# TWO HEADS ARE BETTER THAN ONE: AGGREGATING WEAKER LLMS FOR BETTER FORECASTS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Large language models (LLMs) are increasingly used to forecast real-world events, but access to the strongest individual forecaster may be costly or otherwise constrained. We study weak-to-strong forecast aggregation: can individually weaker LLM forecasters be aggregated to outperform a stronger forecaster? Using ForecastBench (Karger et al., 2025), we evaluate 70 LLM forecasters across 16 comparison groups, each with more than 1,000 shared subquestions, yielding 1,121 weaker-model pairs. Within each group, we identify the strongest individual by test Brier score and evaluate aggregates composed exclusively of weaker forecasters, with aggregation weights learned on separate training data. We find substantial evidence of weak-to-strong improvement. Learned linear pooling identifies a weaker pair that matches or outperforms the strongest individual in 11 of 16 groups and comes within 5% of its Brier score in all 16 groups. We also find that these improvements do not rely on having a near-best constituent and are generally accompanied by good calibration. Additional analyses show that adding more models does not consistently improve performance, and competitive weaker-model aggregates also remain available under practical constraints.

## 1 INTRODUCTION

Forecasting is an important and fundamental task in modern economic decision-making, informing decisions ranging from production and inventory planning to investment, financial markets, and public policy, etc. Accordingly, how to produce good forecasts has been studied for decades across statistics, economics, operations research, and machine learning (see, e.g., Winters, 1960; Armstrong, 2001; Holt, 2004; Gruver et al., 2023; Goswami et al., 2024). While much of this classical literature focuses on forecasting future values from historical data, many important decisions instead require forecasts about future real-world events, for which relevant information may be distributed across heterogeneous textual, numerical, and contextual sources.

Recent remarkable advances in large language models (LLMs), which are trained on large-scale text corpora, have created new possibilities for automated forecasting of future events. LLMs can draw on broad cross-domain knowledge, reason over textual evidence, and, when combined with retrieval or search, incorporate up-to-date information relevant to future events. These capabilities have motivated a rapidly growing literature on LLM-based forecasting, with recent work developing and evaluating them on real-world future events (see, e.g., Zou et al., 2022; Halawi et al., 2024; Karger et al., 2025; Douven, 2026; Yang et al., 2026; Zeng et al., 2026).

As more LLM forecasters become available, however, the relevant question need not be which individual forecaster is the most accurate. When forecasts from multiple models are available, we can instead combine them into a single forecast. Forecast aggregation has a long history in statistics and forecasting, where combining predictions from different sources can improve predictive accuracy (Bates & Granger, 1969; Clemen, 1989). Recent work finds similar benefits for LLM forecasters. Schoenegger et al. (2024), for example, aggregate probabilistic forecasts from twelve LLMs and show that the resulting LLM crowd can achieve accuracy comparable to a human forecasting crowd. ForecastBench also considers ensemble forecasts constructed from multiple LLMs and prompting strategies (Karger et al., 2025). More recent studies further show that individual forecasting accuracy alone does not determine a model's value in an aggregate: weaker but more diverse forecasters may contribute complementary information when their forecasts differ from those of stronger models



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.

(Aitchison et al., 2026), and learned aggregation methods can outperform individual LLM forecasters and classic aggregation rules (Douven, 2026).

These findings suggest that what a collection of LLM forecasters can achieve may not be fully captured by their individual performance rankings. This distinction becomes particularly important when access to the strongest individual forecaster cannot be taken for granted. A stronger model may be more expensive to use, available only through an external API, or incompatible with settings that require open-weight or locally deployable models. In such cases, the relevant question is not whether combining several forecasts, including the strongest one, can further improve performance. Rather, we ask a more stringent question of weak-to-strong aggregation: *Can two or more LLM forecasters that are individually weaker than a stronger forecaster be aggregated to match or outperform that stronger forecaster?*

The possibility of such weak-to-strong improvement is suggested by a broader literature showing that weaker models can collectively outperform stronger ones. Classical boosting results show that weak learners can be combined to construct a substantially stronger model (Schapire, 1990). Related work on LLMs similarly shows that combining multiple outputs or weaker models can improve on the performance of stronger individual models (see, e.g., Wang et al., 2022; Du et al., 2024; Hu et al., 2025). These results provide evidence that individual model strength does not necessarily determine the strength of an aggregate. Yet whether this weak-to-strong phenomenon extends to probabilistic forecasting of real-world events remains largely unexplored. We therefore systematically study whether proper aggregation can turn two individually weaker LLM forecasters into a forecast that matches or outperforms a stronger individual forecaster.

**Our contributions and results.** We summarize our main contributions and results as follows:

- We introduce and systematically study weak-to-strong forecast aggregation for LLMs: whether individually weaker LLM forecasters can be aggregated to match or outperform a stronger forecaster without using its forecast. Using ForecastBench (Karger et al., 2025), we organize model configurations into representative forecasters, construct comparison groups with sufficient shared forecast coverage, identify the strongest individual in each group as the benchmark, and evaluate aggregates formed only from weaker forecasters. Our main experiments cover 70 LLM forecasters, 16 comparison groups, each with more than $1,000$ shared subquestions, and $1,121$ weaker-model pairs, and compare both learned and fixed aggregation rules.
- We find substantial evidence of weak-to-strong improvement. At the group level, there exists an aggregation rule (i.e., learned linear pooling) that identifies a weaker pair that outperforms the strongest model in 11 of 16 groups and comes within 5% of its Brier score in all 16 groups. Moreover, we find that these gains do not rely on having a near-best constituent and are generally accompanied by good calibration. Our additional analysis shows that learning the aggregation weights matters, while adding more models does not consistently improve performance; the gains also vary across question categories and remain available under lower-input-price and open-weight constraints. Finally, our robustness checks, control experiment, and case study show that the weak-to-strong pattern persists under a substantially higher shared-coverage threshold, cannot generally be explained by individual recalibration alone, and can depend on the choice of aggregation rule.

**Paper organization.** Section 2 introduces the forecasting setting, evaluation metrics, and aggregation rules. Section 3 describes the dataset, construction of comparison groups, and experimental protocol. In Section 4, we present our main results, including pairwise and multi-model aggregation, and category-specific analysis. Section 5 concludes. The appendix provides additional related work, robustness checks, practical-constraint analysis, case studies, and supplementary results.

## 2 PRELIMINARIES

We study whether aggregating individually weaker forecasters can outperform a stronger forecaster. In this section, we describe the forecasting task and introduce our performance metric; we then introduce aggregation rules considered in this work.

### 2.1 Forecasting Questions and Forecasts

**Forecasting questions.** We use the dataset from ForecastBench (Karger et al., 2025), which includes *standard questions* and *combination questions*. We denote the question set by $\mathcal{Q} = \{Q_i\}_{i\in[n]}$, where $n$ denotes the total number of questions in $\mathcal{Q}$. Each question $Q_i$ contains a number $s_i$ of binary subquestions $\{Q_{ij}\}_{j\in[s_i]}$, each of which resolves to Yes or No. We treat a standard question as the case $s_i = 1$, with $Q_i = \{Q_{i1}\}$. We use this common representation throughout the paper.

- **Standard question:** "Will it rain tomorrow in City A?" The question itself is its sole subquestion.
- **Combination question:** A question about tomorrow's weather in City A and City B may contain subquestions, such as "Will it rain in both cities?" and "Will it rain in City A but not in City B?"

The binary outcome $y_{ij} \in \{0, 1\}$ equals 1 if subquestion $Q_{ij}$ resolves to Yes and 0 otherwise.

**Forecasters and forecasts.** A *forecaster* is a model configuration specified by its model version, prompt, and information condition. For a given forecast date and resolution date, the forecast $p_{ij}^m \in [0, 1]$ is the probability that forecaster $m$ assigns to $y_{ij} = 1$ before observing the outcome.

### 2.2 Evaluation Metrics

We evaluate forecasts along two dimensions: *accuracy*, which measures how closely predicted probabilities match realized outcomes, and *calibration*, which measures whether forecasted probabilities agree with observed outcome frequencies. For evaluation metrics, we use the Brier score for accuracy and expected calibration error (ECE) for calibration. We evaluate all individual and aggregated forecasts in each comparison on the same pool of questions. We use the evaluation set $\mathcal{I} \subseteq [n]$ to denote the indices of the questions included in the evaluation pool. For each forecaster $m$, we denote the vector of its forecasts for all evaluated questions and their subquestions by $\boldsymbol{p}^m = \left(p_{ij}^m\right)_{i\in\mathcal{I},\, j\in[s_i]}$.

**Brier score.** The Brier score (Brier, 1950) is a scoring rule that evaluates a probability forecast against its realized outcome. The expected score is minimized when a forecaster forecasts the true outcome probability. We compute the Brier score for each question by averaging the squared losses of its subquestions:

$$\mathrm{BS}_i[\boldsymbol{p}^m] = \frac{1}{s_i} \sum\nolimits_{j\in[s_i]} \left(p_{ij}^m - y_{ij}\right)^2 .$$

The overall Brier score is the average across questions $\mathrm{BS}_{\mathcal{I}}[\boldsymbol{p}^m] = \frac{1}{|\mathcal{I}|} \sum_{i\in\mathcal{I}} \mathrm{BS}_i[\boldsymbol{p}^m]$. Here each question therefore receives equal weight, regardless of its number of subquestions. A lower Brier score indicates better accuracy.

**Expected calibration error.** A reliable forecast should match the observed frequencies; that is, among predictions assigning probability 0.7 to Yes, approximately 70% should resolve to Yes. This property is referred to as *calibration* in the machine learning literature (see, e.g., Guo et al., 2017). Following the literature (see, e.g., Yang et al., 2026), we estimate calibration error by pooling predictions with similar probabilities. In particular, we partition $[0, 1]$ into $B = 10$ equal-width bins and index them by $b \in [B]$. We denote the total number of evaluated subquestions by $N = \sum_{i\in\mathcal{I}} s_i$. For forecaster $m$, we define $\mathcal{B}_b^m$ as the set containing the subquestion indices $(i, j)$ such that the forecast $p_{ij}^m$ falls in bin $b$. For each nonempty $\mathcal{B}_b^m$ containing indices $(i, j)$, we denote the mean of the corresponding forecasts $p_{ij}^m$ by $\overline{p}_b^m$ and the mean of their observed outcomes $y_{ij}$ by $\overline{y}_b^m$, respectively.

The empirical ECE averages the absolute differences between these two quantities, weighted by each bin's share of subquestions:

$$\mathrm{ECE}_{\mathcal{I}}[\boldsymbol{p}^m] = \sum\nolimits_{b\in[B]} \frac{|\mathcal{B}_b^m|}{N} \left|\overline{p}_b^m - \overline{y}_b^m\right| .$$

ECE gives each evaluated subquestion equal weight. We note that ECE and the Brier score are not equivalent: a lower ECE does not necessarily imply a lower Brier score.

### 2.3 Aggregation Rules

We aggregate multiple probability forecasts for the same subquestion into a single forecast. We refer to the selected set of forecasters as a *pool* and denote its size by $k$. Within a pool, we index

forecasters by $m \in [k]$ and collect their forecasts in the vector $\boldsymbol{p} = (p^1, \ldots, p^k)$. We then define an aggregation rule $f : [0,1]^k \to [0,1]$ as a deterministic function that maps this vector to a single probability forecast.

Our main analysis focuses on two data-driven aggregation rules, *linear pooling* and *log-odds pooling*, whose coefficients are learned from historical forecasts. These rules allow the contribution of each forecaster to be adapted to the candidate pool and form the basis of our main experiments. We introduce them first in Section 2.3.1. For comparison, we also consider parameter-free classical pooling rules and, for pairwise aggregation, fixed rules from the robust forecast aggregation literature (Arieli et al., 2018; Kong et al., 2024; Guo et al., 2025b; Chen et al., 2026).

### 2.3.1 Learned Aggregation Rules

Our primary aggregation rules use historical data to learn a separate nonnegative coefficient for each forecaster in the pool. These coefficients allow the fitted rule to adjust both the relative contributions of the forecasts and the overall scale of their combination. We denote the coefficient assigned to forecaster $m$ by $w_m$ and collect the coefficients in the vector $\boldsymbol{w} = (w_1, \ldots, w_k) \in \mathbb{R}_+^k$. Using these coefficients, we consider *log-odds pooling* and *linear pooling*.[1]

**Log-odds pooling.** We combine the forecasts in log-odds space using the learned coefficients, then convert the result back to a probability: in particular, letting $\text{LOGIT}(x) = \ln(x/(1-x))$ and $\sigma(z) = 1/(1+\exp(-z))$, we obtain $f^{\log}_{\boldsymbol{w}}(\boldsymbol{p}) = \sigma(\sum_{m\in[k]} w_m \text{LOGIT}(p^m))$.

**Linear pooling.** We also combine the forecasts directly in probability space to obtain $f^{\text{lin}}_{\boldsymbol{w}}(\boldsymbol{p}) = \min\{1, \sum_{m\in[k]} w_m p^m\}$.

For each pool and each learned rule, we fit a separate coefficient vector by minimizing the training Brier score over $\mathbb{R}_+^k$, following the data-splitting procedure in Section 3. Once fitted, we freeze the coefficients and evaluate the same rule using Brier score and ECE on the test data. For the ECE analysis, we also maintain the Brier-based group benchmark and candidate pools specified in Section 3.2.

### 2.3.2 Classical Aggregation Rules

We consider four classical rules that combine forecasts without fitting parameters to historical data.

**Quasi-arithmetic means.** We present the first three rules as instances of the quasi-arithmetic mean (Bullen, 2003; Douven, 2026). We transform each forecast, average the transformed values, and apply the inverse transformation. We denote the transformation by $g$, which is required to be continuous and strictly monotone on an interval containing the forecasts. We write this common construction as

$$f_g(\boldsymbol{p}) = g^{-1}\left(\frac{1}{k}\sum\nolimits_{m\in[k]} g(p^m)\right) .$$

We obtain different pooling rules by choosing different transformations.

- *Arithmetic mean (AM).* We choose $g(x) = x$ to obtain $f_{\text{AM}}(\boldsymbol{p}) = k^{-1}\sum_{m\in[k]} p^m$. Conventionally, this mean is also referred to as the *Simple mean*.
- *Harmonic mean (HM).* We choose $g(x) = 1/x$ to obtain $f_{\text{HM}}(\boldsymbol{p}) = (k^{-1}\sum_{m\in[k]} 1/p^m)^{-1}$.
- *Log-odds mean (GM).* This rule averages forecasts in log-odds space and converts the result back to a probability. We choose $g = \text{LOGIT}$, where $\text{LOGIT}(x) = \ln(x/(1-x))$, and denote its inverse by $\sigma(z) = 1/(1+\exp(-z))$ for $z \in \mathbb{R}$. The resulting rule is $f_{\text{GM}}(\boldsymbol{p}) = \sigma(k^{-1}\sum_{m\in[k]} \text{LOGIT}(p^m))$. The label GM refers to taking the geometric mean of the forecast odds before converting back to a probability.

**Median.** We use the median as the fourth classical baseline when we aggregate more than two models. Under this rule, we order the forecasts and take the middle value, averaging the two middle values when the pool size $k$ is even: $f_{\text{med}}(\boldsymbol{p}) = \text{median}(p^1, \ldots, p^k)$.

[1] We follow the literature (see, e.g., Arieli et al., 2018) to adopt the same boundary convention for all aggregation rules. If at least one forecast is 0 and none is 1, we return 0; if at least one forecast is 1 and none is 0, we return 1. When both 0 and 1 appear, we return 0.5.

**Robust aggregation rules.** Beyond the classical aggregation rules above, for pairwise aggregation we also evaluate four rules from the robust forecast aggregation literature (see Section B for details).

## 3 Experimental Setups and Dataset Statistics

In this section, we describe our experiment setups and report some dataset statistics that are relevant to this work. We construct our experimental comparisons in two stages. We first select representative forecasters from the available model configurations, then organize them into groups with sufficient shared forecast coverage. Within each group, we compare individual and aggregated forecasts on the same subquestions. We describe this construction below, followed by the training procedure and evaluation criteria.

### 3.1 Data and Comparison Groups

We use probability forecasts and resolved outcomes from ForecastBench (Karger et al., 2025). The data catalog contains 8,556 questions and 22,162 binary subquestions, described in Section 2.1. The raw forecasts come from 313 model configurations covering 30 forecast rounds from 2024-07-21 to 2026-06-07. The corresponding questions are resolved between 2024-07-25 and 2026-09-11. To limit the representation of closely related model configurations, we organize them into *model families*. Configurations in the same family share a model series and capability variant, but may differ in release date, prompt, or information condition. For example, GPT-5 configurations with different prompts belong to the same family, whereas GPT-5 and GPT-5-Mini belong to different families. Within each family, we select the configuration with forecasts for the largest number of distinct subquestions as its *representative*. This gives 91 family representatives, whose set we denote by $\mathcal{M}$. We use these representatives to construct the comparison groups.

**Comparison groups and shared coverage.** Forecast coverage differs across the selected forecasters. To compare them on a common sample, we form groups whose members have forecasted a sufficiently large set of shared subquestions. In particular, we index forecasters by $m \in \mathcal{M}$. For each forecaster $m$, we define its coverage set $\mathcal{E}_m$ as the question-subquestion pairs $(i, j)$ for which it provides a probability forecast.

We form a *comparison group* by selecting a subset of forecasters $S \subseteq \mathcal{M}$. To compare these forecasters on the same subquestions, we use only those forecasted by every member. We denote this shared coverage by $\mathcal{E}_S = \bigcap_{m \in S} \mathcal{E}_m$. To guarantee adequate per-group data for our evaluation, we require each group to contain at least three forecasters and at least $N$ shared subquestions, where $N$ is the coverage threshold. A group is therefore *eligible* if $|S| \geq 3, |\mathcal{E}_S| \geq N$. With at least three forecasters, we can combine the forecasts of two members and compare the result with a third forecaster who is not part of the aggregate.

An eligible group may contain many smaller eligible groups. To reduce the number of comparisons, we retain only *maximal groups*: eligible groups to which no additional forecaster can be added while maintaining at least $N$ shared subquestions. Formally, an eligible group $S$ is maximal if

$$|\mathcal{E}_S \cap \mathcal{E}_m| < N \quad \text{for every } m \in \mathcal{M} \setminus S.$$

We keep all groups satisfying this condition. These groups may differ in size and share some forecasters. We evaluate them separately because merging distinct maximal groups would reduce their shared coverage below $N$. For each group, we use the full set of shared subquestions to construct the training and test sets; $N$ specifies a minimum rather than a fixed sample size.

**Construction statistics.** We apply this procedure at coverage thresholds $N \in \{1{,}000, 4{,}000\}$. At $N = 1{,}000$, retaining only maximal groups reduces the number of comparison groups from 567,316 to 16. Across both thresholds, 70 of the 91 family representatives appear in at least one retained group and constitute the forecasters evaluated in our main experiments. Table 1 summarizes the number of distinct forecasters, the number of retained groups, and the ranges of group sizes and shared coverage at each threshold. Increasing the coverage threshold from 1,000 to 4,000 reduces the number of distinct forecasters from 70 to 22. Within each threshold, we order groups by decreasing number of forecasters, breaking ties by decreasing number of shared subquestions. Detailed group-level statistics are provided in Table 3 in Section C.

