curated from HELM and EEE.

Low-rank structure in evaluation. A recurring finding is that benchmark reporting is highly redundant (Burnell et al., 2023b): scores are governed by a few latent capability factors that recur across very different model and benchmark collections. On 27 HELM (Liang et al., 2023) tasks over 29 models, Burnell et al. (2023a) recover three factors (reasoning, comprehension, core language modeling); scaling to 591 Open LLM Leaderboard models across 12 benchmarks, Ilić & Gignac (2024) find a single dominant g-factor explaining 85% of the variance; Ho et al. (2025) place model capabilities and benchmark difficulties on a single scale; on the Epoch AI Capabilities Index, Burnham (2025) reduce scores to a general-capability axis plus a provider-specific residual via PCA; and Maimon et al. (2026) recover low-rank skill factors from an LLM benchmark matrix. This redundancy invites prediction: Ruan et al. (2024) build a low-dimensional capability space from standard benchmarks and use it to estimate selected downstream capabilities. BENCHPRESS builds on the same premise but works from a heterogeneous, frontier-era public matrix, where Sec. 2.2 likewise recovers an effectively rank-2 geometry.

Benchmark compression and design. Perlitz et al. (2024a) showed HELM evaluation can be compressed 100× with minimal ranking reliability loss; their follow-up (Perlitz et al., 2024b) formalized best practices for evaluating whether benchmarks agree with one another. Ni et al. (2024) constructed MixEval, a single compact benchmark from web-query-matched items that achieves 0.96 Chatbot Arena correlation. These approaches select or design a fixed evaluation suite *a priori*. BENCHPRESS instead predicts missing scores from whatever benchmarks happen to be available, requiring no fixed probe set: a practitioner can feed in MMLU (Hendrycks et al., 2021) and GPQA (Rein et al., 2024) today, or LiveCodeBench and AIME tomorrow, without reconfiguration.

Item-level subset selection. A complementary line of work reduces cost *within* individual benchmarks by selecting which test items to run. *IRT-based* methods include MetaBench (Kipnis et al., 2025), which uses item response theory to keep 3% of items across six benchmarks while preserving aggregate conclusions, and tinyBenchmarks (Polo et al., 2024), which builds on Anchor Points using IRT to pick informative items. *Correlation- or embedding-based* methods include Anchor Points (Vivek et al., 2024), which selects items via cross-model correlations; Scales++ (Bean et al., 2025), which uses cognitive-scale embeddings to reduce cost 18× at 2.9% MAE without prior model evaluations; DISCO (Rubinstein et al., 2026), which condenses sets by selecting items where models disagree most; SubLIME (Saranathan et al., 2025), which trains a correlation predictor for compact subsets; EssenceBench (Wang et al., 2026), which applies genetic algorithms for up to 200× compression; and Zhou et al. (2025), who exploit low-rank structure at the example level for up to 20× speedups. Most of these methods require instance-level pass/fail data across many models to calibrate item selection; Scales++ is a notable exception. BENCHPRESS requires only aggregate scores and predicts *across* benchmarks, a complementary approach that could be combined with item-level methods for end-to-end savings.

Score prediction. Closest to our work are methods that predict aggregate benchmark scores directly. The nearest methodological predecessor is Schram et al. (2023), who applied Bayesian matrix factorisation to cross-lingual NLP performance, though in a different domain (languages × tasks, not

LLMs $\times$ benchmarks); on LLM scores specifically, Zhang et al. (2024) applied collaborative filtering, Polo et al. (2025) used latent skill models for cross-benchmark prediction, Pacchiardi et al. (2025) benchmarked predictors of instance-level LLM success, and Ye et al. (2023) showed BIG-bench is 95%+ predictable. Fisch et al. (2026) also use matrix completion over model-prompt evaluation matrices to reduce annotation cost with valid confidence intervals, while BENCHPRESS completes sparse public aggregate scores and studies probe selection and prediction reliability. A second group predicts performance from a fixed panel of densely observed evaluations: Aitchison et al. (2023) select five games from the fully observed Arcade Learning Environment to estimate median performance across the 57-game suite, and Ruan et al. (2024) construct capability measures from a fixed block of standard benchmarks to fit target-specific predictors. Both assume a pre-specified input panel observed across historical models or agents; we compare directly against the latter in Sec. C.4, while Atari-5 does not transfer because such a panel is essentially never fully observed in our matrix. Others predict from outside the score matrix entirely: Park et al. (2025) use LLMs to predict scores from text descriptions alone, with no execution (we revisit this comparison empirically in Sec. 3.2), and Koh et al. (2026) train a small proxy model to transfer reasoning performance to larger models, whereas BENCHPRESS needs no model access at all. BENCHPRESS instead performs general score prediction on an uncontrolled public matrix, where any subset of a new model's scores may be observed and the rest are targets. We differ from these methods in three ways: (1) we operate at substantially larger scale and on a frontier-era snapshot (84 models $\times$ 133 benchmarks, including post-2024 reasoning, coding, and agentic suites), (2) we compare 84 transform–method configurations head-to-head on the same data, and (3) we provide explicit failure analysis, showing where and why prediction breaks.