Table 1: Overall dataset statistics by coverage threshold $N$. Each forecaster is counted once per threshold. The last two columns report the ranges of group sizes and shared-subquestion counts before the training-test split.

| Required shared subquestions ($N$) | Distinct forecasters | No. of maximal groups | Forecasters per group (min-max) | Shared subquestions per group (min-max) |
|---|---|---|---|---|
| 1,000 | 70 | 16 | 7–18 | 1,006–2,969 |
| 4,000 | 22 | 9 | 3–9 | 4,082–5,907 |

### 3.2 Experimental Protocol and Evaluation

We evaluate whether individually weaker forecasters can be aggregated to outperform the strongest individual in their comparison group. We assess accuracy and calibration using the Brier score (BS) and expected calibration error (ECE) defined in Section 2.2. In our evaluation, we divide each group's shared subquestions into approximately equal training and test sets using a fixed random seed. We use the same partition for all forecasters, aggregation rules, and pool sizes within the group. Our category-specific experiments construct groups separately within each category using the same coverage criterion, but split the data by question, keeping each question's subquestions together. We describe additional restrictions alongside the corresponding experiments.

**Group benchmark and candidate pools.** Within each group, we identify the *strongest forecaster* as the individual with the lowest test Brier score. We use this forecaster as the *group benchmark* and exclude it from every candidate aggregate. For a pool size $k \in \{2, 3, 4, 5\}$, we define a candidate pool $P \subseteq S$ as a set of $k$ distinct forecasters, each with a strictly higher test Brier score than the benchmark. We evaluate all such pools in every group with enough eligible members. Our comparison is retrospective because we use test BS to select both the benchmark and eligible forecasters: we ask whether individuals who perform worse on the evaluation sample can collectively outperform its strongest individual. For the ECE analysis, we keep the same benchmark and candidate pools.

We compare the fixed and learned rules defined in Section 2.3. Our baselines for pools of three to five forecasters are the arithmetic mean, harmonic mean, geometric mean of odds, and median. Our pairwise baselines include the arithmetic mean, harmonic mean, geometric mean of odds, and four robust rules: average-prior, heuristic-prior, precision, and prior-agnostic. We evaluate learned log-odds and linear pooling at every pool size. We fit each learned rule separately for each pool by minimizing training BS over nonnegative coefficients, then hold these coefficients fixed for both test BS and test ECE. The fixed rules require no training.

**Evaluation and reporting.** We evaluate every aggregate on its group's common test set. For each metric, we compare the aggregate with the group benchmark using two criteria. An aggregate *matches or improves* the benchmark if its score is no greater than the benchmark's, and is *within 5%* if its score is at most 1.05 times the benchmark's. The latter criterion includes all matching or improving aggregates. We apply these criteria separately to BS and ECE and report two complementary summaries for each rule and pool size:

- Group-level rates report the fraction of eligible groups containing at least one pool that matches or improves the benchmark, or lies within 5% of it. We include only groups with at least one eligible pool in the denominator.
- Pool-level rates report the corresponding fractions across all eligible pool occurrences. We count each group–pool combination once, so the same model combination appearing in multiple groups is evaluated separately on each group's test set.

We additionally report whether aggregation improves on the forecasters being combined. Specifically, we report the fraction of eligible pools whose aggregate score is no greater than that of every member of the pool. For two-forecaster pools, this is reported as *Improve on both members*. We apply these criteria separately to BS and ECE.

For the category-specific analyses in Section 4.3, we construct comparison groups within each category using the same coverage criterion. We state any additional restrictions alongside the corresponding experiments.

# 4 MAIN EXPERIMENT RESULTS

In this section, we focus on pairwise aggregation and compare learned and fixed aggregation rules. We set the coverage threshold to be $N = 1{,}000$ which gives us 1,121 eligible pairs from 16 comparison groups. We then evaluate the additional benefit of multi-model aggregation. We additionally examine whether successful pairs remain available under practical input-price and open-weight constraints in Section H. Our results show that aggregating just two individually weaker forecasters can outperform the group benchmark, where the learned linear pooling achieves the highest overall success rates in both BS and ECE.

## 4.1 PAIRWISE AGGREGATION RESULTS

We first examine whether aggregating two individually weaker forecasters improves forecast accuracy and can close the gap to the group benchmark in their comparison group. We then compare learned and fixed rules to assess the additional benefit of learning the weights. We also examine whether these gains persist when neither constituent is close to the benchmark, and whether the BS improvements are accompanied by good calibration.

**Aggregating two weaker models can outperform the strongest group benchmark.** In Table 2, we report the overall two-model BS results across the 16 comparison groups and 1,121 eligible pairs. For each aggregation rule, we report the fraction of groups and pairs that match or improve on the group benchmark, the corresponding fractions within 5% of the benchmark, and the fraction of pairs whose aggregate improves on both constituent forecasters.[2]

Table 2: Two-model BS results at $N = 1{,}000$. Cells report the group-level and pair-level rates as well as the "Improve on both members" described in Section 3.2 in the form of fractions and percentages.

| | Groups | | Eligible pairs | | |
|---|---|---|---|---|---|
| Aggregation rule | Match or improve benchmark | Within 5% of benchmark | Match or improve benchmark | Within 5% of benchmark | Improve on both members |
| Learned linear pooling | 11/16 (68.75%) | 16/16 (100.00%) | 315/1,121 (28.10%) | 730/1,121 (65.12%) | 1,073/1,121 (95.72%) |
| Learned log-odds pooling | 8/16 (50.00%) | 15/16 (93.75%) | 84/1,121 (7.49%) | 467/1,121 (41.66%) | 1,022/1,121 (91.17%) |
| Simple mean (AM) | 8/16 (50.00%) | 15/16 (93.75%) | 70/1,121 (6.24%) | 345/1,121 (30.78%) | 736/1,121 (65.66%) |
| Harmonic mean (HM) | 8/16 (50.00%) | 16/16 (100.00%) | 46/1,121 (4.10%) | 349/1,121 (31.13%) | 658/1,121 (58.70%) |
| Log-odds mean (GM) | 8/16 (50.00%) | 15/16 (93.75%) | 52/1,121 (4.64%) | 375/1,121 (33.45%) | 689/1,121 (61.46%) |

We find that the learned rules produce successful pairs across many comparison groups. Learned linear pooling identifies at least one pair that matches or improves on the group benchmark in 11 of the 16 groups (68.75%), and at least one pair within 5% of the benchmark in all 16 groups. Notably, even under a tighter 3% threshold, the best learned-linear pair is within 3% of the group benchmark in all 16 comparison groups (see Table 8 for detailed BSs.) Learned log-odds pooling matches or improves on the benchmark in 8 groups (50.00%) and comes within 5% in 15 groups (93.75%). These results show that pairs of individually weaker forecasters can perform close to, or better than, the group benchmark across many comparison groups.[3]

In Figure 1, we visualize these group-level results. For each comparison group, we report the group benchmark together with the lowest-BS pair under each learned rule, where the best pair is selected separately for linear and log-odds pooling. A best-pair bar at or above the benchmark bar indicates a successful pair. We thus observe directly from Figure 1 that learned linear pooling produces a

[2] In Table 8 of Section E, we record the detailed group benchmark and the best weak-pair aggregate under each learned rule for each group, for which we note that the aggregation weights here are all non-zeros, so the aggregation rules are actually utilizing all the forecasters.

[3] As a robustness check, we increase the shared-coverage threshold from $N = 1{,}000$ to $N = 4{,}000$ subquestions and continue to observe the same qualitative pattern of weak-to-strong aggregation; see Section D.1 for details.

successful pair in 11 groups while learned log-odds pooling in 8 groups and pairwise aggregation is more significant in G3, G6, G8, G13 and G16. The figure also shows that their relative performance may vary across groups: linear pooling succeeds where log-odds pooling does not in G9, G12, G15, and G16, while the reverse occurs in G11.

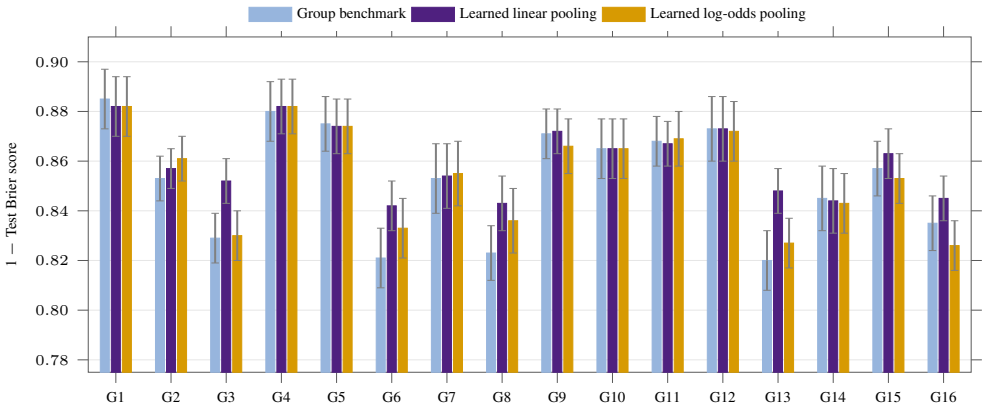



Figure 1: Best learned pair versus the group benchmark across the 16 comparison groups at $N = 1{,}000$. For each group, we plot test $1 - \text{BS}$ for the group benchmark and the lowest-BS pair under learned linear and log-odds pooling, selected separately for each rule. Higher bars ($1 - \text{BS}$) are better; groups are ordered by decreasing number of forecasters.

**Learning the weights matters.** We next examine the additional benefit of learning the aggregation coefficients by comparing each learned rule with its equal-weight counterpart. Table 2 shows that learned linear pooling matches or improves on the benchmark for 28.10% of the eligible pairs, compared with 6.24% for the simple mean. Learned log-odds pooling also improves on its equal-weight counterpart, with success rates of 7.49% and 4.64%, respectively. Learning also makes improvement over both constituent forecasters more common: learned linear and log-odds pooling improve on both members for 95.72% and 91.17% of the pairs, compared with 65.66% and 61.46% for their equal-weight counterparts.[4]

**The effect is not driven only by near-best models.** We next examine whether aggregation remains effective when neither constituent is close to the benchmark. We call a forecaster *near-best* if its individual test BS is at most 5% above that of the group benchmark, and classify a pair as *near* if it contains at least one near-best member and *no-near* otherwise.

In Table 10 (see Section E), we first report group-level results which show that successful aggregation generally does not require a near-best constituent. In particular, under learned linear pooling, 16 groups contain a pair within 5% of the benchmark, and 15 of these groups contain a no-near pair meeting this criterion, compared with 9 containing a near pair. The pair-level results also share a similar pattern mostly, with a massive no-near-pair fraction. These results further indicate that the gains from learned linear pooling do not rely on having a constituent already close to the benchmark.

**The BS gains are generally accompanied by good calibration.** We next evaluate the same aggregates using ECE as a complementary metric. We keep the BS-best group benchmark, the BS-defined candidate pairs, and the coefficients learned by minimizing training BS fixed, and evaluate their test ECE without reselecting the benchmark or refitting the learned rules. In Table 11 (see Section E), we report the corresponding ECE results across the same 16 comparison groups and 1,121 eligible pairs. In Table 11, learned linear pooling finds at least one pair that matches or improves on the group benchmark's ECE in all 16 comparison groups; 772 of the 1,121 eligible pairs (68.87%) meet this criterion. Learned log-odds pooling does so in 13 groups and for 291 pairs (25.96%), with a pair-level success rate similar to that of the log-odds mean (27.92%).

[4]Apart from results in Table 2, we have also compared each learned rule with its equal-weight counterpart on the same 1,121 pairs in Figure 3 (see Section E) and examined the robust aggregation rules in the same manner shown in this section (see Table 9 of Section E for more details).

### 4.2 Multi-Model Aggregation

We next examine whether increasing the number of individually weaker forecasters provides substantial additional gains in the aggregated forecasts. In Figure 8 (see Section G), we summarize and report aggregation performance as the pool size increases from two models to five models. Figure 8a reports the fraction of comparison groups containing at least one pool that matches or improves on the group benchmark. In particular, for learned linear pooling, this fraction increases from 68.75% (11/16 groups) with two forecasters to 75.00% (12/16) with three, and is 68.75% (11/16) with four and five. For learned log-odds pooling, the corresponding fraction increases from 50.00% (8/16) with two forecasters to 68.75% (11/16) with three, and remains at 68.75% with four and five. Thus, increasing the pool size can create additional successful groups, but most of the increase occurs when moving from two to three forecasters and does not continue as the pool grows further.

Figure 8b zooms in group G13. The best learned-linear pool reaches $0.848$ with two members and remains at approximately $0.848$ for exact pool sizes three through five. Thus, larger candidate pools do not materially raise the best observed score in this group.

### 4.3 Question-Category-Based Analysis

We next study whether the aggregation gains differ across question categories. We form comparison groups separately for Politics and Finance at the same coverage threshold $N = 1{,}000$, yielding 103 eligible pairs in 12 Politics groups and 287 pairs in 9 Finance groups. We evaluate the same learned rules and retain each category-specific group's BS-best individual as the benchmark. We also examine other question categories including Climate, Science, Sports and Entertainment. However, these categories do not include enough comparison groups for further investigation. Nevertheless, we provide the dataset statistics of 6 question categories in Table 14 of Section I.

We first observe that the aggregation gains can differ substantially across question categories from Table 15 (see Section I). Results from Table 15 show that learned linear pooling is particularly effective in the Finance comparisons, both in how widely the gains appear across groups and how frequently they appear across eligible pairs (In Section I, we provide detailed group-level statistics in Table 16 while the group benchmark and the best weak-pair aggregate under each learned rule for each group are demonstrated in Table 17).

Figure 10 (see Section I) provides a group-level view of these differences. One possible explanation for why the learned linear pooling performs so well for the Finance category is the following. The Finance benchmarks have substantially higher BS than the Politics benchmarks: across the 9 Finance groups, benchmark BS ranges from 0.210 to 0.247, whereas across the 12 Politics groups it ranges from 0.075 to 0.111. The strongest group benchmark therefore performs worse in the Finance category compared to those in the Politics category, which leaves more room for aggregation to close the gap. We also observe a similar pattern for calibration, as the BS-selected benchmarks generally have higher ECE in Finance than in Politics (see Figure 11 in Section I). This may partly explain why learned linear pooling performs especially well in Finance, including its 97.91% pair-level ECE success rate.[5]

## 5 Conclusions

In this paper, we study weak-to-strong forecast aggregation for LLMs: whether individually weaker forecasters can be aggregated to match or outperform a stronger forecaster. Using ForecastBench, we find substantial evidence of such improvement. Learned linear pooling identifies a weaker pair that matches or outperforms the strongest individual in 11 of 16 comparison groups and comes within 5% of its Brier score in all 16 groups. These gains do not rely on having a near-best constituent and are generally accompanied by good calibration. Our additional analyses show that learning the aggregation weights matters, while adding more forecasters does not consistently improve performance. The weak-to-strong pattern also persists under a higher shared-coverage threshold and remains available under practical cost and access constraints.

[5]Because the category-specific groups differ in both forecasters and shared questions, however, we view this as a possible explanation rather than a causal effect of question category.

## AI DISCLOSURE

In this work, we have not used generative AI tools for designing or providing feedback on research methodology or experiments, implementing methods, assisting with translation, cleaning and reformatting the dataset, supporting qualitative and thematic data analysis, interpreting results. The AI is used for proofreading and improving the readability. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REFERENCES


Mahdi Abolghasemi, Odkhishig Ganbold, and Kristian Rotaru. Humans vs. large language models: Judgmental forecasting in an era of advanced AI. *International Journal of Forecasting*, 41(2): 631–648, 2025.

Matthew Aitchison, Scott Jeen, Toby Shevlane, and Ben Day. Diversity is the strength of the AI crowd. *arXiv preprint arXiv:2606.29661*, 2026.

Rohan Alur, Bradly Stadie, Daniel Kang, Ryan Chen, Matt McManus, Michael Rickert, Tyler Lee, Michael Federici, Richard Zhu, Dennis Fogerty, Hayley Williamson, Nina Lozinski, Aaron Linsky, and Jasjeet Sekhon. AIA Forecaster: Technical report. *arXiv preprint arXiv:2511.07678*, 2025.

Anthropic. Introducing Claude 2.1, 2023. https://www.anthropic.com/news/claude-2-1.

Anthropic. Introducing the next generation of Claude, 2024a. https://www.anthropic.com/news/claude-3-family.

Anthropic. Introducing computer use, a new Claude 3.5 Sonnet, and Claude 3.5 Haiku, 2024b. https://www.anthropic.com/news/3-5-models-and-computer-use.

Anthropic. Introducing Claude 3.5 Sonnet, 2024c. https://www.anthropic.com/news/claude-3-5-sonnet.

Anthropic. Claude 3.7 Sonnet and Claude Code, 2025a. https://www.anthropic.com/news/claude-3-7-sonnet.

Anthropic. Introducing Claude 4, 2025b. https://www.anthropic.com/news/claude-4.

Anthropic. Introducing Claude Haiku 4.5, 2025c. https://www.anthropic.com/news/claude-haiku-4-5.

Anthropic. Claude Opus 4.1, 2025d. https://www.anthropic.com/news/claude-opus-4-1.

Anthropic. Introducing Claude Sonnet 4.5, 2025e. https://www.anthropic.com/news/claude-sonnet-4-5.

Anthropic. Introducing Claude Opus 4.6, 2026a. https://www.anthropic.com/news/claude-opus-4-6.

Anthropic. Introducing Claude Sonnet 4.6, 2026b. https://www.anthropic.com/news/claude-sonnet-4-6.

Itai Arieli, Yakov Babichenko, and Rann Smorodinsky. Robust forecast aggregation. *Proceedings of the National Academy of Sciences*, 115(52):E12135–E12143, 2018.

Jon Armstrong (ed.). *Principles of forecasting: a handbook for researchers and practitioners*, volume 30. Springer, 2001.

Jonathan Baron, Barbara Mellers, Philip Tetlock, Eric Stone, and Lyle Ungar. Two reasons to make aggregated probability forecasts more extreme. *Decision Analysis*, 11(2):133–145, 2014.

John Bates and Clive Granger. The combination of forecasts. *Journal of the Operational Research Society*, 20(4):451–468, 1969.

Federico Bianchi, Junlin Wang, Zain Hasan, Shang Zhu, Roy Yuan, Clémentine Fourrier, and James Zou. Back to the future: Evaluating AI agents on predicting future events. TogetherAI (FutureBench), July 17, 2025. https://www.together.ai/blog/futurebench.

Robert Bordley. A multiplicative formula for aggregating probability assessments. *Management Science*, 28(10):1137–1148, 1982.

Glenn Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3, 1950.

P. Bullen. Quasi-arithmetic means. In *Handbook of Means and Their Inequalities*, pp. 266–320. Springer Netherlands, 2003. doi: 10.1007/978-94-017-0399-4_4.

Zhi Chen, Cheng Peng, and Wei Tang. Prior-agnostic robust forecast aggregation. In *ACM Conference on Economics and Computation*, 2026.

Robert Clemen. Combining forecasts: A review and annotated bibliography. *International Journal of Forecasting*, 5(4):559–583, 1989.

Gheorghe Comanici et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. https://arxiv.org/abs/2507.06261.

Hui Dai, Ryan Teehan, and Mengye Ren. Are LLMs prescient? A continuous evaluation using daily news as the oracle. In *International Conference on Machine Learning*, pp. 12179–12213, 2025.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *International Conference on Machine Learning*, pp. 10148–10167, 2024.

Henrique De Oliveira, Yuhta Ishii, and Xiao Lin. Robust merging of information. In *ACM Conference on Economics and Computation*, pp. 18–23, 2021.

DeepSeek. DeepSeek-V3.1 release, 2025. https://www.deepseek.com/en/news/deepseek-v3-1/.

Samuel Dooley, Gurnoor Khurana, Chirag Mohapatra, Siddartha Naidu, and Colin White. ForecastPFN: Synthetically-trained zero-shot forecasting. In *Advances in Neural Information Processing Systems*, pp. 2403–2426, 2023.

Igor Douven. Wisdom of LLM crowds: Aggregation and contamination in language model ensembles. *ACM Transactions on Social Computing*, 2026. doi: 10.1145/3839337.

Yilun Du, Shuang Li, Antonio Torralba, Joshua Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *International Conference on Machine Learning*, pp. 11733–11763, 2024.

Philip Ernst, Robin Pemantle, Ville Satopää, and Lyle Ungar. Bayesian aggregation of two forecasts in the partial information framework. *Statistics & Probability Letters*, 119:170–180, 2016.

Lukas Fluri, Daniel Paleka, and Florian Tramèr. Evaluating superhuman models with consistency checks. In *IEEE Conference on Secure and Trustworthy Machine Learning*, pp. 194–232, 2024.

Rafael Frongillo, Yiling Chen, and Ian Kash. Elicitation for aggregation. In *AAAI Conference on Artificial Intelligence*, 2015.

Rafael Frongillo, Mary Monroe, Eric Neyman, and Bo Waggoner. Robust forecast aggregation via additional queries. *arXiv preprint arXiv:2512.05271*, 2025.

Christian Genest. A characterization theorem for externally Bayesian groups. *The Annals of Statistics*, pp. 1100–1105, 1984.

Christian Genest and James Zidek. Combining probability distributions: A critique and an annotated bibliography. *Statistical Science*, 1(1):114–135, 1986.

Christian Genest, Samaradasa Weerahandi, and James Zidek. Aggregating opinions through logarithmic pooling. *Theory and Decision*, 17(1):61, 1984.

Google. Gemini 1.5 Pro updates, 1.5 Flash debut and two new Gemma models, 2024a. https://blog.google/innovation-and-ai/technology/developers-tools/gemini-gemma-developer-updates-may-2024/.

Google. Our next-generation model: Gemini 1.5, 2024b. https://blog.google/innovation-and-ai/products/google-gemini-next-generation-model-february-2024/.

Google. Gemini 2.0 model updates: Flash, Flash-Lite and Pro, 2025a. https://blog.google/innovation-and-ai/models-and-research/google-deepmind/gemini-model-updates-february-2025/.

Google. Gemini model thinking updates, 2025b. https://blog.google/innovation-and-ai/models-and-research/google-deepmind/gemini-model-thinking-updates-march-2025/.

Google. We are expanding our Gemini 2.5 family of models, 2025c. https://blog.google/products-and-platforms/products/gemini/gemini-2-5-model-family-expands/.

Google. A new era of intelligence with Gemini 3, 2025d. https://blog.google/products-and-platforms/products/gemini/gemini-3/.

Google. Gemini 3 Flash: frontier intelligence built for speed, 2025e. https://blog.google/products-and-platforms/products/gemini/gemini-3-flash/.

Google. Gemini API release notes, 2025f. https://ai.google.dev/gemini-api/docs/changelog.

Google. Gemini 3.1 Pro: A smarter model for your most complex tasks, 2026. https://blog.google/innovation-and-ai/models-and-research/gemini-models/gemini-3-1-pro/.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. MOMENT: A family of open time-series foundation models. In *International Conference on Machine Learning*, pp. 16115–16152, 2024.

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew Wilson. Large language models are zero-shot time series forecasters. In *Advances in Neural Information Processing Systems*, pp. 19622–19635, 2023.

Yong Guan, Hao Peng, Xiaozhi Wang, Lei Hou, and Juanzi Li. OpenEP: Open-ended future event prediction. *ACM Transactions on Information Systems*, 44(4):1–27, 2026.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pp. 1321–1330, 2017.

Daya Guo, Dejian Yang, Haowei Zhang, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning, 2025a. https://arxiv.org/abs/2501.12948.

Yongkang Guo and Yuqing Kong. Robust aggregation with adversarial experts. In *ACM Web Conference*, pp. 3952–3967, 2025.

Yongkang Guo, Jason Hartline, Zhihuan Huang, Yuqing Kong, Anant Shah, and Fang-Yi Yu. Algorithmic robust forecast aggregation. In *ACM Conference on Economics and Computation*, pp. 1110–1129, 2025b.

Danny Halawi, Fred Zhang, Yueh-Han Chen, and Jacob Steinhardt. Approaching human-level forecasting with language models. In *Advances in Neural Information Processing Systems*, 2024.

Charles Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1):5–10, 2004.

Wenyang Hu, Gregory Lau, Diwen Liu, Jizhuo Chen, See-Kiong Ng, and Bryan Low. Dipper: Diversity in prompts for producing large language model ensembles in reasoning tasks. In *Conference on Empirical Methods in Natural Language Processing*, pp. 35558–35572, 2025.

Younes Jeddi, Jose Segovia-Martin, and Emile Servan-Schreiber. Crowdsourced versus large language models forecasting: Evidence for the accuracy–correlation effect. *Philosophical Transactions B*, 381(1948):20240456, 2026.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-LLM: Time series forecasting by reprogramming large language models. In *International Conference on Learning Representations*, pp. 23857–23880, 2024.

Woojeong Jin, Rahul Khanna, Suji Kim, Dong-Ho Lee, Fred Morstatter, Aram Galstyan, and Xiang Ren. ForecastQA: A question answering challenge for event forecasting with temporal text data. In *Annual Meeting of the Association for Computational Linguistics and International Joint Conference on Natural Language Processing*, pp. 4636–4650, 2021.

Ezra Karger, Houtan Bastani, Yueh-Han Chen, Zachary Jacobs, Danny Halawi, Fred Zhang, and Philip Tetlock. ForecastBench: A dynamic benchmark of AI forecasting capabilities. In *International Conference on Learning Representations*, 2025.

Kimi Team. Kimi K2: Open agentic intelligence, 2025. https://arxiv.org/abs/2507.20534.

Yuqing Kong, Shu Wang, and Ying Wang. The surprising benefits of base rate neglect in robust aggregation. In *ACM Conference on Economics and Computation*, pp. 1131, 2024.

Gilat Levy and Ronny Razin. A maximum likelihood approach to combining forecasts. *Theoretical Economics*, 16(1):49–71, 2021.

Gilat Levy and Ronny Razin. Combining forecasts in the presence of ambiguity over correlation structures. *Journal of Economic Theory*, 199:105075, 2022.

Aixin Liu, Bei Feng, Bing Xue, et al. DeepSeek-V3 technical report, 2024. https://arxiv.org/abs/2412.19437.

Meta AI. Meta and Microsoft introduce the next generation of Llama, 2023. https://ai.meta.com/blog/llama-2/.

Meta AI. Introducing Llama 3.1, 2024a. https://ai.meta.com/blog/meta-llama-3-1/.

Meta AI. Meta AI's Llama 3, 2024b. https://ai.meta.com/blog/meta-llama-3/.

Meta AI. Llama 3.2: Revolutionizing edge AI and vision with open, customizable models, 2024c. https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/.

Meta AI. Llama 3.3 70B Instruct model card, 2024d. https://huggingface.co/meta-llama/Llama-3.3-70B-Instruct.

Meta AI. The Llama 4 herd: The beginning of a new era of natively multimodal AI innovation, 2025. https://ai.meta.com/blog/llama-4-multimodal-intelligence/.

Mistral AI. Mixtral of experts, 2023. https://mistral.ai/news/mixtral-of-experts/.

Mistral AI. Large enough: Mistral Large 2, 2024a. https://mistral.ai/news/mistral-large-2407/.

Mistral AI. Cheaper, better, faster, stronger, 2024b. https://mistral.ai/news/mixtral-8x22b/.

Mistral AI. Magistral Medium 1.0, 2025. https://docs.mistral.ai/models/magistral-medium-1-0-25-06.

Moonshot AI. Kimi K2 Thinking model card, 2025. https://huggingface.co/moonshotai/Kimi-K2-Thinking.

Kevin Murphy. Agentic forecasting using sequential Bayesian updating of linguistic beliefs. *arXiv preprint arXiv:2604.18576*, 2026.

Eric Neyman and Tim Roughgarden. Are you smarter than a random expert? The robust aggregation of substitutable signals. In *ACM Conference on Economics and Computation*, pp. 990–1012, 2022.

Yuqi Nie, Nam Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

OpenAI. Introducing ChatGPT and Whisper APIs, 2023a. https://openai.com/index/introducing-chatgpt-and-whisper-apis/.

OpenAI. GPT-4, 2023b. https://openai.com/index/gpt-4-research/.

OpenAI. New models and developer products announced at DevDay, 2023c. https://openai.com/index/new-models-and-developer-products-announced-at-devday/.

OpenAI. Hello GPT-4o, 2024. https://openai.com/index/hello-gpt-4o/.

OpenAI. Introducing GPT-4.1 in the API, 2025a. https://openai.com/index/gpt-4-1/.

OpenAI. Introducing GPT-4.5, 2025b. https://openai.com/index/introducing-gpt-4-5/.

OpenAI. Introducing GPT-5.1 for developers, 2025c. https://openai.com/index/gpt-5-1-for-developers/.

OpenAI. Introducing GPT-5.2, 2025d. https://openai.com/index/introducing-gpt-5-2/.

OpenAI. Introducing GPT-5 for developers, 2025e. https://openai.com/index/introducing-gpt-5-for-developers/.

OpenAI. OpenAI o3-mini, 2025f. https://openai.com/index/openai-o3-mini/.

OpenAI. Introducing OpenAI o3 and o4-mini, 2025g. https://openai.com/index/introducing-o3-and-o4-mini/.

OpenAI. Introducing GPT-5.4, 2026a. https://openai.com/index/introducing-gpt-5-4/.

OpenAI. Introducing GPT-5.4 mini and nano, 2026b. https://openai.com/index/introducing-gpt-5-4-mini-and-nano/.

Daniel Paleka, Abhimanyu Sudhir, Alejandro Alvarez, Vineeth Bhat, Adam Shen, Evan Wang, and Florian Tramèr. Consistency checks for language model forecasters. In *International Conference on Learning Representations*, 2025.

Long Phan, Andrew Zeng, Mantas Mazeika, Adam Khoja, and Dan Hendrycks. Superhuman automated forecasting. Center for AI Safety Blog, September 8, 2024. https://safe.ai/blog/forecasting.

Sarah Pratt, Seth Blumberg, Pietro Carolino, and Meredith Morris. Can language models use forecasting strategies? *arXiv preprint arXiv:2406.04446*, 2024.

Qwen Team. QwQ: Reflect deeply on the boundaries of the unknown, 2024a. https://qwenlm.github.io/blog/qwq-32b-preview/.

Qwen Team. Qwen1.5-110B: The first 100B+ model of the Qwen1.5 series, 2024b. https://qwenlm.github.io/blog/qwen1.5-110b/.

Qwen Team. Qwen2.5: A party of foundation models, 2024c. https://qwenlm.github.io/blog/qwen2.5/.

Qwen Team. Qwen3-235B-A22B-Thinking-2507 model card, 2025. https://huggingface.co/Qwen/Qwen3-235B-A22B-Thinking-2507.

Roopesh Ranjan and Tilmann Gneiting. Combining probability forecasts. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 72(1):71–91, 2010.

Kashif Rasul, Arjun Ashok, Andrew Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, and Irina Rish. Lag-Llama: Towards foundation models for time series forecasting. In *R0-FoMo: Robustness of Few-shot and Zero-shot Learning in Large Foundation Models*, 2023.

Ville Satopää, Jonathan Baron, Dean Foster, Barbara Mellers, Philip Tetlock, and Lyle Ungar. Combining multiple probability predictions using a simple logit model. *International Journal of Forecasting*, 30(2):344–356, 2014.

Ville Satopää, Robin Pemantle, and Lyle Ungar. Modeling probability forecasts via information diversity. *Journal of the American Statistical Association*, 111(516):1623–1633, 2016.

Robert Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

Philipp Schoenegger and Peter Park. Large language model prediction capabilities: Evidence from a real-world forecasting tournament. *arXiv preprint arXiv:2310.13014*, 2023.

Philipp Schoenegger, Indre Tuminauskaite, Peter Park, Rafael Bastos, and Philip Tetlock. Wisdom of the silicon crowd: LLM ensemble prediction capabilities rival human crowd accuracy. *Science Advances*, 10(45):eadp1528, 2024. doi: 10.1126/sciadv.adp1528.

Mervyn Stone. The opinion pool. *The Annals of Mathematical Statistics*, pp. 1339–1342, 1961.

Wei Tang and Hanrui Zhang. Algorithmic expert aggregation. *arXiv preprint arXiv:2607.08744*, 2026.

Demetre Tsiklauri. Evaluating large language model ensembles as probabilistic forecasters: An empirical study on live real-world events. *Available at SSRN 6709818*, 2026.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.

Zhen Wang, Xi Zhou, Yating Yang, Bo Ma, Lei Wang, Rui Dong, and Azmat Anwar. OpenForecast: A large-scale open-ended event forecasting dataset. In *International Conference on Computational Linguistics*, pp. 5273–5294, 2025.

Jack Wildman, Nikos Bosse, Daniel Hnyk, Peter Mühlbacher, Finn Hambly, Jon Evans, Dan Schwarz, and Lawrence Phillips. Bench to the future: A pastcasting benchmark for forecasting agents. *arXiv preprint arXiv:2506.21558*, 2025.

Peter Winters. Forecasting sales by exponentially weighted moving averages. *Management Science*, 6(3):324–342, 1960.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *International Conference on Machine Learning*, pp. 53140–53164, 2024.

xAI. Announcing the xAI API, 2024. https://x.ai/news/api.

xAI. Grok 4.1 Fast and agent tools API, 2025a. https://x.ai/news/grok-4-1-fast.

xAI. Grok 4, 2025b. https://x.ai/news/grok-4.

xAI. Grok 4 Fast, 2025c. https://x.ai/news/grok-4-fast.

Qi Yan, Raihan Seraj, Jiawei He, Lili Meng, and Tristan Sylvain. AutoCast++: Enhancing world event prediction with zero-shot ranking-based context retrieval. In *International Conference on Learning Representations*, pp. 34523–34539, 2024.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report, 2025. https://arxiv.org/abs/2505.09388.

Qingchuan Yang, Simon Mahns, Sida Li, Anri Gu, Jibang Wu, and Haifeng Xu. LLM-as-a-Prophet: Understanding predictive intelligence with Prophet Arena. In *International Conference on Learning Representations*, 2026.

Z.ai. GLM-4.5: Reasoning, coding, and agentic abilities, 2025. https://z.ai/blog/glm-4.5.

Zhiyuan Zeng, Jiashuo Liu, Siyuan Chen, Tianci He, Yali Liao, Yixiao Tian, Jinpeng Wang, Zaiyuan Wang, Yang Yang, Lingyue Yin, Mingren Yin, Zhenwei Zhu, Tianle Cai, Zehui Chen, Jiecao Chen, Yantao Du, Xiang Gao, Jiacheng Guo, Liang Hu, Jianpeng Jiao, Xiangsheng Li, Jingkai Liu, Shuang Ni, Zhoufutu Wen, Ge Zhang, Kaiyuan Zhang, Xin Zhou, Wenhao Huang, Jose Blanchet, Xipeng Qiu, and Mengdi Wang. FutureX: An advanced live benchmark for LLM agents in future prediction. In *International Conference on Learning Representations*, pp. 19522–19563, 2026.

Zhihan Zhang, Yixin Cao, Chenchen Ye, Yunshan Ma, Lizi Liao, and Tat-Seng Chua. Analyzing temporal complex events with large language models? A benchmark towards temporal, long context understanding. In *Annual Meeting of the Association for Computational Linguistics*, pp. 1588–1606, 2024.

Chenghao Zhu, Nuo Chen, Yufei Gao, Yunyi Zhang, Prayag Tiwari, and Benyou Wang. Is your LLM outdated? A deep look at temporal generalization. In *Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 7433–7457, 2025.

Andy Zou, Tristan Xiao, Ryan Jia, Joe Kwon, Mantas Mazeika, Richard Li, Dawn Song, Jacob Steinhardt, Owain Evans, and Dan Hendrycks. Forecasting future world events with neural networks. In *Advances in Neural Information Processing Systems*, pp. 27293–27305, 2022.

## A Further Related Work

**LLMs' forecasting and benchmarking.** Recent work has explored the use of large language models for automated forecasting of real-world events (Jin et al., 2021; Zou et al., 2022; Schoenegger & Park, 2023; Fluri et al., 2024; Halawi et al., 2024; Phan et al., 2024; Pratt et al., 2024; Yan et al., 2024; Abolghasemi et al., 2025). Early studies generally find that off-the-shelf LLMs lag strong human forecasters, motivating methods that improve forecasting through better prompting, information retrieval, reasoning, fine-tuning, and calibration. In particular, Halawi et al. (2024) develop a retrieval-augmented forecasting system that combines information retrieval, structured reasoning, and multiple model predictions, and show that the resulting system approaches the performance of competitive human forecasting crowds. Subsequent work further studies specific ingredients of LLM forecasting, including forecasting strategies and reasoning (Pratt et al., 2024), retrieval of temporally appropriate information (Yan et al., 2024), temporal generalization and the degradation of

forecasting accuracy over time (Dai et al., 2025; Zhu et al., 2025), and logical consistency of probabilistic forecasts (Paleka et al., 2025). Yang et al. (2026) further study how internalized knowledge, information sources, retrieval, and the ability to integrate available evidence affect LLMs' predictive performance.

A related but distinct literature applies LLMs and transformer-based models to statistical time-series forecasting (Dooley et al., 2023; Gruver et al., 2023; Nie et al., 2023; Rasul et al., 2023; Das et al., 2024; Goswami et al., 2024; Jin et al., 2024; Woo et al., 2024). Our focus, as in the event-forecasting literature above, is instead on probabilistic judgmental forecasting of real-world events, for which relevant evidence may come from heterogeneous textual and contextual sources rather than a well-defined historical time series.

Alongside advances in forecasting methods, a growing literature has developed increasingly challenging benchmarks for evaluating LLMs' predictive capabilities. ForecastQA (Jin et al., 2021) is an early event-forecasting dataset based on future events described in news articles, while Autocast (Zou et al., 2022) incorporates questions from forecasting competitions and prediction platforms. Since then, benchmark construction has expanded along several dimensions, including extracting and representing forecastable events from news and temporal information (Zhang et al., 2024; Wang et al., 2025), developing open-ended future-event prediction tasks (Guan et al., 2026), and constructing forecasting benchmarks that place greater emphasis on temporal generalization and realistic information environments (Wildman et al., 2025). More recently, the literature has shifted toward dynamic and live evaluation, in part to reduce benchmark contamination and avoid relying on uncertain model knowledge cutoffs (Bianchi et al., 2025; Karger et al., 2025; Zeng et al., 2026).

ForecastBench (Karger et al., 2025) continuously collects unresolved forecasting questions from prediction markets, forecasting platforms, and real-world datasets, and evaluates LLMs and human forecasters as these questions resolve. FutureBench (Bianchi et al., 2025) and FutureX (Zeng et al., 2026) further extend live evaluation toward agentic forecasting. In particular, FutureX evaluates LLMs equipped with reasoning, search, and deep-research capabilities on a broad range of frequently updated events drawn from diverse real-world sources. Prophet Arena (Yang et al., 2026) complements these benchmarks by continuously evaluating probabilistic forecasts on live prediction-market events under a multi-horizon and modularized forecasting pipeline, and by assessing models along multiple dimensions, including forecasting loss, calibration, and economic value. Together, these benchmarks provide increasingly rich and contamination-resistant environments for comparing LLM forecasters. Our work builds on this literature using forecasts collected by ForecastBench, but shifts attention from evaluating individual forecasters to understanding what can be gained by combining forecasts from multiple LLMs.

**Forecast aggregation.** Recent work has explored how probabilistic forecasts produced by large language models can be aggregated and how model diversity affects the performance of LLM crowds (Halawi et al., 2024; Schoenegger et al., 2024; Alur et al., 2025; Karger et al., 2025; Aitchison et al., 2026; Douven, 2026; Tsiklauri, 2026). Early studies often incorporate aggregation as one component of a broader forecasting system or evaluate relatively simple pooling rules. Halawi et al. (2024) develop a retrieval-augmented forecasting system that searches for relevant information, generates multiple forecasts, and aggregates them as part of an end-to-end forecasting pipeline. Schoenegger et al. (2024) aggregate probabilistic forecasts from twelve LLMs using the median and show that the resulting LLM crowd achieves accuracy comparable to a human forecasting crowd. Karger et al. (2025) construct LLM ensemble baselines from nine forecasts per question, obtained from three LLMs under three prompting strategies, and compare the median, geometric mean, and geometric mean of log odds. More recently, Alur et al. (2025) develop the AIA Forecaster, in which a supervisor agent reconciles disparate forecasts for the same event as part of a system that combines agentic search, forecast reconciliation, and statistical calibration.

More recent work examines more directly which LLM forecasts should be combined and why aggregation can help. Aitchison et al. (2026) study model selection for forecasting ensembles under a fixed sampling budget and show that individual forecasting accuracy alone does not determine ensemble performance: forecasts from frontier models can be highly correlated, while somewhat weaker but more diverse models can contribute substantial complementary information. Relatedly, Jeddi et al. (2026) document an accuracy–correlation effect between LLM and human forecasts, showing that more accurate LLM forecasts also tend to be more correlated with human forecast aggregates. Douven (2026) compares classical and learned aggregation methods using probabilistic

forecasts from fifteen LLMs and finds that learned logistic-regression and neural-network aggregators outperform both individual models and classical pooling methods; their symbolic-regression analysis further identifies disagreement across model predictions as an important source of the aggregation gain. Together, these findings suggest that the value of an LLM crowd depends not only on the forecasting accuracy of its constituent models, but also on the dependence and complementarity among their predictions. Our work builds on this literature but focuses on a more stringent question: whether two LLMs that are each individually weaker than a third, stronger model can nevertheless be aggregated to match or outperform that stronger model.

More broadly, forecast aggregation has a long history in statistics, economics, and decision theory (Stone, 1961; Bates & Granger, 1969; Genest & Zidek, 1986; Clemen, 1989). For probability forecasts, the literature studies linear, logarithmic, multiplicative, and transformed pooling methods (Bordley, 1982; Genest, 1984; Genest et al., 1984; Ranjan & Gneiting, 2010; Baron et al., 2014; Satopää et al., 2014). Ranjan & Gneiting (2010) study linear pooling and show why recalibration can be necessary even when the component forecasts are calibrated. Satopää et al. (2014) develop a simple logit-based model in which a single parameter controls how strongly the aggregate is extremized, while Baron et al. (2014) identify probability-scale compression and the aggregation of partially independent information as two reasons why aggregated forecasts may benefit from extremization. A related information-based literature explicitly models the information underlying individual forecasts and how information overlap affects aggregation (Ernst et al., 2016; Satopää et al., 2016; Levy & Razin, 2021). In particular, Satopää et al. (2016) model partially overlapping information across forecasters and relate the appropriate degree of extremization to informational diversity. Related work also studies what additional information must be elicited from forecasters to permit effective aggregation (Frongillo et al., 2015). These approaches motivate our comparison between learned linear pooling and learned log-odds pooling.

A complementary theoretical literature studies robust forecast aggregation when the aggregator lacks full knowledge of the information structure or the dependence across information sources (Arieli et al., 2018; De Oliveira et al., 2021; Levy & Razin, 2022; Neyman & Roughgarden, 2022; Kong et al., 2024; Frongillo et al., 2025; Guo et al., 2025b; Guo & Kong, 2025; Chen et al., 2026; Tang & Zhang, 2026). Our experiments complement this literature by evaluating both robust and data-driven aggregation rules on probabilistic forecasts produced by heterogeneous LLMs.

## B Additional Aggregation Rules

We also draw on the robust forecast aggregation literature (Arieli et al., 2018; Kong et al., 2024; Guo et al., 2025b; Chen et al., 2026), which studies how to combine forecasts when the forecasters' underlying information structures are unknown to the aggregator. We consider four fixed rules from this literature as baselines: two prior plug-in rules, a precision rule, and a prior-agnostic robust rule.

**Prior plug-in aggregators.** Let $\mu \in [0,1]$ denote the common prior probability of a Yes outcome. Following Arieli et al. (2018), given two forecasts $p^1, p^2$, we define

$$g(\mu; p^1, p^2) = \frac{p^1 p^2 (1-\mu)}{p^1 p^2 (1-\mu) + (1-p^1)(1-p^2)\mu}.$$

The *average-prior* (AP) rule uses $\widehat{\mu}_{\mathrm{AP}} = (p^1 + p^2)/2$, while the *heuristic-prior* (HP) rule uses

$$\widehat{\mu}_{\mathrm{HP}} = 0.49(p^1 + p^2) + 0.02\,\mathbb{1}\{p^1 + p^2 > 1\}.$$

The corresponding aggregation rules are $f_{\mathrm{AP}}(p^1, p^2) = g(\widehat{\mu}_{\mathrm{AP}}; p^1, p^2)$ and $f_{\mathrm{HP}}(p^1, p^2) = g(\widehat{\mu}_{\mathrm{HP}}; p^1, p^2)$.

**Precision aggregator.** We also use the precision aggregator of Arieli et al. (2018), which gives more weight to forecasts closer to 0 or 1. For $p \in (0,1)$, define $\phi(p) = 1/[p(1-p)]$. The rule is

$$f_{\mathrm{pre}}(p^1, p^2) = \begin{cases} \dfrac{\phi(p^1)p^1 + \phi(p^2)p^2}{\phi(p^1) + \phi(p^2)}, & |p^1 - p^2| \le 0.4, \\ \dfrac{\sqrt{\phi(p^1)}\,p^1 + \sqrt{\phi(p^2)}\,p^2}{\sqrt{\phi(p^1)} + \sqrt{\phi(p^2)}}, & |p^1 - p^2| > 0.4. \end{cases}$$

**Prior-agnostic robust aggregator.** Finally, we use the prior-agnostic robust rule of Chen et al. (2026), which combines the forecasts directly in log-odds space: $f_\alpha(p^1, p^2) = \frac{1}{1+\left(\frac{1-p^1}{p^1}\right)^\alpha \left(\frac{1-p^2}{p^2}\right)^\alpha}$. We refer to this rule as $\alpha$-log-odds. In our experiments, we set $\alpha = 0.56$, following Chen et al. (2026).

## C Additional Experimental Details and Group Statistics

In the following Table 3, we report the size, subquestion number, and model configuration components of the groups we examined in our main experiment.

Table 3: Complete membership of the size-sorted overall groups.

| Index | Size | Subquestions | Models |
|---|---|---|---|
| **Overall, threshold 1,000** | | | |
| 1 | 18 | 1,065 | Claude-3-7-Sonnet; Claude-Haiku-4-5; Claude-Opus-4-1; Claude-Sonnet-4-5; DeepSeek-V3.1; GPT-4.1; GPT-5-Mini; GPT-5-Nano; GPT-5.1; Gemini-2.5-Pro; Gemini-3-Pro-Preview; Grok-4; Grok-4-1-Fast-Non-Reasoning; Grok-4-1-Fast-Reasoning; Grok-4-Fast-Non-Reasoning; Grok-4-Fast-Reasoning; Qwen3-235B-A22B-Fp8-Tput; Qwen3-235B-A22B-Thinking |
| 2 | 16 | 2,969 | Claude-3-5-Sonnet; Claude-3-7-Sonnet; Claude-3-Haiku; Claude-3-Opus; GPT-4-Turbo; GPT-4.1; GPT-4o; Llama-3.2-3B-Instruct-Turbo; Llama-3.3-70B-Instruct-Turbo; Llama-4-Maverick-17B-128E-Instruct-FP8; Llama-4-Scout-17B-16E-Instruct; Meta-Llama-3.1-405B-Instruct-Turbo; Mistral-Large; O4-Mini; Qwen2.5-72B-Instruct-Turbo; Qwen3-235B-A22B-Fp8-Tput |
| 3 | 16 | 1,902 | Claude-2.1; Claude-3-5-Sonnet; Claude-3-Haiku; Claude-3-Opus; GPT-3.5-Turbo; GPT-4; GPT-4-Turbo; GPT-4o; Gemini-1.5-Flash; Gemini-1.5-Pro; Llama-2-70b-Chat-Hf; Llama-3-70b-Chat-Hf; Llama-3-8b-Chat-Hf; Mixtral-8x22B-Instruct-V0.1; Mixtral-8x7B-Instruct-V0.1; Qwen1.5-110B-Chat |
| 4 | 16 | 1,257 | Claude-3-7-Sonnet; Claude-Haiku-4-5; Claude-Opus-4-1; Claude-Sonnet-4-5; DeepSeek-V3.1; GPT-4.1; GPT-5-Mini; GPT-5-Nano; Gemini-2.5-Flash; Gemini-2.5-Pro; Grok-4; Grok-4-Fast-Non-Reasoning; Grok-4-Fast-Reasoning; O3; Qwen3-235B-A22B-Fp8-Tput; Qwen3-235B-A22B-Thinking |
| 5 | 16 | 1,137 | Claude-Haiku-4-5; Claude-Opus-4-1; Claude-Sonnet-4-5; DeepSeek-V3.1; GPT-4.1; GPT-5-Mini; GPT-5-Nano; GPT-5.1; Gemini-2.5-Pro; Gemini-3-Pro-Preview; Grok-4-1-Fast-Non-Reasoning; Grok-4-1-Fast-Reasoning; Grok-4-Fast-Non-Reasoning; Grok-4-Fast-Reasoning; Kimi-K2-Thinking; Qwen3-235B-A22B-Thinking |
| 6 | 15 | 1,512 | Claude-3-5-Sonnet; Claude-3-7-Sonnet; DeepSeek-R1; DeepSeek-V3; GPT-4.1; GPT-4o; Gemini-2.5-Flash-Preview; Gemini-2.5-Pro-Preview; Llama-3.3-70B-Instruct-Turbo; Llama-4-Maverick-17B-128E-Instruct-FP8; Llama-4-Scout-17B-16E-Instruct; Mistral-Large; O3; O3-Mini; QwQ-32B-Preview |
| 7 | 15 | 1,006 | Claude-Haiku-4-5; Claude-Opus-4-6; Claude-Sonnet-4-5; Claude-Sonnet-4-6; DeepSeek-V3.1; GPT-4.1; GPT-5-Mini; GPT-5-Nano; GPT-5.1; GPT-5.2; Gemini-2.5-Pro; Gemini-3-Flash-Preview; Gemini-3.1-Pro-Preview; Grok-4-1-Fast-Non-Reasoning; Grok-4-1-Fast-Reasoning |
| 8 | 14 | 1,397 | Claude-3-5-Sonnet; Claude-3-7-Sonnet; DeepSeek-R1; DeepSeek-V3; GPT-4o; Gemini-2.0-Flash-Lite-001; Gemini-2.5-Pro-Exp; Grok-beta; Llama-3.3-70B-Instruct-Turbo; Llama-4-Maverick-17B-128E-Instruct-FP8; Llama-4-Scout-17B-16E-Instruct; Mistral-Large; O3-Mini; QwQ-32B-Preview |
| 9 | 13 | 1,744 | Claude-3-5-Sonnet; Claude-Opus-4-1; Claude-Sonnet-4; GLM-4.5-Air-FP8; GPT-4.1; GPT-5; GPT-5-Mini; GPT-5-Nano; Gemini-2.5-Flash; Gemini-2.5-Pro; Kimi-K2-Instruct; Magistral-Medium; Qwen3-235B-A22B-Fp8-Tput |
| 10 | 13 | 1,124 | Claude-Haiku-4-5; Claude-Opus-4-1; Claude-Sonnet-4-5; DeepSeek-V3.1; GPT-4.1; GPT-5-Mini; GPT-5-Nano; GPT-5.1; GPT-5.2; Gemini-2.5-Pro; Gemini-3-Flash-Preview; Grok-4-1-Fast-Non-Reasoning; Grok-4-1-Fast-Reasoning |
| 11 | 11 | 1,426 | Claude-3-5-Sonnet; Claude-Opus-4; Claude-Sonnet-4; GPT-4.1; Kimi-K2-Instruct; Llama-4-Maverick-17B-128E-Instruct-FP8; Llama-4-Scout-17B-16E-Instruct; Magistral-Medium; O4-Mini; Qwen3-235B-A22B-Fp8-Tput; claude-3-5-haiku |
| 12 | 10 | 1,060 | Claude-Haiku-4-5; Claude-Sonnet-4; DeepSeek-V3.1; GPT-4.1; GPT-5-Mini; GPT-5-Nano; Gemini-2.5-Pro; Grok-4-Fast-Non-Reasoning; Grok-4-Fast-Reasoning; Qwen3-235B-A22B-Thinking |
| 13 | 9 | 1,557 | Claude-3-5-Sonnet; Claude-3-Haiku; Claude-3-Opus; GPT-4-Turbo; GPT-4o; Llama-3-70b-Chat-Hf; Llama-3-8b-Chat-Hf; Mistral-Large; Qwen3-235B-A22B-Fp8-Tput |
| 14 | 9 | 1,069 | Claude-Haiku-4-5; Claude-Sonnet-4-5; Claude-Sonnet-4-6; GPT-5-Mini; GPT-5-Nano; GPT-5.4; GPT-5.4-Mini; GPT-5.4-Nano; Gemini-3.1-Pro-Preview |
| 15 | 8 | 1,521 | Claude-3-5-Sonnet; Claude-Opus-4-1; Claude-Sonnet-4; GPT-4.1; Kimi-K2-Instruct; Magistral-Medium; O4-Mini; Qwen3-235B-A22B-Fp8-Tput |
| 16 | 7 | 1,860 | Claude-3-5-Sonnet; Claude-3-7-Sonnet; DeepSeek-R1; DeepSeek-V3; GPT-4.5-Preview; Llama-3.3-70B-Instruct-Turbo; O3-Mini |
| **Overall, threshold 4,000** | | | |
| 1 | 9 | 5,112 | Claude-3-5-Sonnet; Claude-3-7-Sonnet; DeepSeek-R1; DeepSeek-V3; GPT-4o; Llama-3.3-70B-Instruct-Turbo; Mistral-Large; O3-Mini; QwQ-32B-Preview |
| 2 | 8 | 4,866 | Claude-3-5-Sonnet; Claude-3-7-Sonnet; GPT-4.1; GPT-4o; Llama-3.3-70B-Instruct-Turbo; Llama-4-Maverick-17B-128E-Instruct-FP8; Llama-4-Scout-17B-16E-Instruct; Mistral-Large |
| 3 | 7 | 5,907 | Claude-3-5-Sonnet; Claude-3-Haiku; Claude-3-Opus; GPT-4-Turbo; GPT-4o; Mistral-Large; Qwen3-235B-A22B-Fp8-Tput |
| 4 | 6 | 5,342 | Claude-3-5-Sonnet; Claude-3-Haiku; Claude-3-Opus; GPT-4-Turbo; GPT-4o; Llama-3-70b-Chat-Hf |
| 5 | 6 | 4,634 | Claude-3-5-Sonnet; Claude-Sonnet-4; GPT-4.1; Kimi-K2-Instruct; Magistral-Medium; Qwen3-235B-A22B-Fp8-Tput |
| 6 | 6 | 4,409 | Claude-3-5-Sonnet; GPT-4.1; Llama-4-Maverick-17B-128E-Instruct-FP8; Llama-4-Scout-17B-16E-Instruct; O4-Mini; Qwen3-235B-A22B-Fp8-Tput |
| 7 | 3 | 4,289 | Claude-Opus-4-1; GPT-4.1; Qwen3-235B-A22B-Fp8-Tput |
| 8 | 3 | 4,203 | Claude-3-7-Sonnet; GPT-4.1; Qwen3-235B-A22B-Fp8-Tput |
| 9 | 3 | 4,082 | Claude-Opus-4-1; Claude-Sonnet-4; GPT-4.1 |

Of the 91 model-family representatives, 70 appear in at least one final Overall group. We list these evaluated forecasters in Table 4, together with a reference for each model and an indicator of whether

its underlying weights are publicly downloadable. We retain the experiment identifiers so that readers can match the models to the group memberships reported above.

Table 4: The 70 evaluated forecasters in the size-sorted Overall comparison groups at $N = 1{,}000$ and $N = 4{,}000$. Models are grouped by provider and ordered by public release date within each provider. "Open weight" refers to publicly downloadable weights of the underlying model.

| LLM | Reference | Open weight? | Comparison groups ($N = 1{,}000$) | Comparison groups ($N = 4{,}000$) |
|---|---|---|---|---|
| GPT-3.5-Turbo | OpenAI (2023a) | No | G3 | N/A |
| GPT-4 | OpenAI (2023b) | No | G3 | N/A |
| GPT-4-Turbo | OpenAI (2023c) | No | G2, G3, G13 | G3, G4 |
| GPT-4o | OpenAI (2024) | No | G2, G3, G6, G8, G13 | G1, G2, G3, G4 |
| O3-Mini | OpenAI (2025f) | No | G6, G8, G16 | G1 |
| GPT-4.5-Preview | OpenAI (2025b) | No | G16 | N/A |
| GPT-4.1 | OpenAI (2025a) | No | G1, G2, G4, G5, G6, G7, G9, G10, G11, G12, G15 | G2, G5, G6, G7, G8, G9 |
| O3 | OpenAI (2025g) | No | G4, G6 | N/A |
| O4-Mini | OpenAI (2025g) | No | G2, G11, G15 | G6 |
| GPT-5 | OpenAI (2025e) | No | G9 | N/A |
| GPT-5-Mini | OpenAI (2025e) | No | G1, G4, G5, G7, G9, G10, G12, G14 | N/A |
| GPT-5-Nano | OpenAI (2025e) | No | G1, G4, G5, G7, G9, G10, G12, G14 | N/A |
| GPT-5.1 | OpenAI (2025c) | No | G1, G5, G7, G10 | N/A |
| GPT-5.2 | OpenAI (2025d) | No | G7, G10 | N/A |
| GPT-5.4 | OpenAI (2026a) | No | G14 | N/A |
| GPT-5.4-Mini | OpenAI (2026b) | No | G14 | N/A |
| GPT-5.4-Nano | OpenAI (2026b) | No | G14 | N/A |
| Claude-2.1 | Anthropic (2023) | No | G3 | N/A |
| Claude-3-Opus | Anthropic (2024a) | No | G2, G3, G13 | G3, G4 |
| Claude-3-Haiku | Anthropic (2024a) | No | G2, G3, G13 | G3, G4 |
| Claude-3-5-Sonnet | Anthropic (2024c) | No | G2, G3, G6, G8, G9, G11, G13, G15, G16 | G1, G2, G3, G4, G5, G6 |
| claude-3-5-haiku | Anthropic (2024b) | No | G11 | N/A |
| Claude-3-7-Sonnet | Anthropic (2025a) | No | G1, G2, G4, G6, G8, G16 | G1, G2, G8 |
| Claude-Opus-4 | Anthropic (2025b) | No | G11 | N/A |
| Claude-Sonnet-4 | Anthropic (2025b) | No | G9, G11, G12, G15 | G5, G9 |
| Claude-Opus-4-1 | Anthropic (2025d) | No | G1, G4, G5, G9, G10, G15 | G7, G9 |
| Claude-Sonnet-4-5 | Anthropic (2025e) | No | G1, G4, G5, G7, G10, G14 | N/A |
| Claude-Haiku-4-5 | Anthropic (2025c) | No | G1, G4, G5, G7, G10, G12, G14 | N/A |
| Claude-Opus-4-6 | Anthropic (2026a) | No | G7 | N/A |
| Claude-Sonnet-4-6 | Anthropic (2026b) | No | G7, G14 | N/A |
| DeepSeek-V3 | Liu et al. (2024) | Yes | G6, G8, G16 | G1 |
| DeepSeek-R1 | Guo et al. (2025a) | Yes | G6, G8, G16 | G1 |
| DeepSeek-V3.1 | DeepSeek (2025) | Yes | G1, G4, G5, G7, G10, G12 | N/A |
| Gemini-1.5-Pro | Google (2024b) | No | G3 | N/A |
| Gemini-1.5-Flash | Google (2024a) | No | G3 | N/A |
| Gemini-2.0-Flash-Lite-001 | Google (2025a) | No | G8 | N/A |
| Gemini-2.5-Pro-Exp | Google (2025b) | No | G8 | N/A |
| Gemini-2.5-Pro-Preview | Google (2025f) | No | G6 | N/A |
| Gemini-2.5-Flash-Preview | Google (2025f) | No | G6 | N/A |
| Gemini-2.5-Flash | Google (2025c) | No | G4, G9 | N/A |
| Gemini-2.5-Pro | Comanici et al. (2025) | No | G1, G4, G5, G7, G9, G10, G12 | N/A |
| Gemini-3-Pro-Preview | Google (2025d) | No | G1, G5 | N/A |
| Gemini-3-Flash-Preview | Google (2025e) | No | G7, G10 | N/A |
| Gemini-3.1-Pro-Preview | Google (2026) | No | G7, G14 | N/A |
| Kimi-K2-Instruct | Kimi Team (2025) | Yes | G9, G11, G15 | G5 |
| Kimi-K2-Thinking | Moonshot AI (2025) | Yes | G5 | N/A |
| Llama-2-70b-Chat-Hf | Meta AI (2023) | Yes | G3 | N/A |
| Llama-3-8b-Chat-Hf | Meta AI (2024b) | Yes | G3, G13 | N/A |
| Llama-3-70b-Chat-Hf | Meta AI (2024b) | Yes | G3, G13 | G4 |
| Meta-Llama-3.1-405B-Instruct-Turbo | Meta AI (2024a) | Yes | G2 | N/A |
| Llama-3.2-3B-Instruct-Turbo | Meta AI (2024c) | Yes | G2 | N/A |
| Llama-3.3-70B-Instruct-Turbo | Meta AI (2024d) | Yes | G2, G6, G8, G16 | G1, G2 |
| Llama-4-Scout-17B-16E-Instruct | Meta AI (2025) | Yes | G2, G6, G8, G11 | G2, G6 |
| Llama-4-Maverick-17B-128E-Instruct-FP8 | Meta AI (2025) | Yes | G2, G6, G8, G11 | G2, G6 |
| Mixtral-8x7B-Instruct-V0.1 | Mistral AI (2023) | Yes | G3 | N/A |
| Mixtral-8x22B-Instruct-V0.1 | Mistral AI (2024b) | Yes | G3 | N/A |
| Mistral-Large | Mistral AI (2024a) | Yes | G2, G6, G8, G13 | G1, G2, G3 |
| Magistral-Medium | Mistral AI (2025) | No | G9, G11, G15 | G5 |
| Qwen1.5-110B-Chat | Qwen Team (2024b) | Yes | G3 | N/A |
| Qwen2.5-72B-Instruct-Turbo | Qwen Team (2024c) | Yes | G2 | N/A |
| QwQ-32B-Preview | Qwen Team (2024a) | Yes | G6, G8 | G1 |
| Qwen3-235B-A22B-Fp8-Tput | Yang et al. (2025) | Yes | G1, G2, G4, G9, G11, G13, G15 | G3, G5, G6, G7, G8 |

Continued on next page

| LLM | Reference | Open weight? | Comparison groups ($N = 1{,}000$) | Comparison groups ($N = 4{,}000$) |
|---|---|---|---|---|
| Qwen3-235B-A22B-Thinking | Qwen Team (2025) | Yes | G1, G4, G5, G12 | N/A |
| Grok-beta | xAI (2024) | No | G8 | N/A |
| Grok-4 | xAI (2025b) | No | G1, G4 | N/A |
| Grok-4-Fast-Non-Reasoning | xAI (2025c) | No | G1, G4, G5, G12 | N/A |
| Grok-4-Fast-Reasoning | xAI (2025c) | No | G1, G4, G5, G12 | N/A |
| Grok-4-1-Fast-Non-Reasoning | xAI (2025a) | No | G1, G5, G7, G10 | N/A |
| Grok-4-1-Fast-Reasoning | xAI (2025a) | No | G1, G5, G7, G10 | N/A |
| GLM-4.5-Air-FP8 | Z.ai (2025) | Yes | G9 | N/A |

# D Robustness Checks and Controls

In this section, we conduct two additional analyses: Section D.1 checks whether our main results survive a much stricter shared-coverage requirement, while Section D.2 rules out individual recalibration as a simple alternative explanation for our weak-to-strong aggregation improvement.

## D.1 Higher Shared-Coverage Threshold $N = 4000$

In the following Table 5, we report statistics at $N = 4{,}000$ following the norm of Table 2 in Section 4.1. In particular, we report the overall two-model BS results across the 9 comparison groups and 97 eligible pairs. Results we report here are similar to what we have observed at $N = 1{,}000$. That is, aggregating two weaker models can still outperform the group benchmark at a higher shared-subquestion threshold.

Learned linear pooling identifies at least one pair that matches or improves on the group benchmark in 8 of the 9 groups (88.89%), and at least one pair within 5% of the benchmark in all 9 groups. Learned log-odds pooling matches or improves on the benchmark in 5 groups (55.56%) and comes within 5% in 6 groups (66.67%). These results show that pairs of individually weaker forecasters can perform close to, or better than, the group benchmark across many comparison groups.

Table 5: Two-model BS results at $N = 4{,}000$. Cells report the group-level and pair-level rates as well as the "Improve on both members" described in Section 3.2 in the form of fractions and percentages.

| | Groups | | Eligible pairs | | |
|---|---|---|---|---|---|
| Aggregation rule | Match or improve benchmark | Within 5% of benchmark | Match or improve benchmark | Within 5% of benchmark | Improve on both members |
| Learned linear pooling | 8/9 (88.89%) | 9/9 (100.00%) | 65/97 (67.01%) | 85/97 (87.63%) | 97/97 (100.00%) |
| Learned log-odds pooling | 5/9 (55.56%) | 6/9 (66.67%) | 15/97 (15.46%) | 30/97 (30.93%) | 97/97 (100.00%) |
| Simple mean (AM) | 4/9 (44.44%) | 5/9 (55.56%) | 12/97 (12.37%) | 18/97 (18.56%) | 71/97 (73.20%) |
| Harmonic mean (HM) | 3/9 (33.33%) | 5/9 (55.56%) | 4/97 (4.12%) | 12/97 (12.37%) | 42/97 (43.30%) |
| Log-odds mean (GM) | 3/9 (33.33%) | 5/9 (55.56%) | 7/97 (7.22%) | 14/97 (14.43%) | 50/97 (51.55%) |
| $\alpha$-log-odds | 3/9 (33.33%) | 5/9 (55.56%) | 6/97 (6.19%) | 13/97 (13.40%) | 40/97 (41.24%) |
| Average prior (AP) | 3/9 (33.33%) | 5/9 (55.56%) | 3/97 (3.09%) | 13/97 (13.40%) | 42/97 (43.30%) |
| Heuristic prior (HP) | 3/9 (33.33%) | 5/9 (55.56%) | 4/97 (4.12%) | 13/97 (13.40%) | 46/97 (47.42%) |
| Precision | 3/9 (33.33%) | 5/9 (55.56%) | 6/97 (6.19%) | 13/97 (13.40%) | 49/97 (50.52%) |

In Figure 2, we visualize these group-level results. For each comparison group, we report the group benchmark together with the lowest-BS pair under each learned rule, where the best pair is selected separately for linear and log-odds pooling. A best-pair point at or above the benchmark bar indicates that the corresponding group contains a successful pair. We thus observe directly from Figure 2 that

learned linear pooling produces a successful pair in 8 groups while learned log-odds pooling in 5 groups and pairwise aggregation is more significant in G3, G4 and G5.

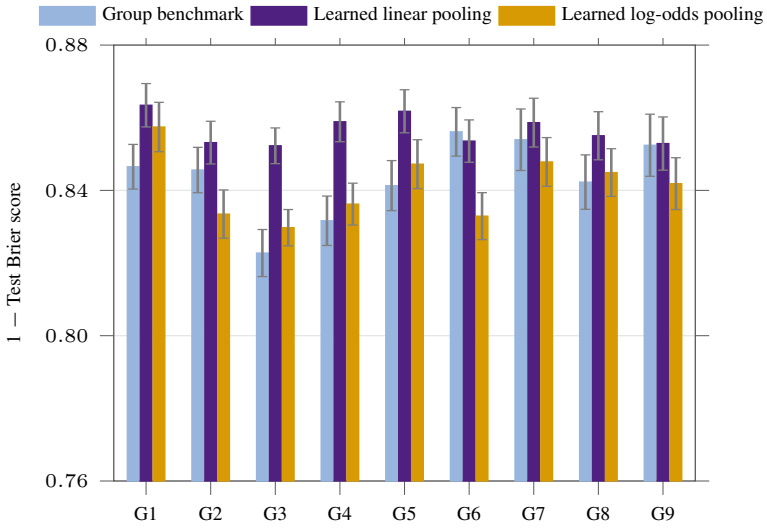



Figure 2: Best learned pair versus the group benchmark across the 9 comparison groups at $N = 4{,}000$. For each group, we plot test $1 - \text{BS}$ for the group benchmark and the lowest-BS pair under learned linear and log-odds pooling, selected separately for each rule. Higher bars ($1 - \text{BS}$) are better; groups are ordered by decreasing number of forecasters.

In the following Table 6, we report the group benchmark and the best weak-pair aggregate under each learned rule for each group of $N = 4{,}000$, selected by test Brier score (lower is better). Each Brier entry shows the point estimate and symmetric 95% CI half-width from 5,000 test-subquestion bootstrap resamples with the selected models and training-fitted coefficients held fixed. The intervals condition on the test-selected pairs. Each pair reports its training-fitted coefficients $(\omega_1, \omega_2)$ in the displayed model order, rounded to four decimals; coefficients are nonnegative and need not sum to one. Bold pair scores are smaller than the best individual's score, with comparisons made before rounding.

Table 6: Group benchmark and the best weak-pair aggregate under each pooling method for each of the size-sorted overall groups at $N = 4{,}000$, selected by test Brier score.

| Group | Size | Group Benchmark | Brier (95% CI) | Log-odds pooling best pair | Brier (95% CI) | Linear pooling best pair | Brier (95% CI) |
|---|---|---|---|---|---|---|---|
| **Overall, threshold 4,000** | | | | | | | |
| 1 | 9 | Llama-3.3-70B-Instruct-Turbo | 0.153 ($\pm$ 0.006) | O3-Mini + QwQ-32B-Preview $(\omega_1, \omega_2) = (0.3409, 0.5087)$ | **0.143** ($\pm$ 0.007) | Claude-3-5-Sonnet + QwQ-32B-Preview $(\omega_1, \omega_2) = (0.3828, 0.4115)$ | **0.137** ($\pm$ 0.006) |
| 2 | 8 | Llama-3.3-70B-Instruct-Turbo | 0.154 ($\pm$ 0.006) | Claude-3-7-Sonnet + GPT-4.1 $(\omega_1, \omega_2) = (0.4461, 0.2871)$ | 0.167 ($\pm$ 0.007) | Claude-3-5-Sonnet + GPT-4.1 $(\omega_1, \omega_2) = (0.4196, 0.2509)$ | **0.147** ($\pm$ 0.006) |
| 3 | 7 | Claude-3-Opus | 0.177 ($\pm$ 0.006) | Claude-3-5-Sonnet + Mistral-Large $(\omega_1, \omega_2) = (0.4287, 0.3415)$ | **0.170** ($\pm$ 0.005) | Claude-3-5-Sonnet + Mistral-Large $(\omega_1, \omega_2) = (0.5215, 0.1413)$ | **0.148** ($\pm$ 0.005) |
| 4 | 6 | Claude-3-Opus | 0.168 ($\pm$ 0.007) | Claude-3-5-Sonnet + GPT-4-Turbo $(\omega_1, \omega_2) = (0.4946, 0.3134)$ | **0.164** ($\pm$ 0.006) | Claude-3-5-Sonnet + Llama-3-70b-Chat-Hf $(\omega_1, \omega_2) = (0.5947, 0.0749)$ | **0.141** ($\pm$ 0.005) |
| 5 | 6 | Magistral-Medium | 0.159 ($\pm$ 0.007) | Claude-3-5-Sonnet + GPT-4.1 $(\omega_1, \omega_2) = (0.3594, 0.4522)$ | **0.153** ($\pm$ 0.007) | Claude-3-5-Sonnet + GPT-4.1 $(\omega_1, \omega_2) = (0.3725, 0.3621)$ | **0.138** ($\pm$ 0.006) |
| 6 | 6 | O4-Mini | 0.144 ($\pm$ 0.007) | Claude-3-5-Sonnet + GPT-4.1 $(\omega_1, \omega_2) = (0.3313, 0.3399)$ | 0.167 ($\pm$ 0.006) | Claude-3-5-Sonnet + GPT-4.1 $(\omega_1, \omega_2) = (0.4427, 0.2476)$ | 0.146 ($\pm$ 0.006) |
| 7 | 3 | GPT-4.1 | 0.146 ($\pm$ 0.008) | Claude-Opus-4-1 + Qwen3-235B-A22B-Fp8-Tput $(\omega_1, \omega_2) = (0.4916, 0.2936)$ | 0.152 ($\pm$ 0.007) | Claude-Opus-4-1 + Qwen3-235B-A22B-Fp8-Tput $(\omega_1, \omega_2) = (0.5025, 0.2241)$ | **0.141** ($\pm$ 0.007) |
| 8 | 3 | Claude-3-7-Sonnet | 0.158 ($\pm$ 0.007) | GPT-4.1 + Qwen3-235B-A22B-Fp8-Tput $(\omega_1, \omega_2) = (0.4500, 0.2026)$ | **0.155** ($\pm$ 0.007) | GPT-4.1 + Qwen3-235B-A22B-Fp8-Tput $(\omega_1, \omega_2) = (0.4760, 0.2545)$ | **0.145** ($\pm$ 0.007) |

Continued on next page

| Group | Size | Group Benchmark | Brier (95% CI) | Log-odds pooling best pair | Brier (95% CI) | Linear pooling best pair | Brier (95% CI) |
|---|---|---|---|---|---|---|---|
| 9 | 3 | GPT-4.1 | 0.148 ($\pm$ 0.009) | Claude-Opus-4-1 + Claude-Sonnet-4 $(\omega_1, \omega_2) = (0.4047, 0.4146)$ | 0.158 ($\pm$ 0.007) | Claude-Opus-4-1 + Claude-Sonnet-4 $(\omega_1, \omega_2) = (0.4455, 0.2827)$ | **0.147** ($\pm$ 0.007) |

### D.2 Individual Recalibration as a Control

We use an individual-recalibration control to examine whether fitted probability rescaling alone can explain the BS gains under learned linear pooling. In each of the 16 overall comparison groups, we compare the lowest-test-BS calibrated weak forecaster with the best aggregation pair in the group and the group benchmark.

We calibrate each individually weaker forecaster separately using two-parameter Platt scaling, following the post-processing form studied by Murphy (2026). For each forecaster, we fit a slope $a$ and an intercept $b$ on its group's original training set by minimizing instance-weighted log loss. We transform its forecast probability $p$ as $\tilde{p} = \sigma(a \operatorname{logit}(p) + b)$. We then keep the fitted parameters fixed and evaluate the calibrated forecast on the group's common test set.

Following the norm of the evaluation procedure in Section 4.1, we compare these calibrated forecasters with the group benchmark and the aggregation pairs reported in Section 4.1. In Table 7, we report the fraction of the 16 comparison groups in which the best calibrated forecaster matches or improves on the group benchmark or the best aggregation pair of the group.

Table 7: Group-level BS results for individual recalibration at $N = 1{,}000$. Cells report fractions and percentages of the 16 overall comparison groups in which the lowest-test-BS calibrated forecaster matches or improves on the group benchmark or the best aggregation pair of the group.

| | Comparison groups | |
|---|---|---|
| Calibration rule | Match or improve group benchmark | Match or improve best aggregation pair |
| Per-model Platt scaling | 7/16 (43.75%) | 2/16 (12.50%) |

**Individual recalibration is not that competent.** Using the same match-or-improve criterion as in Section 3.2, we find that the best calibrated forecaster matches or improves on the group benchmark in 7 of the 16 groups (43.75%) and on the best aggregation pair in 2 groups (12.50%).

These results show that individual-recalibration does not reproduce the best aggregation pair's test BS in most comparison groups. These results show that aggregation can improve forecast accuracy beyond individual recalibration.

## E Additional Pairwise Aggregation Results

**Best pairs across comparison groups.** In Figure 1, we visualize the group-level results reported in Section 4. For each comparison group, we report the group benchmark together with the lowest-BS pair under each learned rule, where the best pair is selected separately for linear and log-odds pooling. A best-pair point at or above the benchmark bar indicates that the corresponding group contains a successful pair. We thus observe directly from Figure 1 that learned linear pooling produces a successful pair in 11 groups while learned log-odds pooling in 8 groups and pairwise aggregation is more significant in G3, G6, G8, G13 and G16. The figure also shows that their relative performance may vary across groups: linear pooling succeeds where log-odds pooling does not in G9, G12, G15, and G16, while the reverse occurs in G11.

In the following Table 8, we report the group benchmark and the best weak-pair aggregate under each learned rule for each group of $N = 1{,}000$, selected by test Brier score (lower is better). Each Brier entry shows the point estimate and symmetric 95% CI half-width from 5,000 test-subquestion bootstrap resamples with the selected models and training-fitted coefficients held fixed. The intervals condition on the test-selected pairs. Each pair reports its training-fitted coefficients $(\omega_1, \omega_2)$ in the

displayed model order, rounded to four decimals; coefficients are nonnegative and need not sum to one. Bold pair scores are smaller than the best individual's score, with comparisons made before rounding.

Table 8: Group benchmark and the best weak-pair aggregate under each pooling method for each of the size-sorted overall groups at $N = 1{,}000$, selected by test Brier score.

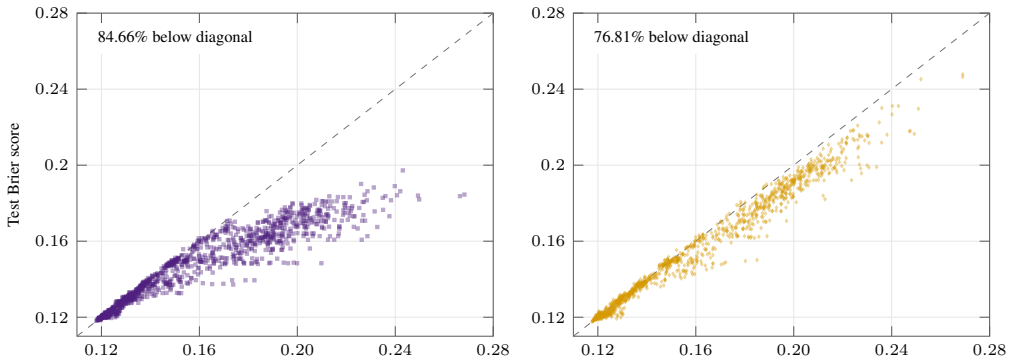
| Group | Size | Group Benchmark | Brier (95% CI) | Log-odds pooling best pair | Brier (95% CI) | Linear pooling best pair | Brier (95% CI) |
|---|---|---|---|---|---|---|---|
| **Overall, threshold 1,000** | | | | | | | |
| 1 | 18 | Gemini-3-Pro-Preview | 0.115 ($\pm$ 0.012) | Claude-3-7-Sonnet + GPT-5-Mini $(\omega_1, \omega_2) = (0.6117, 0.4237)$ | 0.118 ($\pm$ 0.012) | Claude-Opus-4-1 + GPT-5-Mini $(\omega_1, \omega_2) = (0.4390, 0.5893)$ | 0.118 ($\pm$ 0.012) |
| 2 | 16 | O4-Mini | 0.147 ($\pm$ 0.009) | Llama-3.3-70B-Instruct-Turbo + Meta-Llama-3.1-405B-Instruct-Turbo $(\omega_1, \omega_2) = (1.0825, 0.4939)$ | **0.139** ($\pm$ 0.009) | Llama-3.3-70B-Instruct-Turbo + Meta-Llama-3.1-405B-Instruct-Turbo $(\omega_1, \omega_2) = (0.3395, 0.4624)$ | **0.143** ($\pm$ 0.008) |
| 3 | 16 | Gemini-1.5-Pro | 0.171 ($\pm$ 0.010) | Claude-3-5-Sonnet + Llama-3-8b-Chat-Hf $(\omega_1, \omega_2) = (0.6165, 0.2859)$ | **0.170** ($\pm$ 0.010) | Claude-3-5-Sonnet + GPT-4o $(\omega_1, \omega_2) = (0.5320, 0.1326)$ | **0.148** ($\pm$ 0.009) |
| 4 | 16 | GPT-5-Mini | 0.120 ($\pm$ 0.012) | Claude-3-7-Sonnet + Grok-4 $(\omega_1, \omega_2) = (0.6628, 0.3542)$ | **0.118** ($\pm$ 0.011) | Claude-Opus-4-1 + Grok-4 $(\omega_1, \omega_2) = (0.5310, 0.4611)$ | **0.118** ($\pm$ 0.011) |
| 5 | 16 | Gemini-3-Pro-Preview | 0.125 ($\pm$ 0.011) | DeepSeek-V3.1 + GPT-5-Mini $(\omega_1, \omega_2) = (0.5378, 0.5612)$ | 0.126 ($\pm$ 0.011) | Claude-Opus-4-1 + GPT-5-Mini $(\omega_1, \omega_2) = (0.4005, 0.6186)$ | 0.126 ($\pm$ 0.011) |
| 6 | 15 | Llama-3.3-70B-Instruct-Turbo | 0.179 ($\pm$ 0.012) | DeepSeek-R1 + QwQ-32B-Preview $(\omega_1, \omega_2) = (0.3993, 0.3646)$ | **0.167** ($\pm$ 0.012) | DeepSeek-R1 + Gemini-2.5-Flash-Preview $(\omega_1, \omega_2) = (0.4370, 0.3543)$ | **0.158** ($\pm$ 0.010) |
| 7 | 15 | Grok-4-1-Fast-Reasoning | 0.147 ($\pm$ 0.014) | Claude-Haiku-4-5 + GPT-5-Nano $(\omega_1, \omega_2) = (0.7454, 0.2705)$ | **0.145** ($\pm$ 0.013) | GPT-5.2 + Gemini-3.1-Pro-Preview $(\omega_1, \omega_2) = (0.5315, 0.4918)$ | **0.146** ($\pm$ 0.013) |
| 8 | 14 | Llama-3.3-70B-Instruct-Turbo | 0.177 ($\pm$ 0.011) | O3-Mini + QwQ-32B-Preview $(\omega_1, \omega_2) = (0.3661, 0.4486)$ | **0.164** ($\pm$ 0.013) | Claude-3-5-Sonnet + QwQ-32B-Preview $(\omega_1, \omega_2) = (0.3989, 0.3729)$ | **0.157** ($\pm$ 0.011) |
| 9 | 13 | GPT-5 | 0.129 ($\pm$ 0.010) | Claude-Opus-4-1 + Gemini-2.5-Pro $(\omega_1, \omega_2) = (0.6456, 0.3601)$ | 0.134 ($\pm$ 0.011) | Claude-Opus-4-1 + Gemini-2.5-Pro $(\omega_1, \omega_2) = (0.4733, 0.3017)$ | **0.128** ($\pm$ 0.009) |
| 10 | 13 | Grok-4-1-Fast-Reasoning | 0.135 ($\pm$ 0.012) | Claude-Haiku-4-5 + GPT-5-Mini $(\omega_1, \omega_2) = (0.4209, 0.6278)$ | **0.135** ($\pm$ 0.012) | GPT-5-Mini + Gemini-2.5-Pro $(\omega_1, \omega_2) = (0.4174, 0.5728)$ | **0.135** ($\pm$ 0.012) |
| 11 | 11 | O4-Mini | 0.132 ($\pm$ 0.010) | Claude-Opus-4 + claude-3-5-haiku $(\omega_1, \omega_2) = (0.5125, 0.7849)$ | **0.131** ($\pm$ 0.011) | Claude-Opus-4 + Magistral-Medium $(\omega_1, \omega_2) = (0.5112, 0.2484)$ | 0.133 ($\pm$ 0.009) |
| 12 | 10 | Gemini-2.5-Pro | 0.127 ($\pm$ 0.013) | GPT-4.1 + GPT-5-Mini $(\omega_1, \omega_2) = (0.1859, 0.6261)$ | 0.128 ($\pm$ 0.012) | GPT-4.1 + GPT-5-Mini $(\omega_1, \omega_2) = (0.2663, 0.7316)$ | **0.127** ($\pm$ 0.013) |
| 13 | 9 | Claude-3-Opus | 0.180 ($\pm$ 0.012) | Claude-3-5-Sonnet + Mistral-Large $(\omega_1, \omega_2) = (0.4045, 0.4091)$ | **0.173** ($\pm$ 0.010) | Claude-3-5-Sonnet + Mistral-Large $(\omega_1, \omega_2) = (0.4535, 0.2284)$ | **0.152** ($\pm$ 0.009) |
| 14 | 9 | Gemini-3.1-Pro-Preview | 0.155 ($\pm$ 0.013) | Claude-Haiku-4-5 + Claude-Sonnet-4-6 $(\omega_1, \omega_2) = (0.1206, 0.7256)$ | 0.157 ($\pm$ 0.012) | Claude-Sonnet-4-6 + GPT-5-Nano $(\omega_1, \omega_2) = (0.9308, 0.1264)$ | 0.156 ($\pm$ 0.013) |
| 15 | 8 | O4-Mini | 0.143 ($\pm$ 0.011) | Claude-3-5-Sonnet + Magistral-Medium $(\omega_1, \omega_2) = (0.5238, 0.5429)$ | 0.147 ($\pm$ 0.010) | Claude-3-5-Sonnet + GPT-4.1 $(\omega_1, \omega_2) = (0.4780, 0.2474)$ | **0.137** ($\pm$ 0.010) |
| 16 | 7 | Llama-3.3-70B-Instruct-Turbo | 0.165 ($\pm$ 0.011) | Claude-3-7-Sonnet + DeepSeek-R1 $(\omega_1, \omega_2) = (0.4528, 0.4301)$ | 0.174 ($\pm$ 0.010) | DeepSeek-R1 + GPT-4.5-Preview $(\omega_1, \omega_2) = (0.2698, 0.4374)$ | **0.155** ($\pm$ 0.009) |

**Robust aggregation baselines.** Table 9 below is regarded as an extension of Table 2; we report the overall two-model BS results across the 16 comparison groups and 1,121 eligible pairs for each robust aggregation rule. In particular, we report the fraction of groups and pairs that match or improve on the group benchmark, the corresponding fractions within 5% of the benchmark, and the fraction of pairs whose aggregate improves on both constituent forecasters in the same format as Table 2.

Table 9: Additional robust-rule results corresponding to Table 2

| | Groups | | Eligible pairs | | |
|---|---|---|---|---|---|
| Aggregation rule | Match or improve benchmark | Within 5% of benchmark | Match or improve benchmark | Within 5% of benchmark | Improve on both members |
| $\alpha$-log-odds | 9/16 (56.25%) | 15/16 (93.75%) | 44/1,121 (3.93%) | 351/1,121 (31.31%) | 605/1,121 (53.97%) |
| Average prior (AP) | 5/16 (31.25%) | 14/16 (87.50%) | 40/1,121 (3.57%) | 336/1,121 (29.97%) | 600/1,121 (53.52%) |
| Heuristic prior (HP) | 6/16 (37.50%) | 14/16 (87.50%) | 36/1,121 (3.21%) | 326/1,121 (29.08%) | 622/1,121 (55.49%) |
| Precision | 10/16 (62.50%) | 15/16 (93.75%) | 48/1,121 (4.28%) | 362/1,121 (32.29%) | 658/1,121 (58.70%) |

**Learned versus equal-weight pooling.** We compare each learned rule with its equal-weight counterpart on the same 1,121 pairs in the following Figure 3. The horizontal axis reports the BS under the equal-weight rule, while the vertical axis reports the BS under the corresponding learned rule. A point below the diagonal therefore indicates that learning the coefficients reduces BS for that pair. We find that learned linear pooling has lower BS than the simple mean for 949 of the 1,121 pairs (84.66%), while learned log-odds pooling has lower BS than the log-odds mean for 861 pairs (76.81%). These matched comparisons show that the benefit of learning extends well beyond the pairs that cross the group benchmark. The figure also shows that learning does not improve every pair, since some points lie above the diagonal. Overall, however, most pairs have lower BS under the learned rules than under their equal-weight counterparts.




(a) Learned linear pooling vs. simple mean. (b) Learned log-odds pooling vs. log-odds mean.

Figure 3: Learned aggregation versus the corresponding equal-weight rule on the same 1,121 pairs. Each point compares the test BS of a learned aggregate with its equal-weight counterpart. Points below the diagonal indicate lower BS under the learned rule.

**Near-best and no-near pairs.** In the following Table 10, we demonstrate the near/no-near decomposition of the group- and pair-level within-5% BS results at $N = 1{,}000$. Group-level columns report, among the 16 comparison groups, the fraction containing a pair within 5% of the benchmark and, separately, the fractions of the within 5% groups whose pair set includes at least one near pair or at least one no-near pair. We note that a group may contain both types, so the latter two fractions need not sum to one. Pair-level columns decompose the pairs within 5% of the benchmark into near and no-near pairs, together with the fraction of all pairs meeting the criterion.

Table 10: Near/no-near decomposition of the group- and pair-level within-5% BS results at $N = 1{,}000$.

| | Groups with a pair within 5% | | | Pairs within 5% | | |
|---|---|---|---|---|---|---|
| Aggregation rule | Fraction of all groups | Share with a near pair within 5% | Share with a no-near pair within 5% | Fraction of all pairs | Near-pair share | No-near-pair share |
| Learned linear pooling | 16/16 (100.00%) | 9/16 (56.25%) | 15/16 (93.75%) | 730/1,121 (65.12%) | 323/730 (44.25%) | 407/730 (55.75%) |
| Learned log-odds pooling | 15/16 (93.75%) | 9/15 (60.00%) | 15/15 (100.00%) | 467/1,121 (41.66%) | 320/467 (68.52%) | 147/467 (31.48%) |
| Simple mean (AM) | 15/16 (93.75%) | 9/15 (60.00%) | 15/15 (100.00%) | 345/1,121 (30.78%) | 250/345 (72.46%) | 95/345 (27.54%) |
| Harmonic mean (HM) | 16/16 (100.00%) | 9/16 (56.25%) | 15/16 (93.75%) | 349/1,121 (31.13%) | 263/349 (75.36%) | 86/349 (24.64%) |
| Log-odds mean (GM) | 15/16 (93.75%) | 9/15 (60.00%) | 15/15 (100.00%) | 375/1,121 (33.45%) | 262/375 (69.87%) | 113/375 (30.13%) |
| $\alpha$-log-odds | 15/16 (93.75%) | 9/15 (60.00%) | 15/15 (100.00%) | 351/1,121 (31.31%) | 255/351 (72.65%) | 96/351 (27.35%) |
| Average prior (AP) | 14/16 (87.50%) | 9/14 (64.29%) | 14/14 (100.00%) | 336/1,121 (29.97%) | 251/336 (74.70%) | 85/336 (25.30%) |
| Heuristic prior (HP) | 14/16 (87.50%) | 9/14 (64.29%) | 14/14 (100.00%) | 326/1,121 (29.08%) | 251/326 (76.99%) | 75/326 (23.01%) |
| Precision | 15/16 (93.75%) | 9/15 (60.00%) | 15/15 (100.00%) | 362/1,121 (32.29%) | 261/362 (72.10%) | 101/362 (27.90%) |

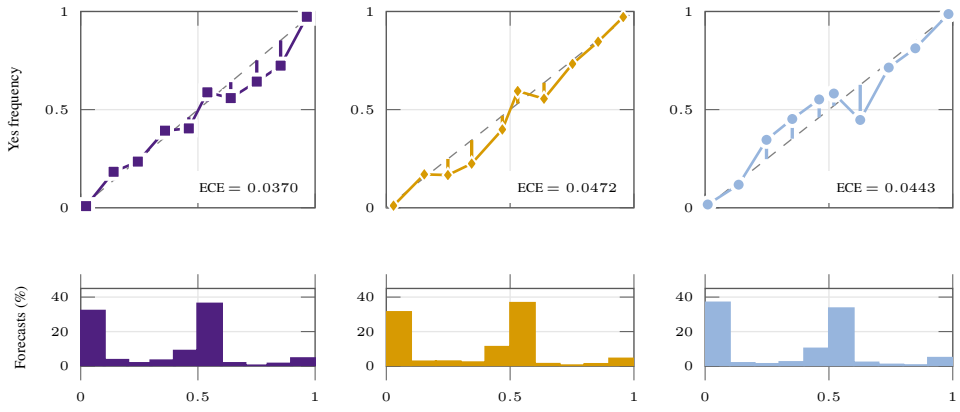
**Calibration results.** In Table 11 below, we report the group and pair level results evaluated by ECE. We retain the BS-best benchmark, eligible pairs, and BS-trained coefficients used in Table 2 and evaluate the same forecasts using ECE. Cells report group- and pair-level rates and improvement over both members.

Table 11: Two-model ECE results at $N = 1{,}000$, retaining the BS-selected group benchmark, eligible pairs, and BS-trained coefficients.

| | Groups | | Eligible pairs | | |
|---|---|---|---|---|---|
| Aggregation rule | Match or improve benchmark | Within 5% of benchmark | Match or improve benchmark | Within 5% of benchmark | Improve on both members |
| Learned linear pooling | 16/16 (100.00%) | 16/16 (100.00%) | 772/1,121 (68.87%) | 826/1,121 (73.68%) | 705/1,121 (62.89%) |
| Learned log-odds pooling | 13/16 (81.25%) | 13/16 (81.25%) | 291/1,121 (25.96%) | 336/1,121 (29.97%) | 577/1,121 (51.47%) |
| Simple mean (AM) | 12/16 (75.00%) | 12/16 (75.00%) | 271/1,121 (24.17%) | 317/1,121 (28.28%) | 471/1,121 (42.02%) |
| Harmonic mean (HM) | 11/16 (68.75%) | 13/16 (81.25%) | 277/1,121 (24.71%) | 318/1,121 (28.37%) | 514/1,121 (45.85%) |
| Log-odds mean (GM) | 11/16 (68.75%) | 12/16 (75.00%) | 313/1,121 (27.92%) | 348/1,121 (31.04%) | 470/1,121 (41.93%) |
| $\alpha$-log-odds | 9/16 (56.25%) | 10/16 (62.50%) | 296/1,121 (26.40%) | 336/1,121 (29.97%) | 420/1,121 (37.47%) |
| Average prior (AP) | 9/16 (56.25%) | 9/16 (56.25%) | 288/1,121 (25.69%) | 325/1,121 (28.99%) | 378/1,121 (33.72%) |
| Heuristic prior (HP) | 8/16 (50.00%) | 9/16 (56.25%) | 261/1,121 (23.28%) | 311/1,121 (27.74%) | 393/1,121 (35.06%) |
| Precision | 9/16 (56.25%) | 10/16 (62.50%) | 300/1,121 (26.76%) | 339/1,121 (30.24%) | 441/1,121 (39.34%) |

Below is Figure 4, showing the additional reliability diagrams for a representative comparison group G1 of Section 4. The linear pair is Claude-Opus-4-1 with GPT-5-Mini; the log-odds pair is Claude-3-7-Sonnet with GPT-5-Mini; the group benchmark is Gemini-3-Pro-Preview. We select the pairs by BS. The top panels use ten equal-width probability bins. The dashed diagonal indicates perfect

calibration, and vertical segments show the gap between each bin's mean forecast and its Yes frequency. Empty bins have no marker. The bottom panels show the percentage of test forecasts in each bin. The three panels evaluate the same 533 test subquestions in G1.




(a) Linear pooling. (b) Log-odds pooling. (c) Group benchmark.

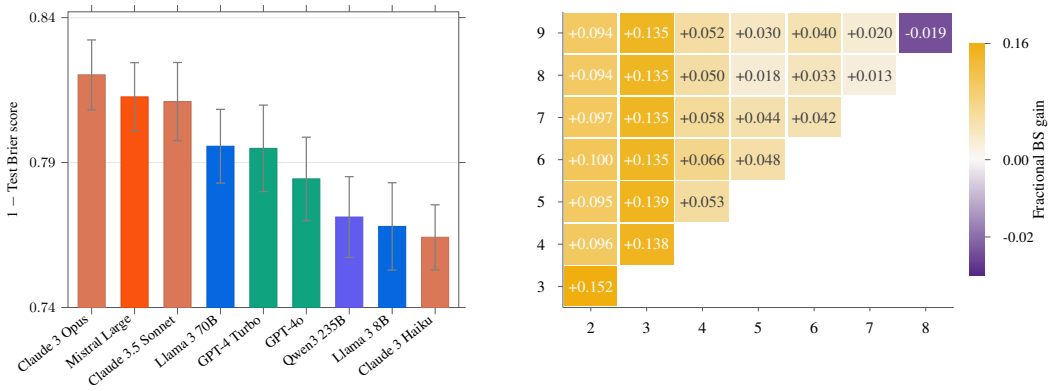
Figure 4: Reliability and forecast distributions for the rule-specific eligible pairs and group benchmark in G1.

## F Case Study: When Learned Linear Pooling Succeeds

In this section, under our pairwise aggregation results, we zoom into Group G13 to understand why the choice of pooling rule can matter even when the constituent forecasters are held fixed. This group contains 9 forecasters with 1,557 shared subquestions, of which 779 belong to the test set. In Figure 5a, we report the $1 - \text{BS}$ of each individual forecaster, where we can see that the group benchmark is Claude 3 Opus. The remaining eight forecasters form 28 eligible pairs. In Figure 5b, we report their aggregated performance. As we can see, linear pooling succeeds broadly across pair compositions. Learned linear pooling matches or improves on the group benchmark for 27 of the 28 pairs (96.43%), compared with only 2 pairs (7.14%) under learned log-odds pooling. Figure 5a shows the group benchmark has test $1-\text{BS}$ of $0.820$. Figure 5b further shows that the linear-pooling gains are not concentrated among pairs containing the strongest of these weaker forecasters. The improved forecast performance appears across almost the entire pair-rank matrix, including pairs formed by relatively low-ranked individuals; only one pair has a negative gain. Thus, the strong performance of linear pooling is not driven by a small number of favorable pair compositions.

In addition, we also observe that the pooling rule matters even for the same pair. Figure 6 compares learned linear and log-odds pooling on the same 28 pairs. We observe higher test $1 - \text{BS}$ under linear pooling in all 28 matched comparisons, with differences ranging from 0.0179 to 0.038. The best observed linear aggregate reaches $1 - \text{BS} = 0.848$, compared with 0.827 under learned log-odds pooling. These matched comparisons show that the aggregation rule itself can substantially affect whether a pair of weaker forecasters closes the gap to the group benchmark.

Figure 7 looks into G13 through the lens of calibration. Figure 7a and Figure 7b combine Claude-3-5-Sonnet with Llama-3-70b-Chat-Hf using learned linear and log-odds pooling, respectively; panel Figure 7c shows the BS-best individual, Claude-3-Opus. The three panels evaluate the same 779 test subquestions in G13. The binning and plotting conventions follow Figure 4.




(a) Individual forecasts 1 - test BS.

(b) Learned-linear BS gain by pair ranks.

Figure 5: Individual and pairwise forecasting performance in Group G13. (a) Individual forecasters' performance. (b) Fractional test-BS reduction of learned-linear pairs relative to the benchmark, indexed by the ranks of their two members; positive values favor the pair.

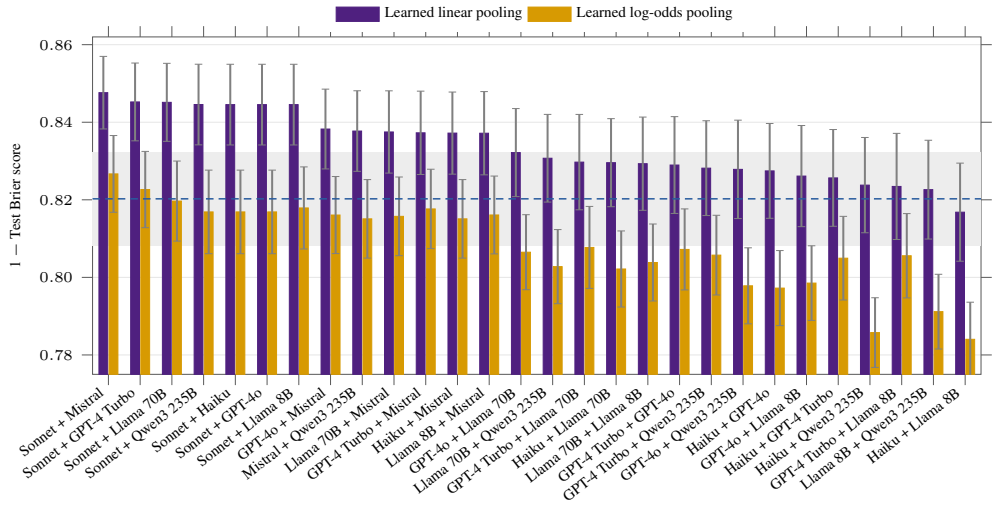
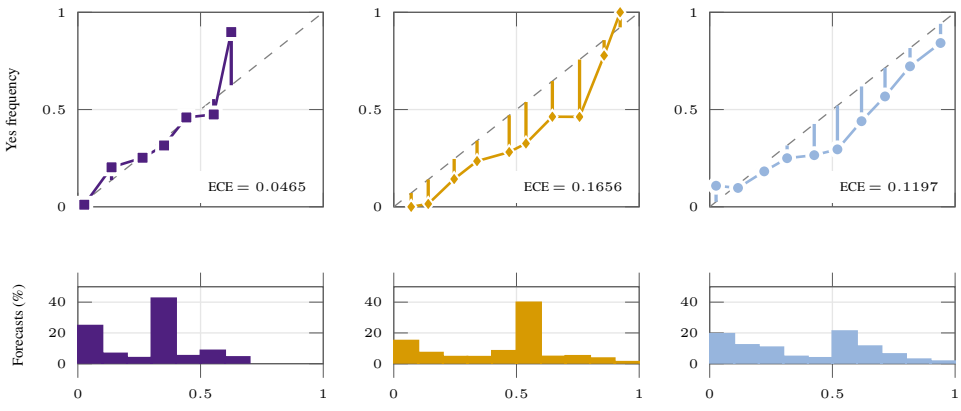



Figure 6: Learned linear and log-odds pooling on the same 28 eligible pairs of this section. The dashed line and gray band are the benchmark 1 - Test BS and its 95% CI, respectively.




(a) Linear pooling.

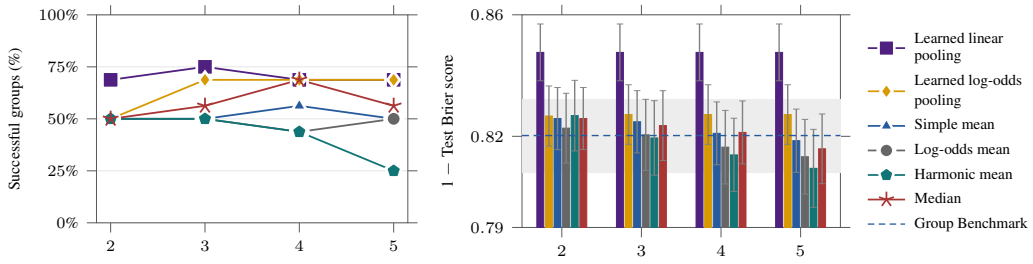
(b) Log-odds pooling.

(c) Group benchmark.

Figure 7: Reliability and forecast distributions for a fixed eligible pair under each learned pooling rule and the group benchmark in G13.

## G Additional Multi-Model Aggregation Results

We provide additional comparisons across aggregation rules for the multi-model experiments in Section 4.2. In Figure 8, we report results for exact pool sizes $k \in \{2, 3, 4, 5\}$. For each aggregation rule and pool size, the best-performing pool is selected separately. Thus, the curves compare the best available pools at each exact size rather than tracking a fixed pool as additional forecasters are added. In Figure 8, we additionally compare the learned rules with the classical aggregation rules. At the group level, learned linear pooling remains among the strongest-performing rules across all pool sizes, whereas the classical rules exhibit more variable performance as the pool grows. In particular, their success rates do not generally increase with pool size, and some decline for larger pools. Figure 8b shows a similar pattern within G13: the best performance under the classical rules fluctuates or decreases as the pool grows and generally remains below that of learned linear pooling. Thus, the lack of systematic gains from larger pools is also visible across the fixed aggregation rules, rather than being specific to the learned rules.




(a) Group-level success by number of forecasters.

(b) Best performance in G13 by number of forecasters.

Figure 8: Aggregation performance across exact pool sizes. (a) Fraction of the 16 comparison groups containing at least one pool that matches or improves on the group benchmark in terms of test BS, for pool sizes $k \in \{2, 3, 4, 5\}$. (b) Best test $1 - \text{BS}$ in group G13 for each aggregation rule and pool size. The dashed line marks the group benchmark in group G13. For each aggregation rule and pool size, the best-performing pool is selected separately.

## H Practical Constraints

We next examine whether useful weak-pair aggregates remain available when access to the strongest individual forecaster is constrained in practice. We consider two restrictions: *lower input price* and *open-weight access*. In both analyses, we start from the same two-model comparisons as in Section 4.1 and keep the fitted aggregation rules and group benchmarks fixed.

**Lower input price.** For each model, we use the published API input price, measured per million input tokens, at the time the specific model version (snapshot) evaluated in our experiments was released. We use input price as a simple and comparable cost proxy because the forecasting task typically requires only a short probability output. For a pair to satisfy the lower-input-price constraint, the sum of the two models' published input prices must be strictly below that of the group benchmark. We then ask whether such a cheaper pair can still match or approach the benchmark after aggregation.

**Open-weight access.** We also consider settings in which the group benchmark is closed-weight but locally deployable models are required. We restrict attention to the 10 comparison groups with a closed-weight benchmark and at least one eligible open-weight pair, and evaluate aggregates formed only from such open-weight forecasters. Our open-weight classifications are reported in Table 4.

**Group-level results.** In Table 12, we summarize the performance under both constraints. Under the lower-input-price restriction, learned linear pooling identifies a pair that matches or improves on the benchmark in 2 of the 16 groups and comes within 5% in 5 groups; learned log-odds pooling does so in 1 and 3 groups, respectively. Under the open-weight restriction, learned linear pooling matches or improves on the benchmark in 3 of the 10 eligible groups and comes within 5% in 4, compared with 1 and 3 groups for learned log-odds pooling. Thus, although these restrictions

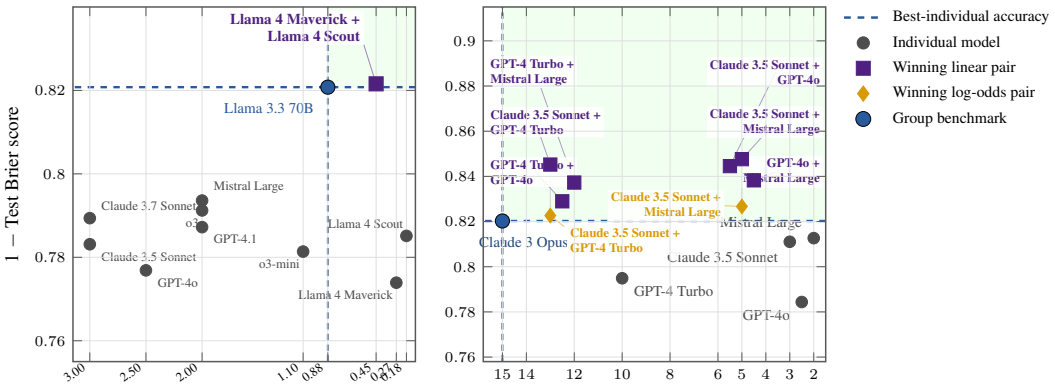
substantially narrow the candidate set, they still leave comparison groups in which two individually weaker models provide a competitive alternative to the stronger benchmark

Table 12: Group-level results under practical constraints. Entries report the fraction of groups with at least one constrained pair that matches or improves on, or comes within 5% of, the group benchmark. Denominators are 16 groups for lower input price and 10 groups for open weights.

| | Lower input price | | Open weights | |
|---|---|---|---|---|
| Aggregation rule | Match or improve benchmark | Within 5% of benchmark | Match or improve benchmark | Within 5% of benchmark |
| Learned linear pooling | 2/16 (12.50%) | 5/16 (31.25%) | 3/10 (30.00%) | 4/10 (40.00%) |
| Learned log-odds pooling | 1/16 (6.25%) | 3/16 (18.75%) | 1/10 (10.00%) | 3/10 (30.00%) |

**Price-accuracy trade-offs.** In Figure 9, we provide a more detailed view of the lower-input-price constraint in groups G6 and G13. The figure places individual models and successful constrained pairs in the same price–accuracy space, where the cost of a pair is the sum of the published input prices of its two members. A successful constrained pair therefore provides both lower input cost and higher forecast accuracy than the group benchmark. As we can see, G6 contains one such pair, whereas G13 contains several alternatives across different price levels.

The gains can be substantial when such alternatives exist. In G13, for example, Claude-3-5-Sonnet combined with Mistral-Large has a total input price of $5 per million tokens, compared with $15 for the Claude-3-Opus benchmark. The pair reduces BS by 15.23% under learned linear pooling and by 3.58% under learned log-odds pooling. Figure 9 therefore illustrates that weak-to-strong aggregation can generate a meaningful price–accuracy trade-off rather than merely a near-benchmark substitute




(a) Group G6. (b) Group G13.

Figure 9: Input-price versus BS frontiers for groups. Figure 9a shows G6 (1,512 common subquestions) and Figure 9b shows G13 (1,557 common subquestions) at the 1,000-coverage threshold. The x-axis shows published input list prices (USD per million tokens), decreasing from left to right. Gray circles are individual models; the blue circle is each group's BS-best model. Purple squares and gold diamonds show, respectively, linear and log-odds input-cheaper weak-model pairs whose learned aggregate strictly exceeds the benchmark on the test half; Figure 9a contains only one such pair. Pair cost is the sum of the two members' standard input-token rates. Complete pair names are attached directly to their aggregate points, and method-specific scores example appear in Table 13.

**Best-performing constrained pairs.** In Table 13, we report, for each practical constraint and learned aggregation rule, the constrained pair with the largest relative BS reduction. These examples complement the group-level results in Table 12 by showing the magnitude of the gains when a successful constrained alternative exists. Under the open-weight constraint, for example, the best learned-linear pair combines Mistral-Large with Qwen3-235B-A22B and reduces BS by 9.72% relative to the Claude-3-Opus benchmark. The corresponding learned log-odds example combines Llama-3.3-70B with Llama-3.1-405B and reduces BS by 4.87% relative to O4-Mini.

Table 13: Pairs with the largest relative BS reduction under each practical constraint and learned rule. We select each example retrospectively relative to its own group's benchmark. Both members are individually weaker than that benchmark under BS. The final column reports the percentage reduction in BS; larger is better. Model names are shortened for readability.

| Constraint and group | Learned rule | Pair | Group benchmark | Benchmark BS ↓ | Pair BS ↓ | Relative BS reduction ↑ |
|---|---|---|---|---|---|---|
| Input price G13 | Learned linear | Claude-3-5-Sonnet + Mistral-Large | Claude-3-Opus | 0.1797 | 0.1524 | 15.23% |
| Input price G13 | Learned log-odds | Claude-3-5-Sonnet + Mistral-Large | Claude-3-Opus | 0.1797 | 0.1733 | 3.58% |
| Open weights G13 | Learned linear | Mistral-Large + Qwen3-235B-A22B | Claude-3-Opus | 0.1797 | 0.1623 | 9.72% |
| Open weights G2 | Learned log-odds | Llama-3.3-70B + Llama-3.1-405B | O4-Mini | 0.1466 | 0.1395 | 4.87% |

# I Additional Category-Specific Results

We provide additional details for the category-specific analysis in Section 4.3. We first summarize the available data and comparison groups across all question categories, and then report the full group- and pair-level results for Politics and Finance, together with detailed group compositions and group-level comparisons.

**Category-specific data and comparison groups.** In Table 14, we summarize the available data after constructing comparison groups separately within each question category at the shared-coverage threshold $N = 1{,}000$. We note that the 6 categories do not cover all the subquestions as some subquestions cannot be categorized properly. Politics and Finance yield 12 and 9 eligible maximal groups, respectively, while Climate yields only two groups and Science, Sports, and Entertainment yield none. This limited coverage motivates our focus on Politics and Finance in Section 4.3. The table also reports the number of distinct forecasters and the ranges of group sizes and shared subquestions within each category

Table 14: Dataset statistics by question category at $N = 1{,}000$.

| Question Category | Category subquestions | Distinct forecasters | No. of maximal groups | Forecasters per group (min–max) | Shared subquestions per group (min–max) |
|---|---|---|---|---|---|
| Politics | 5,264 | 25 | 12 | 3–9 | 1,003–1,337 |
| Finance | 7,222 | 33 | 9 | 5–16 | 1,085–1,260 |
| Climate | 2,220 | 5 | 2 | 3–3 | 1,275–1,308 |
| Science | 459 | 0 | 0 | – | – |
| Sports | 1,300 | 0 | 0 | – | – |
| Entertainment | 512 | 0 | 0 | – | – |

**Aggregate performance within categories.** In Table 15, we report the complete group- and pair-level results for learned linear and learned log-odds pooling in Politics and Finance. For ECE, we retain the BS-selected benchmark, eligible pairs, and fitted coefficients, so the calibration results evaluate exactly the same aggregates rather than reselecting models or refitting the pooling rules. These results complement the summary in Section 4.3 by showing how frequently the category-level patterns appear across individual eligible pairs as well as across comparison groups.

**Group composition.** In Table 16, we report the complete membership of the Politics and Finance comparison groups used in the analysis. The groups differ in both size and model composition, and some forecasters appear in multiple groups. These details are useful for interpreting the category comparison because the Politics and Finance groups are constructed separately and therefore do not hold the set of forecasters fixed across categories. Accordingly, the differences reported in Section 4.3 should be interpreted descriptively rather than as a causal effect of question category.

Table 15: Pairwise aggregation within Politics and Finance at $N = 1{,}000$. Cells report group- and pair-level rates for aggregates that match or improve on the category-specific BS-best group benchmark under BS or ECE. For ECE, we keep the BS-selected group benchmark, eligible pairs, and fitted coefficients.

| | | BS: match or improve benchmark | | ECE: match or improve benchmark | |
|---|---|---|---|---|---|
| Question category | Aggregation rule | Groups | Pairs | Groups | Pairs |
| Politics | Learned linear pooling | 8/12 (66.67%) | 23/103 (22.33%) | 12/12 (100.00%) | 57/103 (55.34%) |
| | Learned log-odds pooling | 8/12 (66.67%) | 17/103 (16.50%) | 10/12 (83.33%) | 44/103 (42.72%) |
| Finance | Learned linear pooling | 9/9 (100.00%) | 235/287 (81.88%) | 8/9 (88.89%) | 281/287 (97.91%) |
| | Learned log-odds pooling | 5/9 (55.56%) | 31/287 (10.80%) | 5/9 (55.56%) | 42/287 (14.63%) |

Table 16: Complete membership of the size-sorted Politics and Finance groups at the 1,000-common-subquestion threshold.

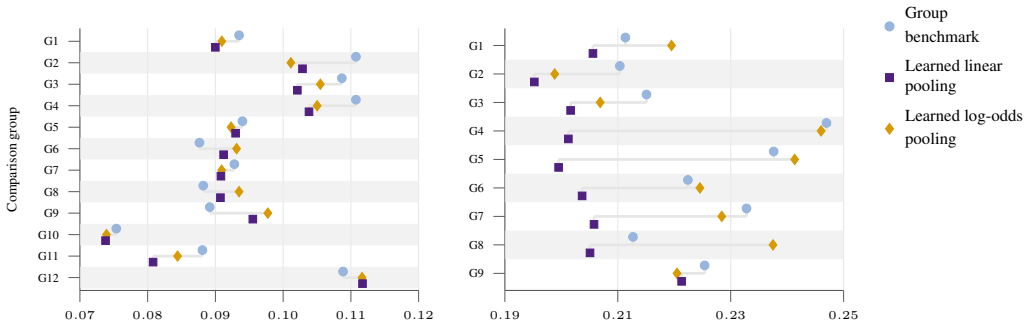
| Group | Size | Subquestions | Models |
|---|---|---|---|
| **Politics, threshold 1,000** | | | |
| 1 | 9 | 1,116 | Claude-3-5-Sonnet; Claude-3-7-Sonnet; DeepSeek-R1; DeepSeek-V3; GPT-4o; Llama-3.3-70B-Instruct-Turbo; Mistral-Large; O3-Mini; QwQ-32B-Preview |
| 2 | 8 | 1,035 | Claude-3-5-Sonnet; Claude-3-7-Sonnet; GPT-4.1; GPT-4o; Llama-3.3-70B-Instruct-Turbo; Llama-4-Maverick-17B-128E-Instruct-FP8; Llama-4-Scout-17B-16E-Instruct; Mistral-Large |
| 3 | 7 | 1,337 | Claude-3-5-Sonnet; Claude-3-Haiku; Claude-3-Opus; GPT-4-Turbo; GPT-4o; Mistral-Large; Qwen3-235B-A22B-Fp8-Tput |
| 4 | 6 | 1,129 | Claude-3-5-Sonnet; Claude-3-Haiku; Claude-3-Opus; GPT-4-Turbo; GPT-4o; Llama-3-70b-Chat-Hf |
| 5 | 5 | 1,032 | Claude-3-5-Sonnet; Claude-Sonnet-4; GPT-4.1; Kimi-K2-Instruct; Magistral-Medium |
| 6 | 5 | 1,011 | Claude-3-5-Sonnet; GPT-4.1; Llama-4-Maverick-17B-128E-Instruct-FP8; Llama-4-Scout-17B-16E-Instruct; O4-Mini |
| 7 | 5 | 1,003 | Claude-3-5-Sonnet; GPT-4.1; Kimi-K2-Instruct; Magistral-Medium; Qwen3-235B-A22B-Fp8-Tput |
| 8 | 4 | 1,313 | Claude-3-5-Sonnet; GPT-4.1; O4-Mini; Qwen3-235B-A22B-Fp8-Tput |
| 9 | 4 | 1,120 | Claude-Sonnet-4; GPT-4.1; Kimi-K2-Instruct; Qwen3-235B-A22B-Fp8-Tput |
| 10 | 4 | 1,071 | GPT-4.1; GPT-5-Mini; GPT-5-Nano; Gemini-2.5-Pro |
| 11 | 3 | 1,072 | Claude-Opus-4-1; GPT-4.1; Qwen3-235B-A22B-Fp8-Tput |
| 12 | 3 | 1,067 | Claude-3-7-Sonnet; GPT-4.1; Qwen3-235B-A22B-Fp8-Tput |
| **Finance, threshold 1,000** | | | |
| 1 | 16 | 1,170 | Claude-3-5-Sonnet; Claude-3-7-Sonnet; Claude-3-Haiku; Claude-3-Opus; GPT-4-Turbo; GPT-4.1; GPT-4o; Llama-3.2-3B-Instruct-Turbo; Llama-3.3-70B-Instruct-Turbo; Llama-4-Maverick-17B-128E-Instruct-FP8; Llama-4-Scout-17B-16E-Instruct; Meta-Llama-3.1-405B-Instruct-Turbo; Mistral-Large; O4-Mini; Qwen2.5-72B-Instruct-Turbo; Qwen3-235B-A22B-Fp8-Tput |
| 2 | 12 | 1,171 | Claude-3-5-Sonnet; Claude-3-7-Sonnet; DeepSeek-R1; DeepSeek-V3; GPT-4o; Gemini-2.0-Flash-Lite-001; Gemini-2.5-Pro-Exp; Grok-beta; Llama-3.3-70B-Instruct-Turbo; Mistral-Large; O3-Mini; QwQ-32B-Preview |
| 3 | 11 | 1,191 | Claude-3-5-Sonnet; Claude-3-7-Sonnet; DeepSeek-R1; DeepSeek-V3; GPT-4o; Llama-3.3-70B-Instruct-Turbo; Llama-4-Maverick-17B-128E-Instruct-FP8; Llama-4-Scout-17B-16E-Instruct; Mistral-Large; O3-Mini; QwQ-32B-Preview |
| 4 | 8 | 1,182 | Claude-3-5-Sonnet; Claude-3-Haiku; Claude-3-Opus; GPT-4-Turbo; GPT-4o; Llama-3-70b-Chat-Hf; Mistral-Large; Qwen3-235B-A22B-Fp8-Tput |
| 5 | 7 | 1,230 | Claude-3-5-Sonnet; Claude-3-Haiku; Claude-3-Opus; GPT-4-Turbo; GPT-4o; Llama-3-70b-Chat-Hf; Llama-3-8b-Chat-Hf |
| 6 | 7 | 1,174 | Claude-3-5-Sonnet; Claude-Sonnet-4; GPT-4.1; Kimi-K2-Instruct; Magistral-Medium; O4-Mini; Qwen3-235B-A22B-Fp8-Tput |
| 7 | 7 | 1,157 | Claude-3-5-Sonnet; Claude-Opus-4-1; Claude-Sonnet-4; GPT-4.1; Kimi-K2-Instruct; Magistral-Medium; Qwen3-235B-A22B-Fp8-Tput |
| 8 | 6 | 1,260 | Claude-3-5-Sonnet; Claude-3-7-Sonnet; DeepSeek-R1; DeepSeek-V3; GPT-4.5-Preview; Llama-3.3-70B-Instruct-Turbo |
| 9 | 5 | 1,085 | Claude-Opus-4-1; GPT-4.1; GPT-5-Mini; GPT-5-Nano; Gemini-2.5-Pro |

**Best weak-pair aggregates by group.** In Table 17, we provide the detailed group-level comparisons underlying the category-specific results. For each group, we report the BS-best individual benchmark and the lowest-test-BS weak pair under learned linear and learned log-odds pooling, together with the fitted coefficients and bootstrap confidence intervals. This allows us to examine not only whether a category contains successful weak-to-strong aggregates, but also how the identity and performance of the best pair vary across groups.

Table 17: Group benchmark and the best weak-pair aggregate under each pooling method for each of the size-sorted Politics and Finance groups at the 1,000-common-subquestion threshold, selected by test Brier score.

| Group | Size | Group Benchmark | Brier (95% CI) | Log-odds pooling best pair | Brier (95% CI) | Linear pooling best pair | Brier (95% CI) |
|---|---|---|---|---|---|---|---|
| **Politics, threshold 1,000** | | | | | | | |
| 1 | 9 | O3-Mini | 0.093 ($\pm$ 0.017) | Mistral-Large + QwQ-32B-Preview $(\omega_1, \omega_2) = (0.5087, 0.5308)$ | **0.091** ($\pm$ 0.013) | Mistral-Large + QwQ-32B-Preview $(\omega_1, \omega_2) = (0.5694, 0.4300)$ | **0.090** ($\pm$ 0.012) |
| 2 | 8 | Claude-3-5-Sonnet | 0.111 ($\pm$ 0.017) | Claude-3-7-Sonnet + GPT-4.1 $(\omega_1, \omega_2) = (0.4040, 0.5103)$ | **0.101** ($\pm$ 0.015) | GPT-4.1 + Mistral-Large $(\omega_1, \omega_2) = (0.5201, 0.3794)$ | **0.103** ($\pm$ 0.016) |
| 3 | 7 | Claude-3-5-Sonnet | 0.109 ($\pm$ 0.014) | Claude-3-Opus + Mistral-Large $(\omega_1, \omega_2) = (0.6572, 0.4858)$ | **0.105** ($\pm$ 0.013) | Claude-3-Opus + Mistral-Large $(\omega_1, \omega_2) = (0.4971, 0.4154)$ | **0.102** ($\pm$ 0.011) |
| 4 | 6 | Claude-3-5-Sonnet | 0.111 ($\pm$ 0.017) | Claude-3-Opus + Llama-3-70b-Chat-Hf $(\omega_1, \omega_2) = (0.8566, 0.2804)$ | **0.105** ($\pm$ 0.014) | Claude-3-Opus + Llama-3-70b-Chat-Hf $(\omega_1, \omega_2) = (0.7538, 0.1329)$ | **0.104** ($\pm$ 0.012) |
| 5 | 5 | GPT-4.1 | 0.094 ($\pm$ 0.017) | Kimi-K2-Instruct + Magistral-Medium $(\omega_1, \omega_2) = (0.5169, 0.5702)$ | **0.092** ($\pm$ 0.014) | Claude-3-5-Sonnet + Magistral-Medium $(\omega_1, \omega_2) = (0.6958, 0.2150)$ | **0.093** ($\pm$ 0.013) |
| 6 | 5 | O4-Mini | 0.088 ($\pm$ 0.012) | Claude-3-5-Sonnet + GPT-4.1 $(\omega_1, \omega_2) = (0.7774, 0.3183)$ | 0.093 ($\pm$ 0.014) | Claude-3-5-Sonnet + GPT-4.1 $(\omega_1, \omega_2) = (0.6453, 0.2540)$ | 0.091 ($\pm$ 0.012) |
| 7 | 5 | GPT-4.1 | 0.093 ($\pm$ 0.018) | Kimi-K2-Instruct + Magistral-Medium $(\omega_1, \omega_2) = (0.5071, 0.5840)$ | **0.091** ($\pm$ 0.014) | Claude-3-5-Sonnet + Magistral-Medium $(\omega_1, \omega_2) = (0.6966, 0.2022)$ | **0.091** ($\pm$ 0.013) |
| 8 | 4 | O4-Mini | 0.088 ($\pm$ 0.010) | Claude-3-5-Sonnet + GPT-4.1 $(\omega_1, \omega_2) = (0.8719, 0.2802)$ | 0.093 ($\pm$ 0.013) | Claude-3-5-Sonnet + GPT-4.1 $(\omega_1, \omega_2) = (0.7033, 0.2140)$ | 0.091 ($\pm$ 0.011) |
| 9 | 4 | GPT-4.1 | 0.089 ($\pm$ 0.016) | Kimi-K2-Instruct + Qwen3-235B-A22B-Fp8-Tput $(\omega_1, \omega_2) = (0.5563, 0.6586)$ | 0.098 ($\pm$ 0.014) | Kimi-K2-Instruct + Qwen3-235B-A22B-Fp8-Tput $(\omega_1, \omega_2) = (0.4537, 0.4680)$ | 0.096 ($\pm$ 0.012) |
| 10 | 4 | GPT-5-Mini | 0.075 ($\pm$ 0.014) | GPT-4.1 + Gemini-2.5-Pro $(\omega_1, \omega_2) = (0.3139, 0.7821)$ | **0.074** ($\pm$ 0.014) | GPT-4.1 + Gemini-2.5-Pro $(\omega_1, \omega_2) = (0.2453, 0.7142)$ | **0.074** ($\pm$ 0.013) |
| 11 | 3 | GPT-4.1 | 0.088 ($\pm$ 0.017) | Claude-Opus-4-1 + Qwen3-235B-A22B-Fp8-Tput $(\omega_1, \omega_2) = (0.5158, 0.4456)$ | **0.084** ($\pm$ 0.015) | Claude-Opus-4-1 + Qwen3-235B-A22B-Fp8-Tput $(\omega_1, \omega_2) = (0.4742, 0.3733)$ | **0.081** ($\pm$ 0.014) |
| 12 | 3 | GPT-4.1 | 0.109 ($\pm$ 0.018) | Claude-3-7-Sonnet + Qwen3-235B-A22B-Fp8-Tput $(\omega_1, \omega_2) = (0.5874, 0.5654)$ | 0.112 ($\pm$ 0.016) | Claude-3-7-Sonnet + Qwen3-235B-A22B-Fp8-Tput $(\omega_1, \omega_2) = (0.3616, 0.4461)$ | 0.112 ($\pm$ 0.015) |
| **Finance, threshold 1,000** | | | | | | | |
| 1 | 16 | Llama-3.3-70B-Instruct-Turbo | 0.211 ($\pm$ 0.011) | Llama-4-Scout-17B-16E-Instruct + O4-Mini $(\omega_1, \omega_2) = (0.3082, 0.8885)$ | 0.220 ($\pm$ 0.011) | Claude-3-7-Sonnet + Llama-4-Scout-17B-16E-Instruct $(\omega_1, \omega_2) = (0.3486, 0.3024)$ | **0.206** ($\pm$ 0.013) |
| 2 | 12 | Llama-3.3-70B-Instruct-Turbo | 0.210 ($\pm$ 0.011) | DeepSeek-R1 + QwQ-32B-Preview $(\omega_1, \omega_2) = (0.2377, 0.3719)$ | **0.199** ($\pm$ 0.012) | O3-Mini + QwQ-32B-Preview $(\omega_1, \omega_2) = (0.3449, 0.4018)$ | **0.195** ($\pm$ 0.014) |
| 3 | 11 | Llama-3.3-70B-Instruct-Turbo | 0.215 ($\pm$ 0.012) | DeepSeek-R1 + QwQ-32B-Preview $(\omega_1, \omega_2) = (0.4337, 0.2798)$ | **0.207** ($\pm$ 0.012) | DeepSeek-R1 + QwQ-32B-Preview $(\omega_1, \omega_2) = (0.5413, 0.2715)$ | **0.202** ($\pm$ 0.013) |
| 4 | 8 | Claude-3-Opus | 0.247 ($\pm$ 0.010) | Claude-3-5-Sonnet + Mistral-Large $(\omega_1, \omega_2) = (0.1765, 0.1133)$ | **0.246** ($\pm$ 0.003) | Claude-3-5-Sonnet + GPT-4o $(\omega_1, \omega_2) = (0.5378, 0.0000)$ | **0.201** ($\pm$ 0.012) |
| 5 | 7 | Claude-3-Opus | 0.238 ($\pm$ 0.009) | Claude-3-5-Sonnet + GPT-4-Turbo $(\omega_1, \omega_2) = (0.1751, 0.1803)$ | 0.241 ($\pm$ 0.004) | Claude-3-5-Sonnet + Llama-3-70b-Chat-Hf $(\omega_1, \omega_2) = (0.4631, 0.0836)$ | **0.200** ($\pm$ 0.011) |
| 6 | 7 | O4-Mini | 0.222 ($\pm$ 0.009) | GPT-4.1 + Magistral-Medium $(\omega_1, \omega_2) = (0.4252, 0.6643)$ | 0.225 ($\pm$ 0.010) | Claude-3-5-Sonnet + GPT-4.1 $(\omega_1, \omega_2) = (0.4053, 0.1665)$ | **0.204** ($\pm$ 0.012) |
| 7 | 7 | Magistral-Medium | 0.233 ($\pm$ 0.010) | Claude-Sonnet-4 + GPT-4.1 $(\omega_1, \omega_2) = (0.1522, 0.4135)$ | **0.228** ($\pm$ 0.008) | Claude-3-5-Sonnet + GPT-4.1 $(\omega_1, \omega_2) = (0.4321, 0.1317)$ | **0.206** ($\pm$ 0.012) |
| 8 | 6 | Llama-3.3-70B-Instruct-Turbo | 0.213 ($\pm$ 0.010) | Claude-3-7-Sonnet + GPT-4.5-Preview $(\omega_1, \omega_2) = (0.1910, 0.3798)$ | 0.237 ($\pm$ 0.005) | Claude-3-5-Sonnet + Claude-3-7-Sonnet $(\omega_1, \omega_2) = (0.3737, 0.1702)$ | **0.205** ($\pm$ 0.011) |
| 9 | 5 | GPT-4.1 | 0.225 ($\pm$ 0.011) | Claude-Opus-4-1 + Gemini-2.5-Pro $(\omega_1, \omega_2) = (0.5254, 0.3201)$ | **0.220** ($\pm$ 0.009) | Claude-Opus-4-1 + Gemini-2.5-Pro $(\omega_1, \omega_2) = (0.4559, 0.3650)$ | **0.221** ($\pm$ 0.010) |

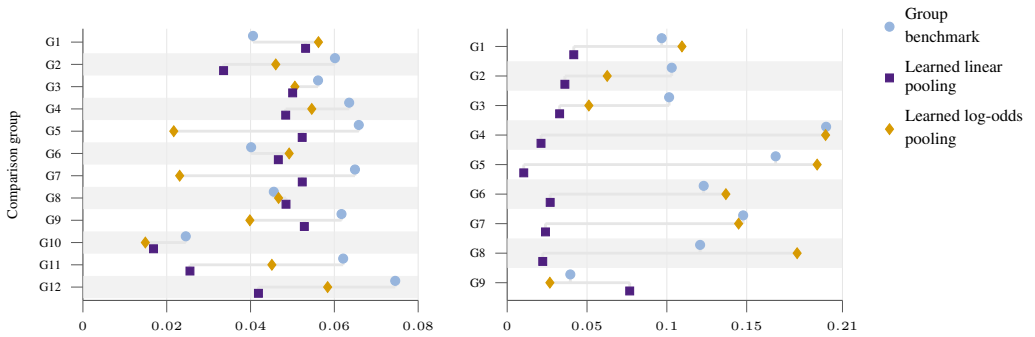
In Figure 10, we visualize the comparisons in Table 17. The figure makes the contrast between the two categories more transparent: learned linear pooling performs consistently close to or better than the benchmark across the Finance groups, whereas the Politics results are more heterogeneous. The performance of learned log-odds pooling also varies more substantially across groups in both categories. Because the groups differ in both questions and constituent forecasters, these patterns should again be interpreted as descriptive differences across the resulting comparison groups.




(a) Politics: BS. (b) Finance: BS.

Figure 10: Test BS of the BS-best group benchmark and the lowest-test-BS pair under each learned rule in Figure 10a Politics and Figure 10b Finance. We select the pair separately for learned linear and learned log-odds pooling in each category-specific comparison group. Lower BS is better. The pale horizontal segments connect the three point estimates within a group; they are not confidence intervals. Group labels are local to each category and ordered by decreasing group size.

**Calibration of the BS-selected aggregates.** Lastly, in Figure 11, we evaluate the same group benchmarks and rule-specific weak pairs selected by test BS in Figure 10 using ECE, without reselection or refitting. We observe a broadly similar category pattern for calibration: learned linear pooling performs particularly well in Finance, where the pair-level ECE success rate is substantially higher than in Politics. This mirrors the corresponding BS comparison and suggests that the stronger aggregation gains observed in Finance are generally accompanied by improved calibration as well.




(a) Politics: ECE. (b) Finance: ECE.

Figure 11: Test ECE of the BS-best individual and the learned linear and log-odds pairs in the Politics and Finance comparison groups.